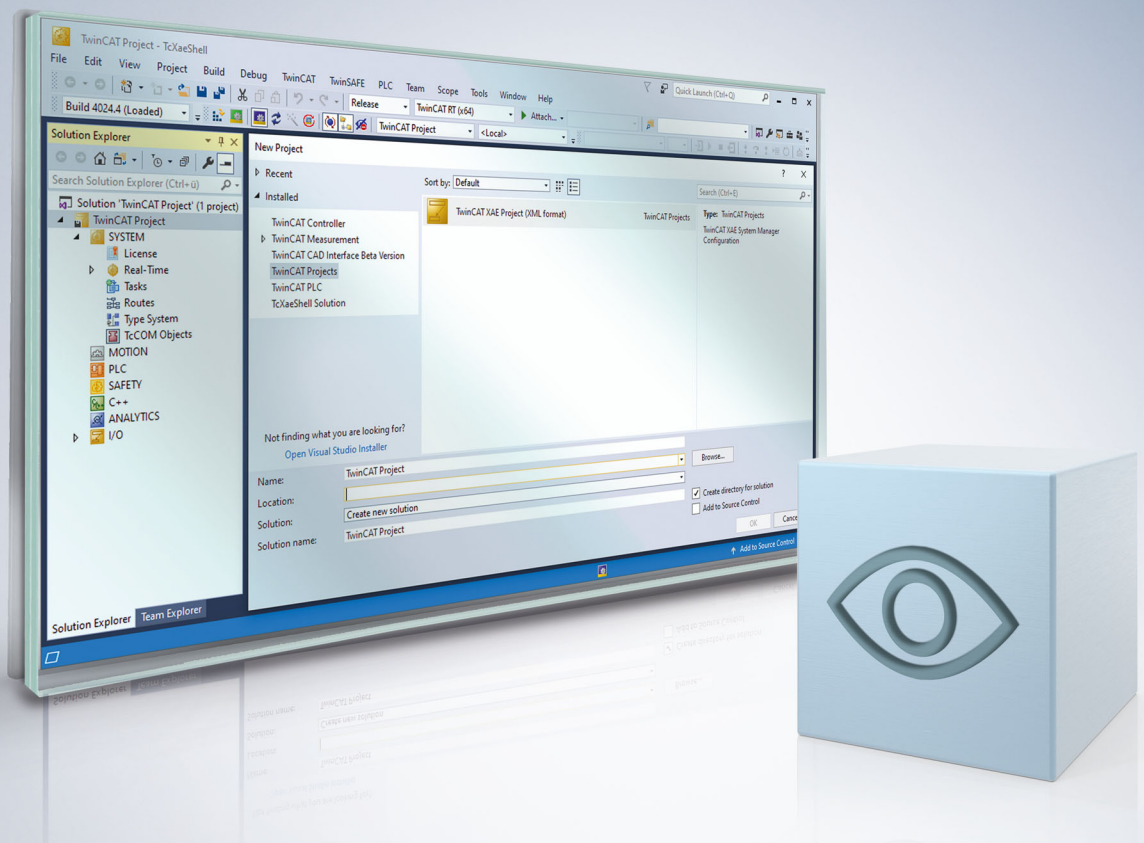
BECKHOFF New Automation Technology

Manual | EN

TF7000 - TF7810

TwinCAT 3 | Vision



2024-12-04 | Version: 2.2.0

Table of contents

1 Foreword 7
1.1 Notes on the documentation 7
1.2 For your safety 8
1.3 Notes on information security 9
1.4 Documentation issue status 9

2 Overview 10
2.1 System components 10
2.2 Licensing model 12

3 Installation 15
3.1 System requirements 15
3.2 Version overview 16
3.3 Installation instructions TwinCAT 3.1 Build 4024 17
3.4 Installation instructions from TwinCAT 3.1 Build 4026 23
3.5 Licensing 24

4 First steps 27
4.1 Build Project 27
4.1.1 Creating a TwinCAT project 27
4.1.2 Choose Target System 29
4.1.3 System configuration 31
4.2 Setting up a Vision device 33
4.2.1 Displaying a Vision node and creating an application 33
4.2.2 Creating a File Source Control 35
4.2.3 Creating a GigE Vision camera 38
4.3 Creating a PLC project 45
4.3.1 Creating a PLC project 45
4.3.2 Integrating a PLC library 46
4.3.3 Writing a PLC program 48
4.3.4 Initializing a function block 49
4.4 Run 50
4.4.1 Activate TwinCAT project and start PLC project 50
4.4.2 Image display in ADS Image Watch 51

5 Development environment 53
5.1 System configuration 54
5.1.1 Router memory 54
5.1.2 Stack size 56
5.1.3 Network adapter 56
5.1.4 CPU cores and tasks 61
5.1.5 Job tasks 64
5.2 VISION node 65
5.2.1 Service Configuration 66
5.2.2 Logging 68
5.3 Application node 68
5.4 GigE Vision Camera objects 69

5.4.1 General ... 71
5.4.2 Configuration Assistant ... 75
5.4.3 Record/Playback ... 101
5.4.4 Camera Calibration ... 107
5.4.5 GigE Vision Standard ... 122
5.4.6 TcCOM objects ... 123
5.5 Beckhoff Camera objects ... 125
5.6 File Source objects ... 125
5.6.1 File Source Control ... 126
5.6.2 TcCOM objects ... 130
5.7 Vision Job Pool ... 131
5.8 ADS Image Watch ... 131
5.9 Exporting/Importing nodes ... 135

6 API reference ... 137
6.1 PLC ... 137
6.1.1 Software concept ... 137
6.1.2 Data Types ... 159
6.1.3 Interfaces ... 246
6.1.4 Functions ... 433
6.1.5 Function Blocks ... 1570
6.2 C++ ... 1659
6.2.1 Data Types ... 1659
6.2.2 Interfaces ... 1731
6.2.3 Functions ... 1890
6.2.4 Function Blocks ... 2653

7 TwinCAT HMI packages ... 2704
7.1 Server extension ... 2704
7.2 Controls ... 2706
7.2.1 Image Watch ... 2707
7.2.2 Color ... 2731
7.2.3 Rectangle (Interactive) ... 2744
7.2.4 Polygon (Interactive) ... 2746
7.2.5 Help Functions ... 2747
7.2.6 Data types ... 2750

8 Samples and tools ... 2753
8.1 Function samples ... 2754
8.1.1 Basic Container Operations ... 2755
8.1.2 Basic Image Operations ... 2757
8.1.3 Code Reading ... 2759
8.1.4 Contour analysis ... 2771
8.1.5 Image analysis ... 2780
8.1.6 Image Color and Contrast Processing ... 2800
8.1.7 Image filtering ... 2805
8.1.8 Image segmentation ... 2810
8.1.9 Measurement ... 2813

8.1.10 Self-written functions ... 2830
8.2 Function block samples ... 2832
8.2.1 Camera Register Access ... 2832
8.2.2 File Access ... 2834
8.2.3 Image acquisition ... 2836
8.3 Samples of assistants ... 2838
8.3.1 Calibration Assistant ... 2838
8.4 Camera configuration samples ... 2845
8.4.1 Region of Interest (ROI) ... 2845
8.4.2 Binning ... 2846
8.4.3 Image acquisition and trigger ... 2849
8.5 Neural Network Compatibility Check ... 2855
9 Appendix ... 2859
9.1 FAQ ... 2859
9.2 Important paths ... 2860
9.3 Troubleshooting ... 2861
9.3.1 TwinCAT system start ... 2861
9.3.2 PLC runtime environment ... 2865
9.3.3 Assistants ... 2872
9.3.4 TwinCAT HMI packages ... 2876
9.3.5 Camera communication ... 2877
9.4 ADS Return Codes ... 2879
9.5 Support and Service ... 2883
9.6 Third party Components ... 2885

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all requirements for safety, including all the relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
No claims to modify products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT®

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document as well as the use and communication of its contents without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product



This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

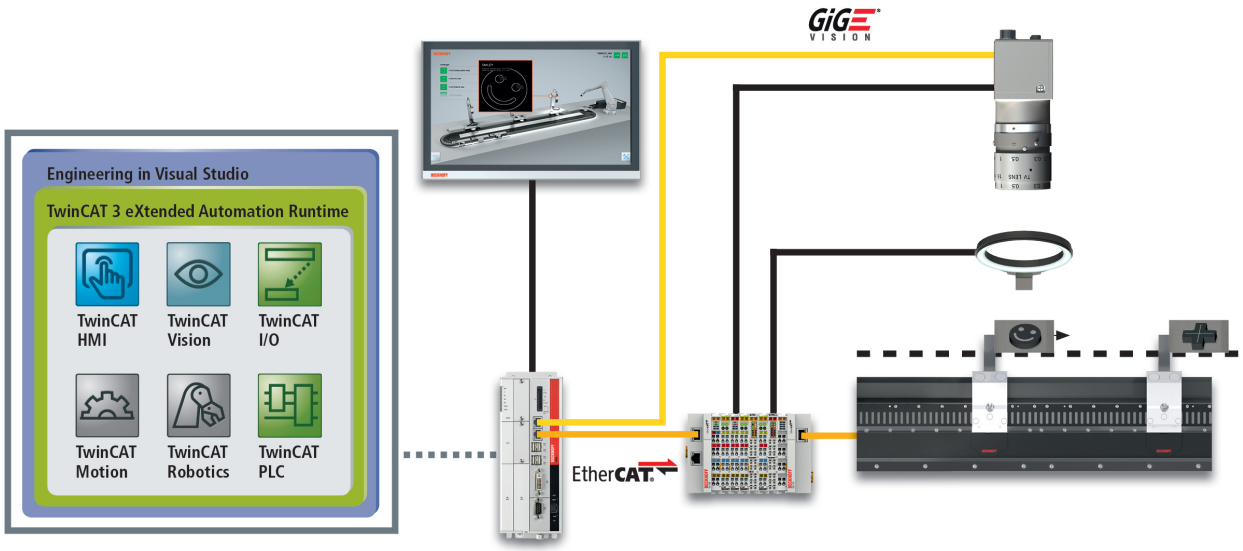
Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

1.4 Documentation issue status

Version	Change
2.2.x	**New:** Neural Network und Neural Network C++
2.1.x	**New:** various functions, functions Code Quality [▶ 818] and Code Quality C++ [▶ 2196] **New:** Installation instructions for TwinCAT 3.1 Build 4026 [▶ 23]
2.0.x	**New:** various function blocks, functions, data types and interfaces [▶ 137] **New:** API for C++ [▶ 1659]
1.5.x	**New:** various function blocks and functions

2 Overview



TwinCAT Vision is a TwinCAT 3 function for machine vision. Tasks such as the detection, identification, and measurement of objects can be handled directly in the PLC in real-time. By integrating image processing into the TwinCAT platform, highly synchronized control applications and extremely short response times can be achieved. Application development is aided by the large number of existing TwinCAT Engineering tools.

Image processing sequences are programmed in the IEC61131-3 languages using the Tc3_Vision library directly in the PLC code. Alternatively, programming can also be carried out in TwinCAT 3 C++. Intermediate results can be checked directly using an image display tool.

TwinCAT Vision is real-time capable because the image processing algorithms are executed synchronously with the controller directly in the TwinCAT 3 runtime environment. In addition, suitable algorithms can run automatically in parallel on several cores.

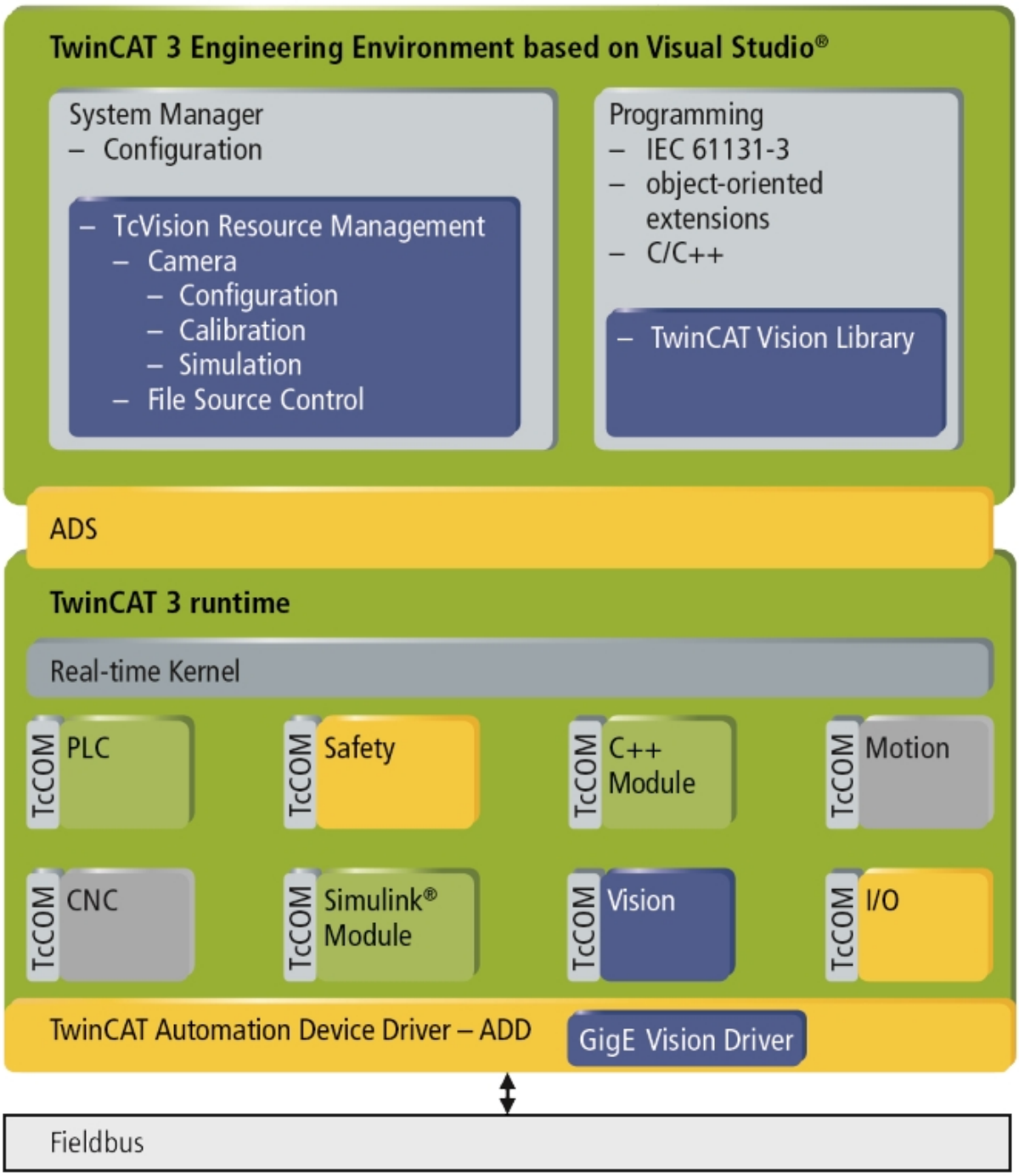
A large number of industrial cameras can be connected via the GigE Vision interface. The familiar TwinCAT development environment is used for configuring and calibrating cameras, and for system setup and debugging.

The individual chapters provide detailed information on the following topics:

- System components [▶ 10] for understanding the system structure of TwinCAT Vision
- Installation [▶ 15] for guidance on installation and licensing
- First steps [▶ 27] for a quick introduction to TwinCAT Vision
- Development environment [▶ 53] for configuration, calibration and simulation of cameras
- API reference [▶ 137] (software documentation)
- Examples [▶ 2753] of some image processing sequences
- Notes [▶ 2861] on troubleshooting

2.1 System components

Like other TwinCAT components, TwinCAT Vision is integrated into the Visual Studio-based TwinCAT development environment and the real-time capable TwinCAT runtime environment. The components of the TwinCAT Vision system are explained below.



Configuration Assistants

- Industrial cameras can be connected via the GigE Vision [▶ 122] interface. Assistants [▶ 69] are available for configuration, calibration and simulation of the cameras.
- Images can be fed from the file system into the PLC runtime via a File Source [▶ 125] element.
- The configuration [▶ 54] of the cores and the router memory for TwinCAT Vision must always be carried out in the TwinCAT real-time settings.
- TwinCAT Vision can utilize TwinCAT's multi-core support to execute certain algorithms in parallel. Corresponding settings are made in the Vision Job Pool [▶ 131].

Library

TwinCAT Vision provides a library with a large number of algorithms for programming image processing sequences. It also contains function blocks that serve as interfaces for cameras and the file system. The API reference [▶ 137] contains detailed descriptions of all library elements.

Debugging tools

TwinCAT Vision provides the following mechanisms for debugging image processing sequences:

- ADS Image Watch [▶ 131] can be used for live visualization of camera images and intermediate results. It transfers images from the PLC runtime to the development environment via ADS and displays them there.
- All TwinCAT Vision functions provide information on their execution status based on ADS Return Codes [▶ 2879] (see HRESULT [▶ 141]).
- All TwinCAT debugging functions such as breakpoints etc. can be used for programming.

GigE Vision driver

TwinCAT Vision includes a real-time capable GigE Vision [▶ 122] driver. It is used in the development environment for camera configuration and in the TwinCAT runtime. It communicates with industrial cameras via UDP/IP, based on the GigE Vision standard. This standard is used to find and configure cameras in the network and to receive image data from the cameras.

TwinCAT Vision service

For some functionalities, TwinCAT Vision has to communicate with components outside the real-time at runtime. For example, a File Source Control loads images from the file system. The TwinCAT Vision Service [▶ 66] is a centralized, ADS-based communication interface between the TwinCAT runtime and external Vision components.

TwinCAT HMI Vision Extension

The TwinCAT HMI Vision Extension [▶ 2704] is an extension for the TwinCAT HMI that enables transfer of images from the PLC to the TwinCAT HMI. Images from the PLC can be displayed directly in the web-based HMI with HmiImageControl.

TwinCAT HMI Vision Controls

The TwinCAT HMI Vision Controls [▶ 2706] package provides vision specific controls as an extension for the TwinCAT HMI. For example, it includes a control that provides many more tools and information displays in addition to the image display. The functions and displays can be easily adapted by configuration and used directly without additional programming.

2.2 Licensing model

The TF7xxx products for TwinCAT Vision comprise licenses for camera connections and an image processing library. To generate licenses, please follow these instructions [▶ 24]. For testing or development with TwinCAT Vision, you can create licenses for a 7-day trial version.

For currently available product licenses, please refer to the product website.

TF700x | TwinCAT 3 GigE Vision Connector

The TwinCAT 3 GigE Vision Connector enables the integration of GigE Vision cameras as well as Beckhoff cameras. The basic TF7000 license is always required and enables connection of two cameras. If required, the capacity can be increased with expansion licenses:

TF7000	Basic license incl. 2 camera connections (always required)
TF7001	Extension for 2 additional connections
TF7002	Extension for 4 additional connections
TF7003	Extension for 8 additional connections

The extension licenses can be combined so that a maximum of 2+2+4+8=16 cameras can be connected.

For example, the following licenses are required for 11 camera connections:

- TF7000 base (including 2 camera connections)
- TF7001 2 Camera
- TF7003 8 Camera

→ 12 cameras can now be connected in total.

TF7020 | TwinCAT 3 Vision Beckhoff Camera Connector

The TwinCAT 3 Vision Beckhoff Camera Connector enables the integration of up to 64 Beckhoff cameras. Cameras from other manufacturers cannot be used with this license.

TF7100-TF7810 | TwinCAT 3 Vision Algorithms

The TF7100-TF7810 product licenses provide image processing functions in the "Tc3_Vision" library. The basic TF7100 license is required for any use of the library. It provides access to basic algorithms such as filter operations, format conversions and contour tracing. This already allows many applications to be implemented. The extension licenses supplement the range of functions as described in the table. The assignment of functions to licenses can be found in the API reference [▶ 435].

TF7100	Basic license for the "Tc3_Vision" library. Contains algorithms for the solving of image processing tasks, such as algebraic operations, filters, Fourier analysis, color image processing, segmentation, contour and blob analysis, results display and for the reading and writing of camera parameters.
TF7200	Extension for Matching 2D: Objects can be found and compared on the basis of learned references, contours, feature points or other properties (Template Matching / KeyPoint Detection and Descriptor Matching). A common application is the classification of objects.
TF7250	Extension for Code Reading: Detection and reading of 1D and 2D codes.
TF7255	Extension for Code Quality: Quality assessment of 1D and 2D codes.
TF7260	Extension for OCR: Functions for optical character recognition that identify characters in an image region and return the recognized character string.
TF7300	Extension for Metrology 2D: Tools for optical measurement of geometric object properties (distances, radii etc.). • Edge localization on the measured images with subpixel accuracy • Compensation of optical distortions (prior calibration of the camera necessary!) • Transformation between pixel coordinates and real-world coordinates (prior calibration of the camera necessary!)
TF7800	Extension for Machine Learning: Functions for the training and execution of various classic ML models.
TF7810	Extension for Neural Network: Functions for executing trained neural networks in Open Neural Network Exchange (ONNX) format.

Automatic determination of the required Vision library licenses

The required licenses are only determined when compiling the respective TwinCAT project. At the same time, a check is carried out to see which function blocks are declared and which functions are used in the code.

Therefore, please note that required licenses may be need to be added or omitted in the case of project changes (at short notice). If new licenses are added, the change must be loaded by means of an **Activate Configuration**; an online change is not possible in this case.

3 Installation

This chapter contains the system requirements [▶ 15] for TwinCAT Vision, the version overview [▶ 16] and instructions for installation [▶ 17] and licensing [▶ 24].

3.1 System requirements

IPC hardware

Technical data	Description
TwinCAT 3 platform level	Minimum P50 Performance Plus, e.g. Intel 4-core Atom CPU
Network adapter	TwinCAT real-time Ethernet-compatible Gigabit network cards (for connecting GigE Vision cameras) **At least one network port should be available exclusively for cameras. It is recommended to use a separate network port for each camera.**
Main memory	Minimum 4 GB recommended for pure runtime systems Minimum 8 GB recommended for development systems Minimum 8 GB recommended for the use of record/playback [▶ 101] streams on runtime systems
Hard disk	SSD hard disks with high reading/writing rates are recommended, especially if a lot of data is to be exchanged between PLC and hard disk.

Camera

Technical data	Description
Interface	Must support GigE Vision Standard [▶ 122]. (usually marked with the GigE Vision logo) **The TwinCAT GigE Vision Connector is certified according to the GigE Vision Standard version 2.0. However, this does not mean that all camera or GigE Vision features of the version are supported.**

Software

Technical data	Description
Operating system	Windows 10 (only 64-bit!)
Supported Visual Studio versions	2017 (Community, Professional) 2019 (Community, Professional) 2022 (Community, Professional) from TwinCAT 3.1 Build 4026 TwinCAT XAE Shell (TcXaeShell) TwinCAT XAE Shell (TcXaeShell64) from TwinCAT 3.1 Build 4026
TwinCAT	Minimum version 3.1.4024.59 Minimum version 3.1.4026.5
.NET Framework	Minimal version 4.7.2 (only for the development environment [▶ 53])

TwinCAT HMI Server extension

Technical data	Description
.NET	Minimum version .NET Desktop Runtime 6.0

3.2 Version overview

TwinCAT 3.1.4024 Setup	TwinCAT 3.1.4026 Work-load	Comment
4.0.6.5	5.6.5	Neural Network added.
4.0.5.10	5.5.10	Beckhoff Camera Connector and Code Quality added.
4.0.4.8	-	C++ API, OCR models and Machine Learning added.
4.0.3.5	-	Automatic check of initialization commands during compilation process or activation of configuration was removed.
4.0.2.13	-	
4.0.1.3	-	Repository files are deleted with the installation of a newer setup. **If you want to keep this version, save the "%TWINCAT3DIR%\3.1\Repository\Beckhoff Automation GmbH" folder before installing a new setup and copy it back after the installation.**

NOTICE

Compatibility

If you have several TwinCAT Vision versions installed on one system or if the development and runtime environment are on different systems, you must ensure that you use all components with the same version.

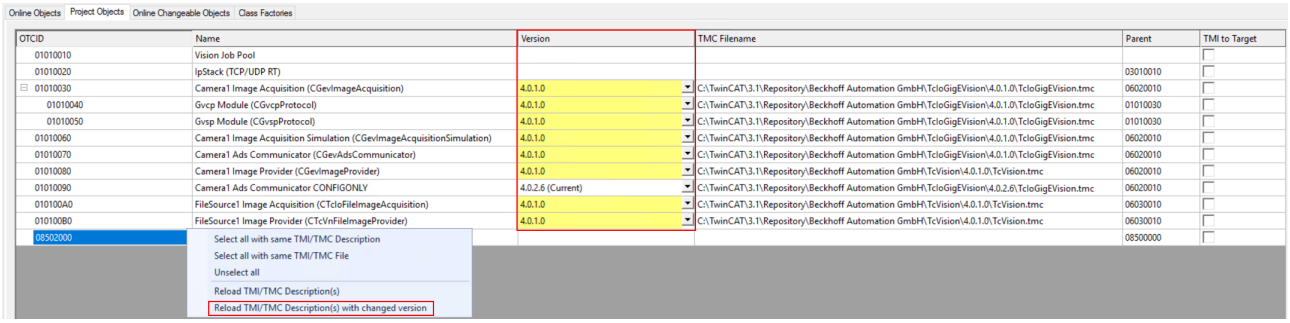
The TwinCAT Vision Service and the Engineering Extensions are only available in the latest version. Therefore, these components determine the TcCOM and library version to be used. Mixing versions is not permitted and can lead to malfunctions.

Hints for the selection of a version

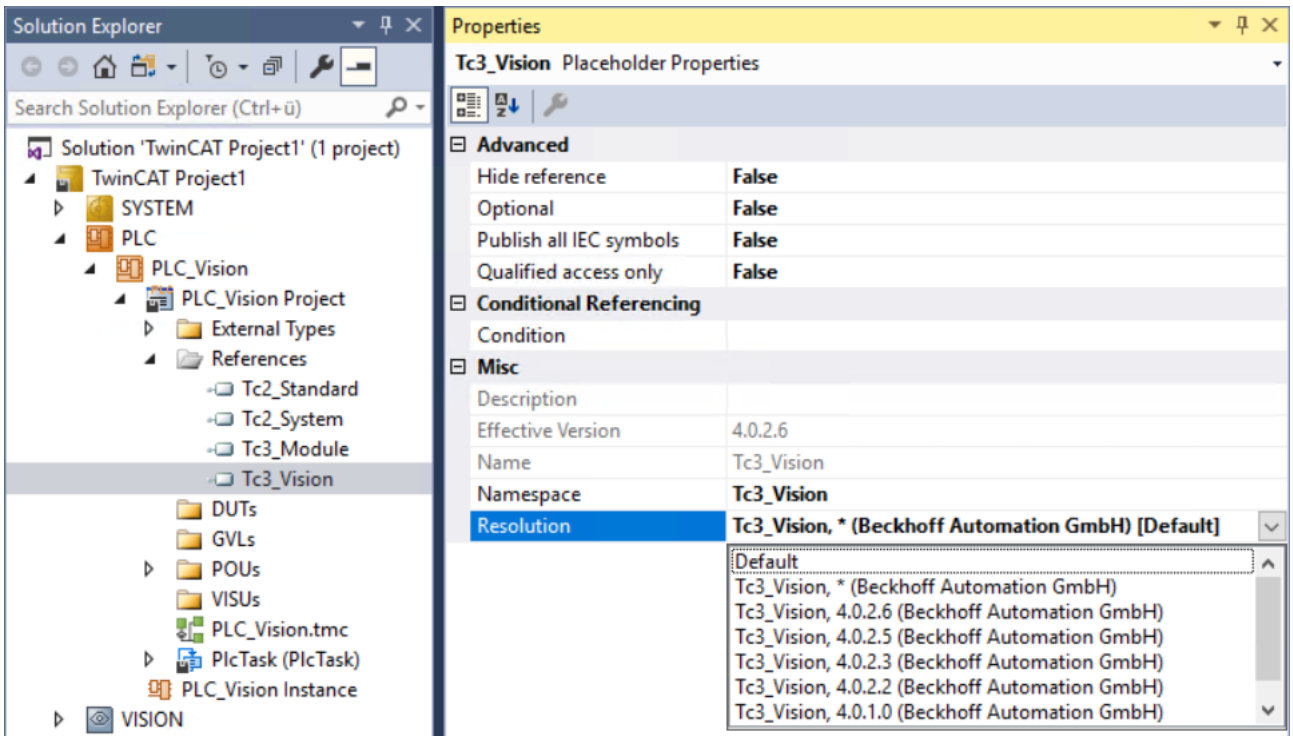
If you want to use TwinCAT Vision with a specific version, all TcCOM objects of the project must be updated to the corresponding version. An exception is the Ads Communicator CONFIGONLY. This module is only used internally in the Configuration Assistant and must always have the latest version that matches the extension. Therefore, this module automatically sets the latest version when loading a project.

To check or change the TcCOM versions:

1. Open the window **SYSTEM > TcCOM Objects > Project Objects**.
2. To change, select the desired version for all objects.
3. Right-click on one of the object IDs in the first column.
4. Select **Reload TMI/TMC Description(s) with changed version**.



Furthermore, the same version of the Tc3_Vision library, corresponding to the selected TcCOM object version, must be used in the PLC project. Therefore check in the properties of the Tc3_Vision library which version is effectively used or adjust it accordingly.



The current version number of the TwinCAT Vision Service and thus also of the Engineering Extensions can be found in the Service Configuration [▶ 66] under the Vision node.

NOTICE

Online Change after update

If you update TwinCAT Vision on a development system and thus make an Online Change on a target system with an older version, this can lead to version conflicts with interface pointers (images and containers).

Deleting older versions

To delete older versions starting with 4.x.x.x, two steps are necessary:

1. First you have to delete the corresponding version folders in the Tc3_Vision, TcIoGigEVision and TcVision directories under "%TWINCAT3DIR%\3.1\Repository\Beckhoff Automation GmbH" or under "C:\ProgramData\Beckhoff\TwinCAT\3.1\Repository\Beckhoff Automation GmbH".
2. To remove the corresponding PLC library versions, you must also uninstall them using the Library Repository dialog.

To delete older versions that start with 5.x.x, you must uninstall the corresponding packages by specifying the version using the following commands.

- To uninstall from the TcIoGigEVision repository:

```
tcpkg uninstall TwinCAT.XAE.TMX.GigEVision=5.x.x
```

- To uninstall from the Tc3_Vision and TcVision repositories:

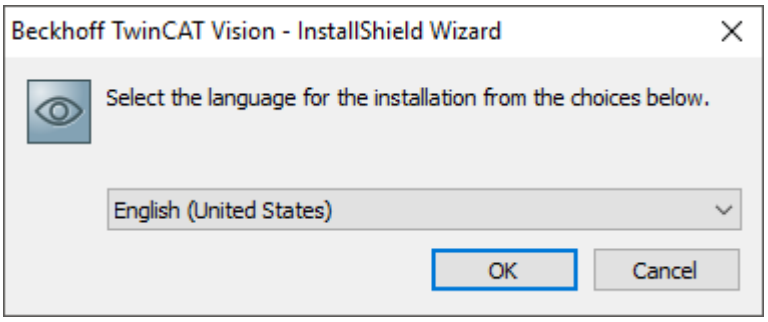
```
tcpkg uninstall TwinCAT.XAE.TMX.Vision=5.x.x
```

3.3 Installation instructions TwinCAT 3.1 Build 4024

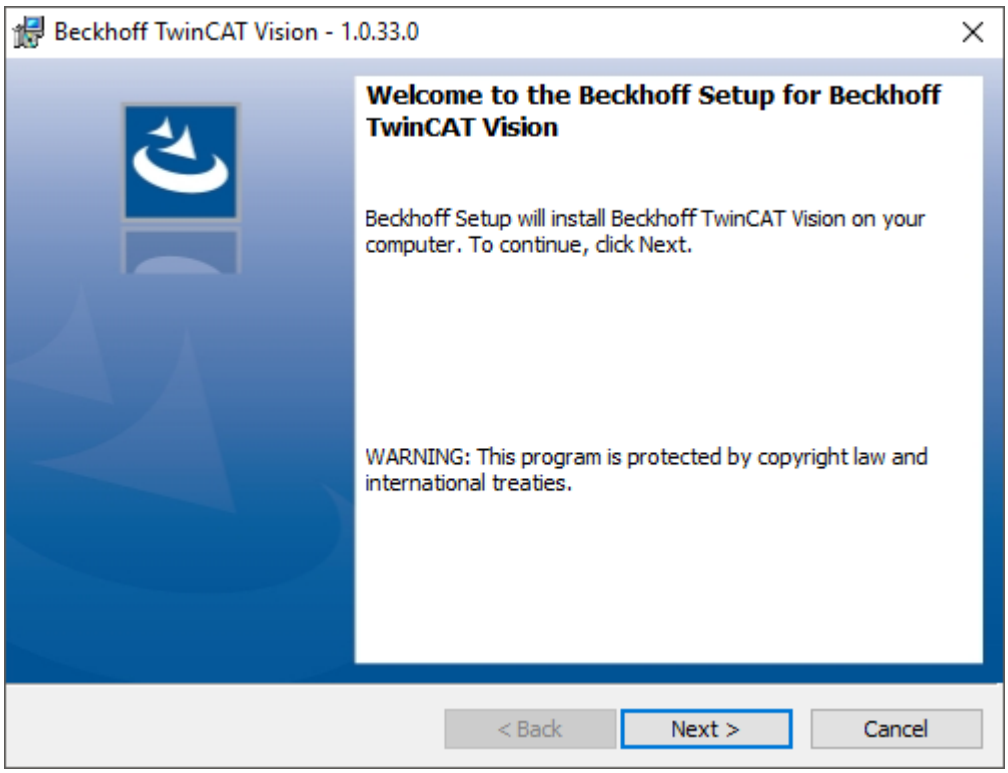
The installation of TwinCAT Vision on a Windows operating system consists of the following steps:

1. Make sure that TwinCAT is in Config mode and that no Visual Studio instance or TcXaeShell is open.
2. Run the installation file TF7xxx-Vision.exe as administrator. To do this, right-click the file and select **Run As Admin**. Confirm by clicking **Yes**.

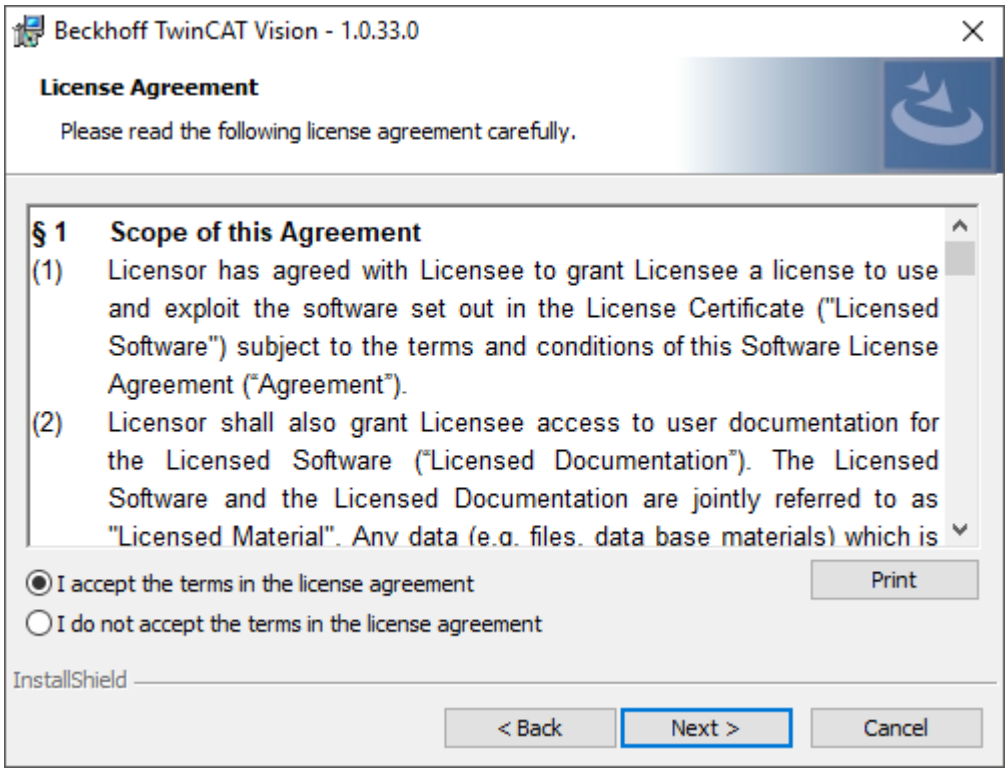
3. Select the installation language. Then click **OK**.



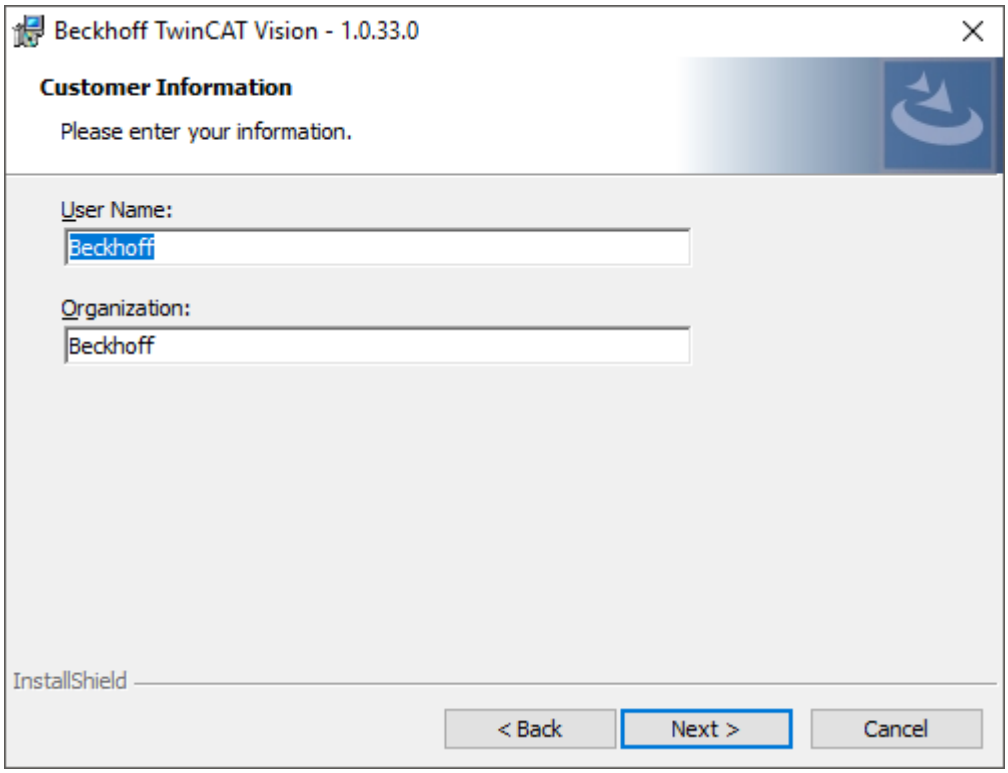
4. Wait a moment while the installation is being prepared, then click **Next**.



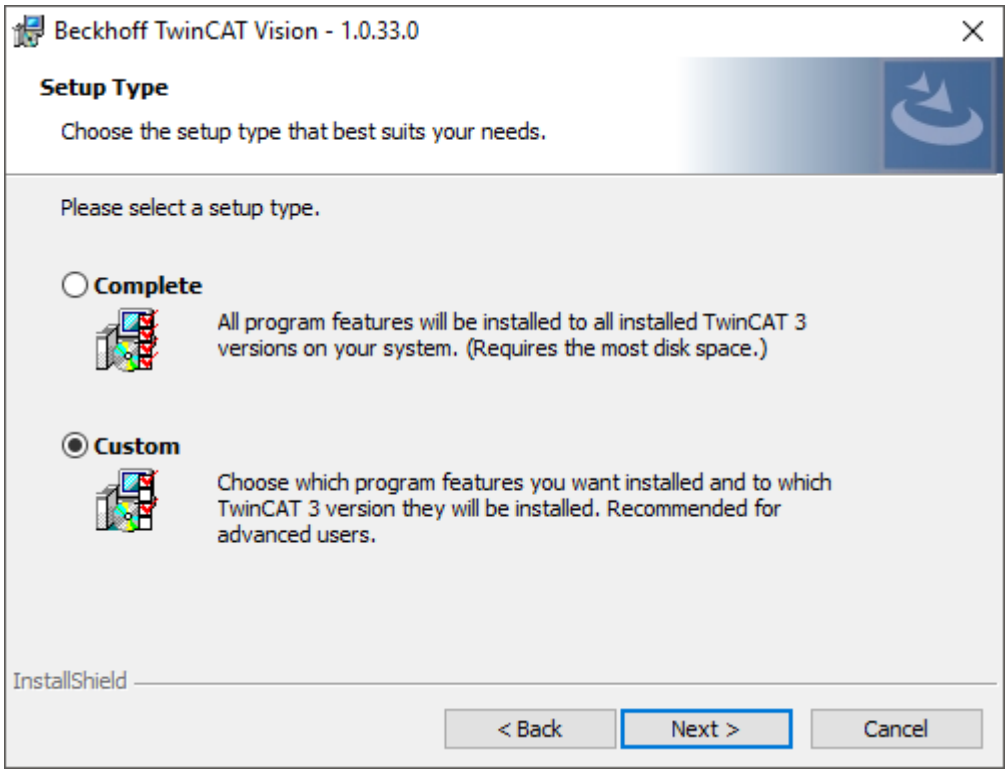
5. Make sure that you agree to the license agreement by checking the option **I accept the terms of the license agreement**. You can **print** the license agreement, if required. Then click **Next**.



6. Enter your user name and company as user information. Then click **Next**.

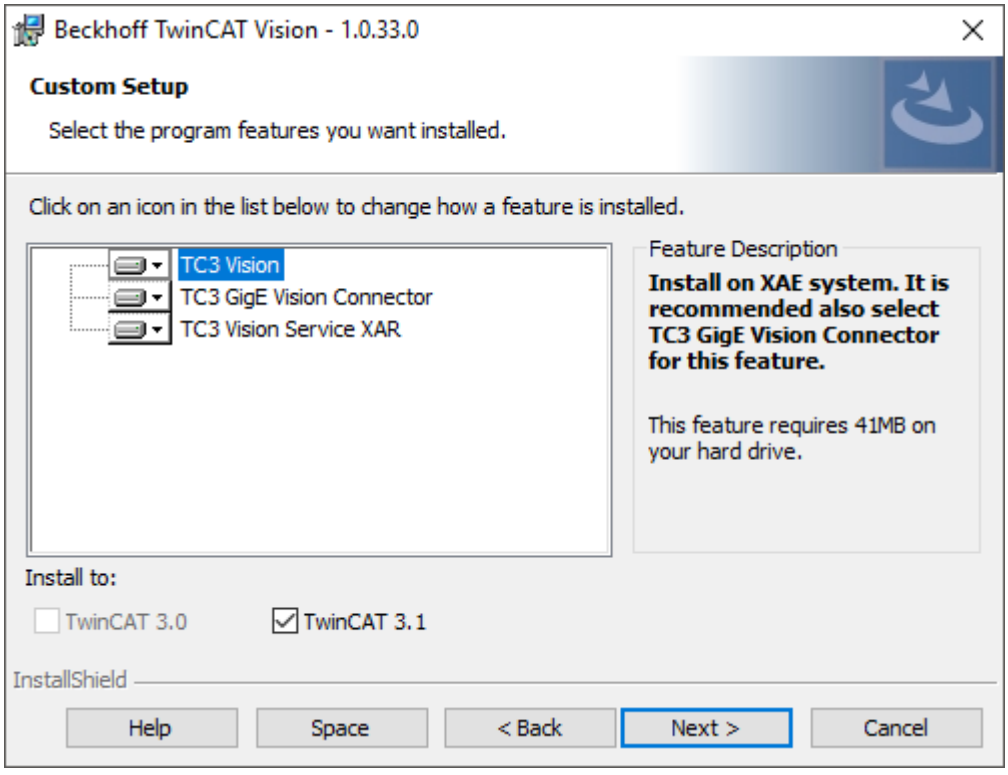


7. Choose between the two setup types **Complete** or **Custom**. Then click **Next**.



- A Complete setup installs all TwinCAT Vision components and requires the most memory space. This option is recommended for XAE systems. Since only the Vision Service can be installed on pure runtime systems and the other options are not available for selection, this option is also recommended here.
- A custom setup lets you choose in the next step which TwinCAT Vision components you want to install.

8. For each of the following TwinCAT Vision components, select whether you want to install them or not. In addition, select at least one TwinCAT 3 version in which the selected components are to be installed. Then click **Next**.

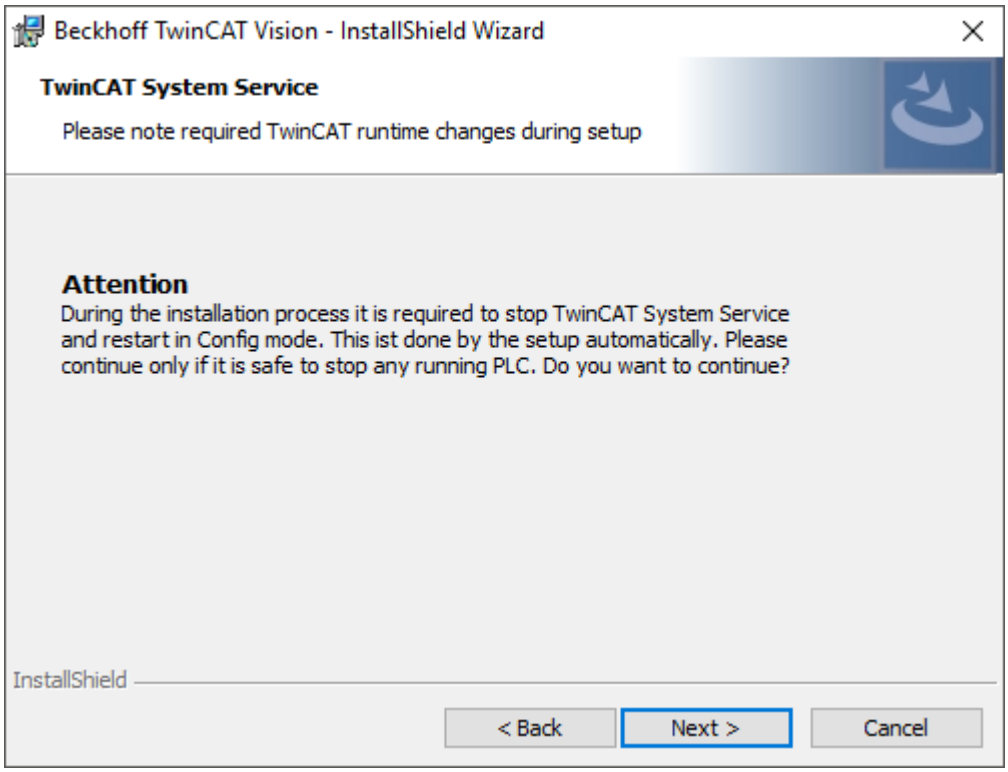


- **TC3 Vision**: Required for developing TwinCAT Vision applications and should be installed on XAE systems.
- **TC3 GigE Vision Connector**: Required for developing with GigE Vision devices and should be installed on XAE systems.
- **TC3 Vision Service XAR**: Contains the TwinCAT Vision Service and must be installed on target systems of a TwinCAT Vision application.

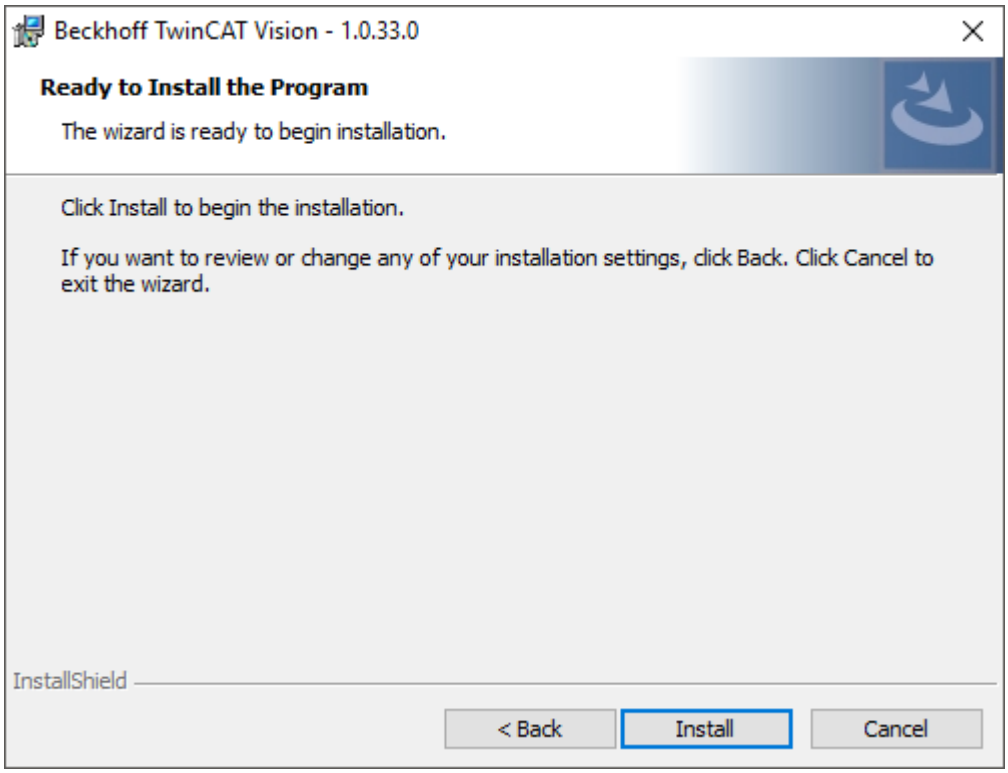
Explanation

The Vision Service must be installed on the target system. The components for TC3 Vision and TC3 GigE Vision Connector, conversely, are always copied automatically from the development system to the target system when activating the TwinCAT project. Therefore, they do not necessarily have to be installed on the target system.

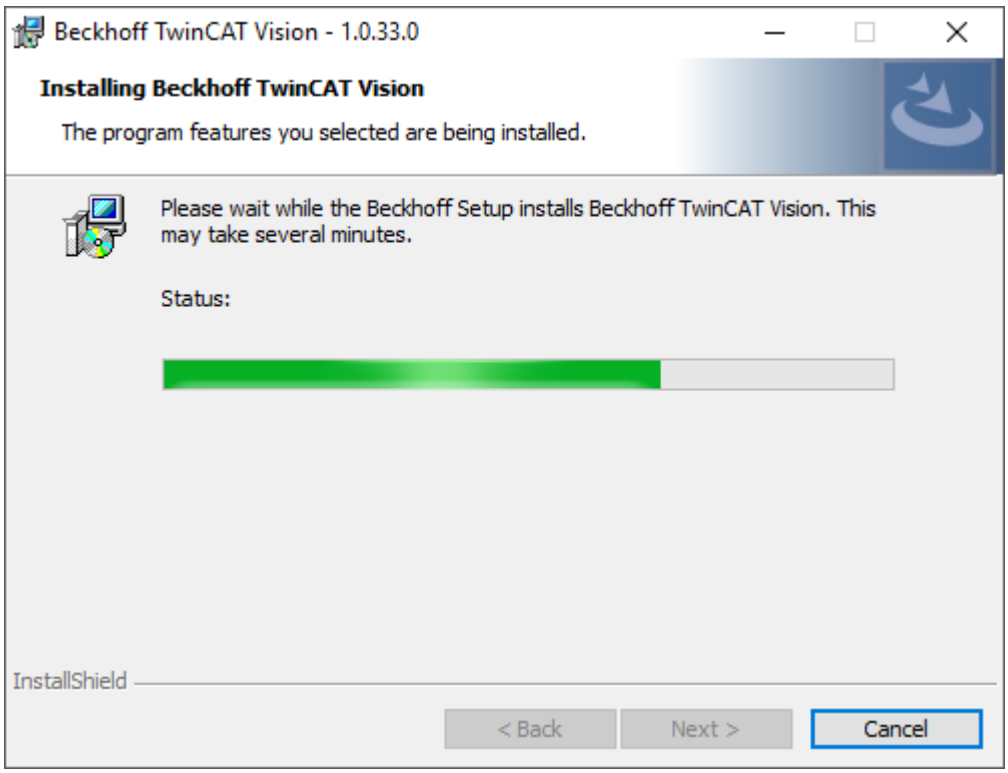
9. Please note that during the following installation the TwinCAT system service is stopped and restarted in Config mode. Continue with the installation only if a running PLC system can be stopped. In this case, click **Next**.



10. Make sure that you agree with the selected installation settings. Then click **Next**.

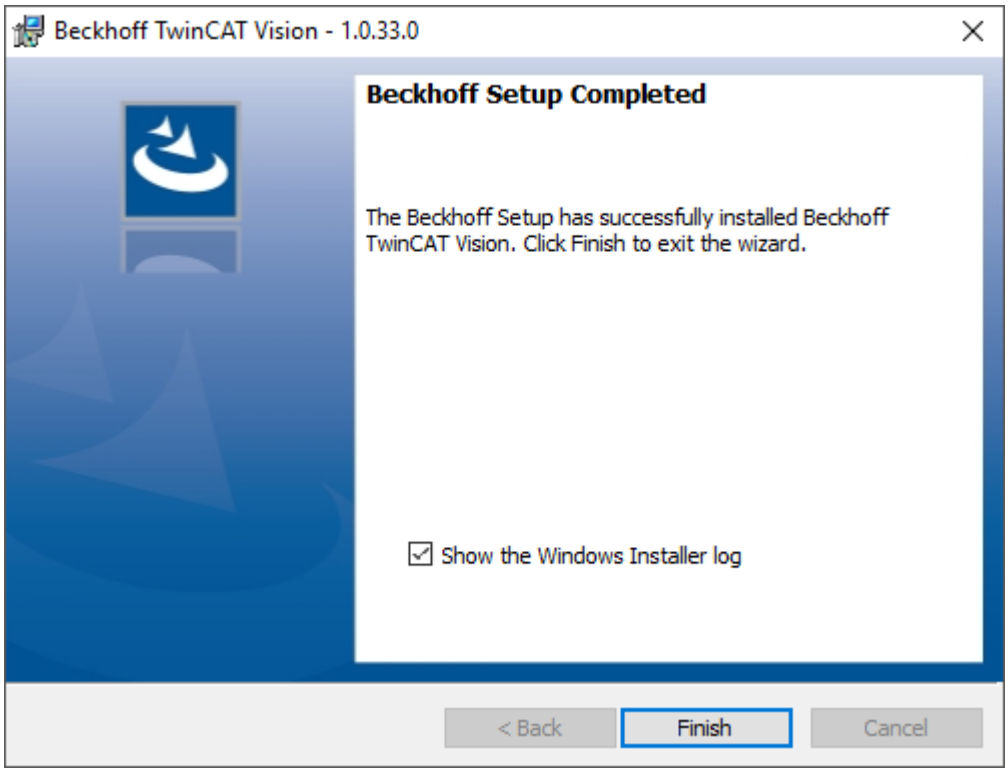


11. Wait while the selected TwinCAT Vision components are installed.



12. A dialog may appear showing a list of applications that have to be closed for a complete installation. Choose whether the applications should be automatically closed or whether you would like to restart the PC after the installation instead.

⇨ The installation is now complete. Optionally, you can choose to display the log file of the installation. Click **Finish**.



TwinCAT Vision must be licensed before it can be used after a successful installation. The procedure is described in the chapter on Licensing [▶ 24].

Information about versions and updating existing projects can be found at Version overview [▶ 16].

If the installation fails, you have the following options:

- Solve the problem using troubleshooting [▶ 2861].
- Contact customer support and provide the installation log. The installation log *TF7xxx-Vision.exe.log* is located in the following folder in the path: *C:\Users\<username>*

3.4 Installation instructions from TwinCAT 3.1 Build 4026

TwinCAT Package Manager

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, you can install TwinCAT Vision via the TwinCAT Package Manager, see Installing TwinCAT 3.1 Build 4026.

Normally you install TwinCAT Vision via the corresponding Workload; however, you can also install the packages contained in the Workload individually. This documentation briefly describes the installation process via the workload.

Command line program TcPkg

You can use the TcPkg **C**ommand **L**ine **I**nterface (CLI) to display the available workloads on the system:

```
tcpkg list -t workload
```

To display only the TwinCAT Vision Workloads, you can use this command:

```
tcpkg list TF7
```

You can install the Workloads of the TF7000-TF7810 Vision Functions using the following commands.

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

TwinCAT Package Manager UI

You can use the **U**ser **I**nterface (UI) to display all available workloads and install them if required.
To do this, follow the corresponding instructions in the interface.

3.5 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

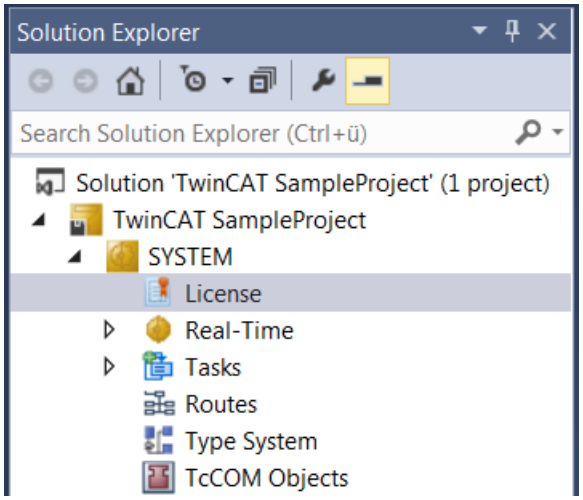
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function



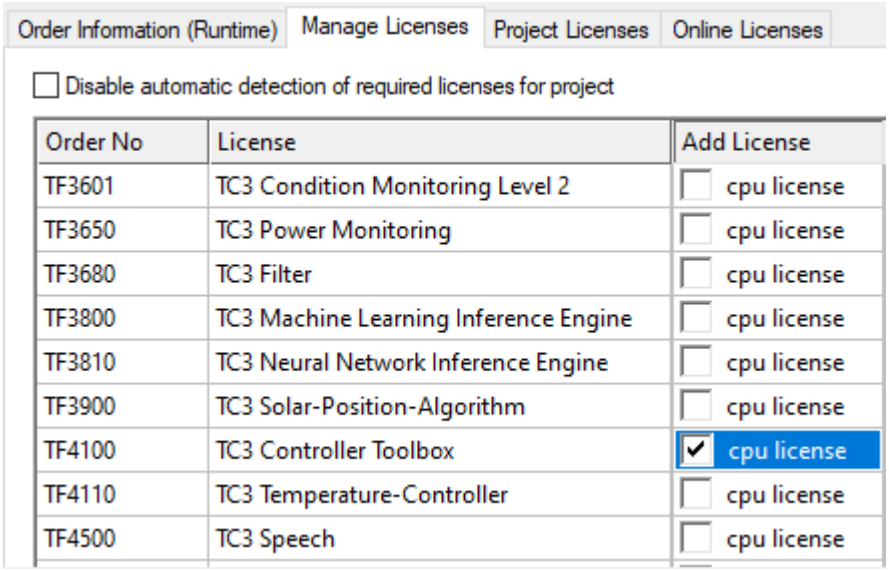
A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

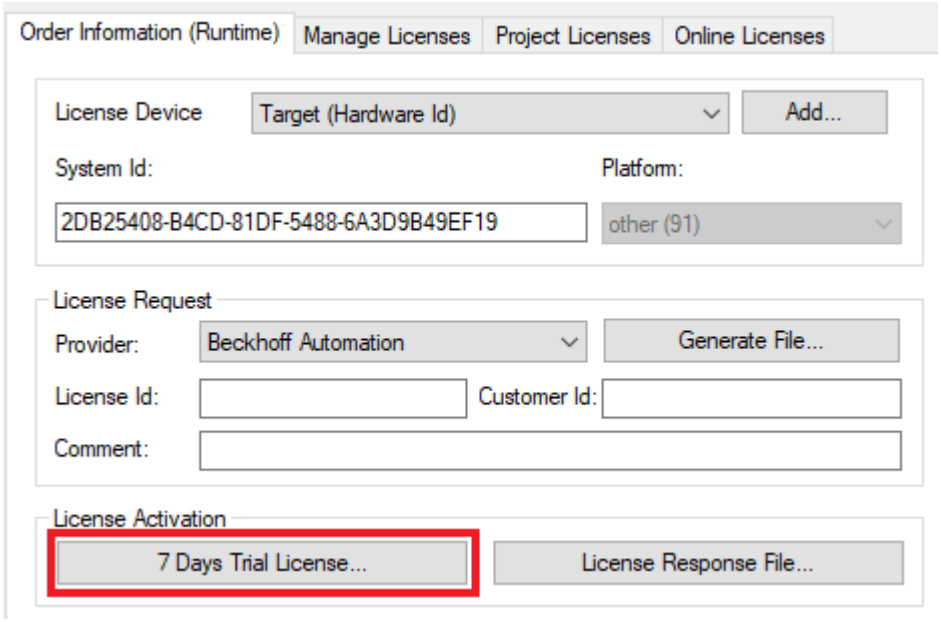


⇨ The TwinCAT 3 license manager opens.

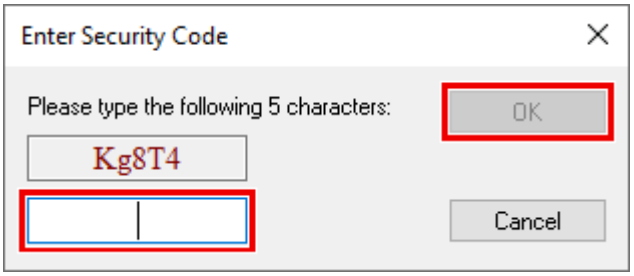
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**
7. Click **7-Day Trial License...** to activate the 7-day trial license.



   ⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.

10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 First steps

The following guide provides an introduction to TwinCAT Vision (after successful installation [▶ 15]).

1. Build Project [▶ 27]
   - Creating a TwinCAT project [▶ 27]
   - Choose Target System [▶ 29]
   - System configuration [▶ 31]
2. Displaying a Setting up a Vision device [▶ 33] and creating a Vision device
   - Displaying a Vision node and creating an application [▶ 33]
   - Creating a File Source Control [▶ 35] or
   - Creating a GigE Vision camera [▶ 38]
3. Creating a PLC project [▶ 45]
   - Creating a PLC project [▶ 45]
   - Integrating a PLC library [▶ 46]
   - Writing a PLC program [▶ 48]
   - Initializing a function block [▶ 49]
4. Run [▶ 50]
   - Activate TwinCAT project and start PLC project [▶ 50]
   - Image display in ADS Image Watch [▶ 51]
   - Download sample project: https://github.com/Beckhoff/TF7xxx_Samples

More detailed information can be found at Development environment [▶ 53], API reference [▶ 137] and Samples [▶ 2753].

4.1 Build Project

Video example for setting up a system for TwinCAT Vision:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/5906036107.mp4

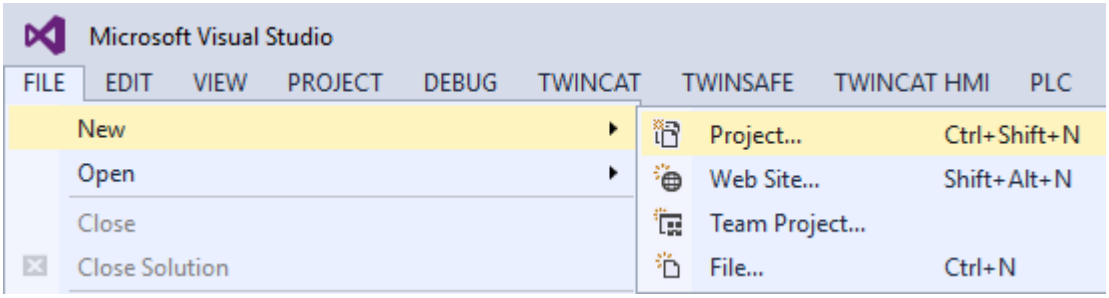
4.1.1 Creating a TwinCAT project

TwinCAT Vision is fully integrated in the TwinCAT system. A standard TwinCAT project is therefore the starting point for every TwinCAT Vision application:

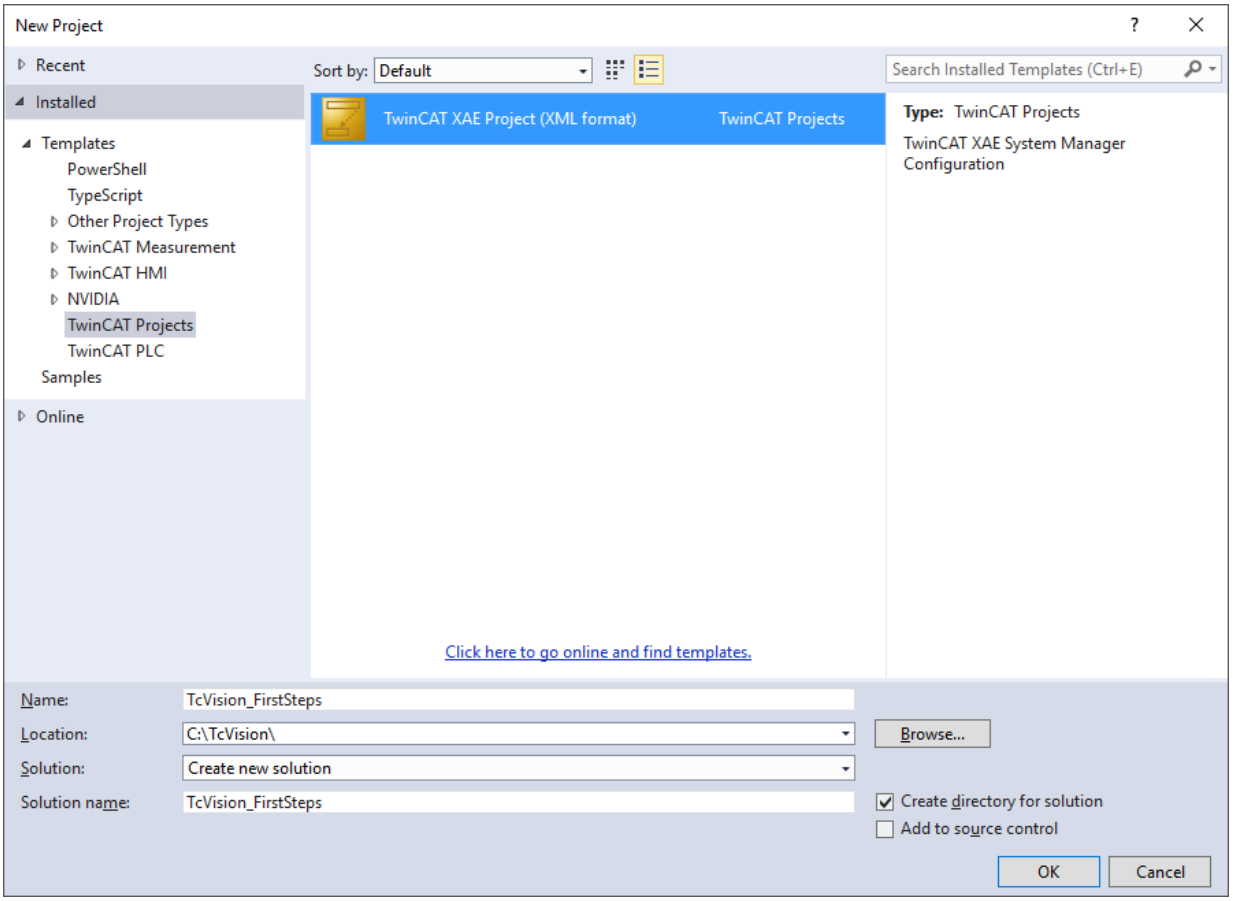
1. Open Visual Studio with integrated TwinCAT.



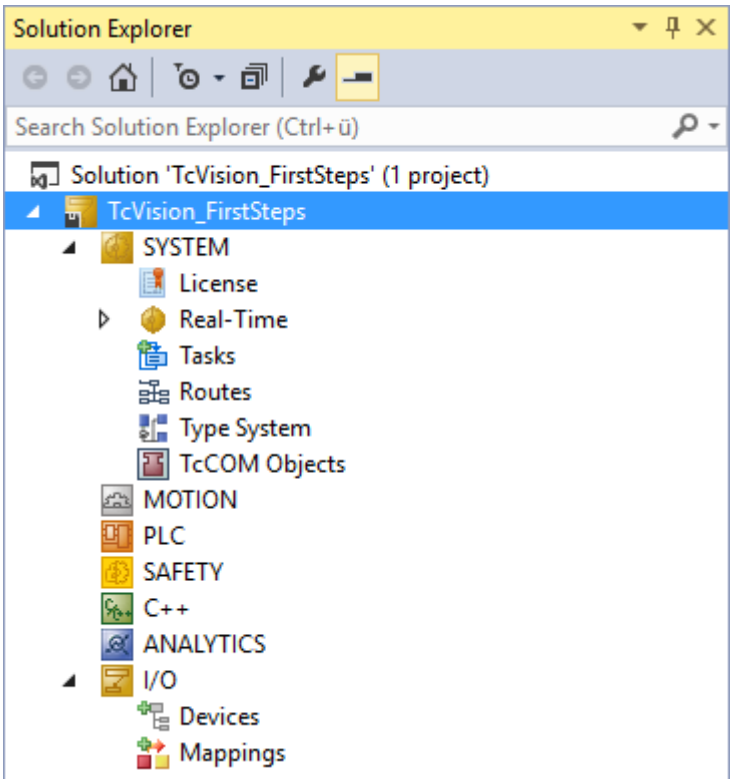
2. Open the **New/Project** dialog.



3. Select name and location.



⇨ A TwinCAT project is created.



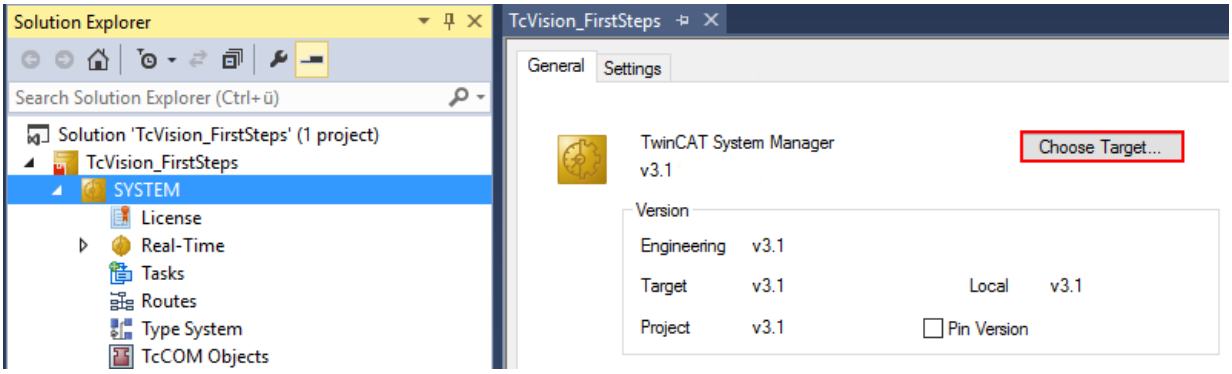
Next step

Choose Target System [▶ 29]

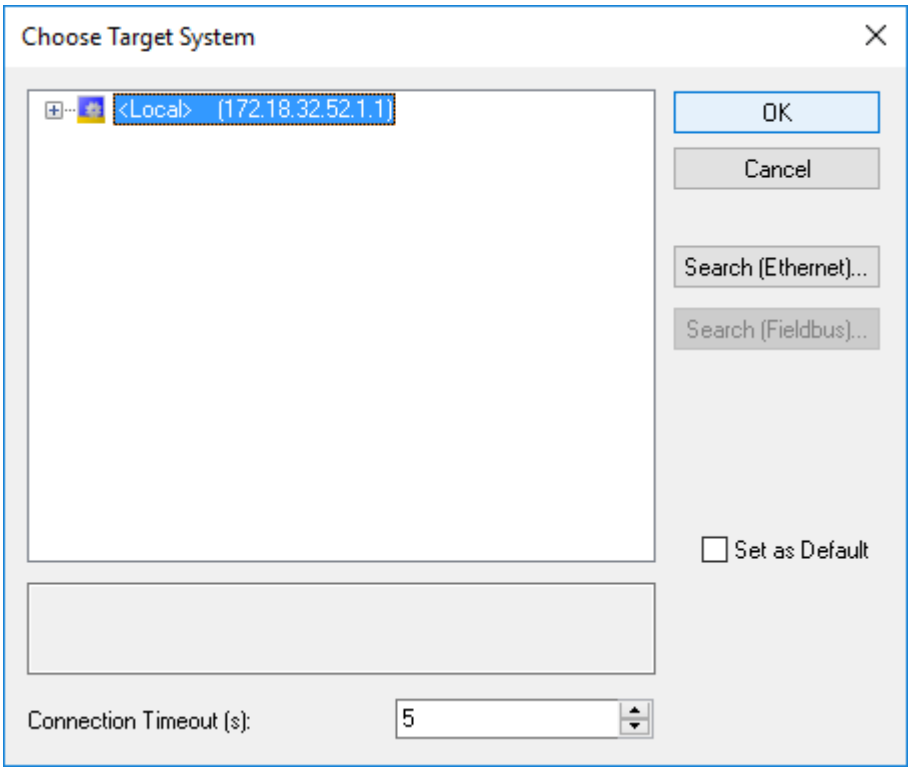
4.1.2 Choose Target System

TwinCAT distinguishes between development and runtime systems. If your development system is not your runtime system, create a route to the runtime system and select it. This can be done by following the steps below, for example:

1. Click **Solution Explorer > SYSTEM > General > Choose Target...**
   The **Choose Target System** dialog opens, listing all target systems to which routes exist from the development system.

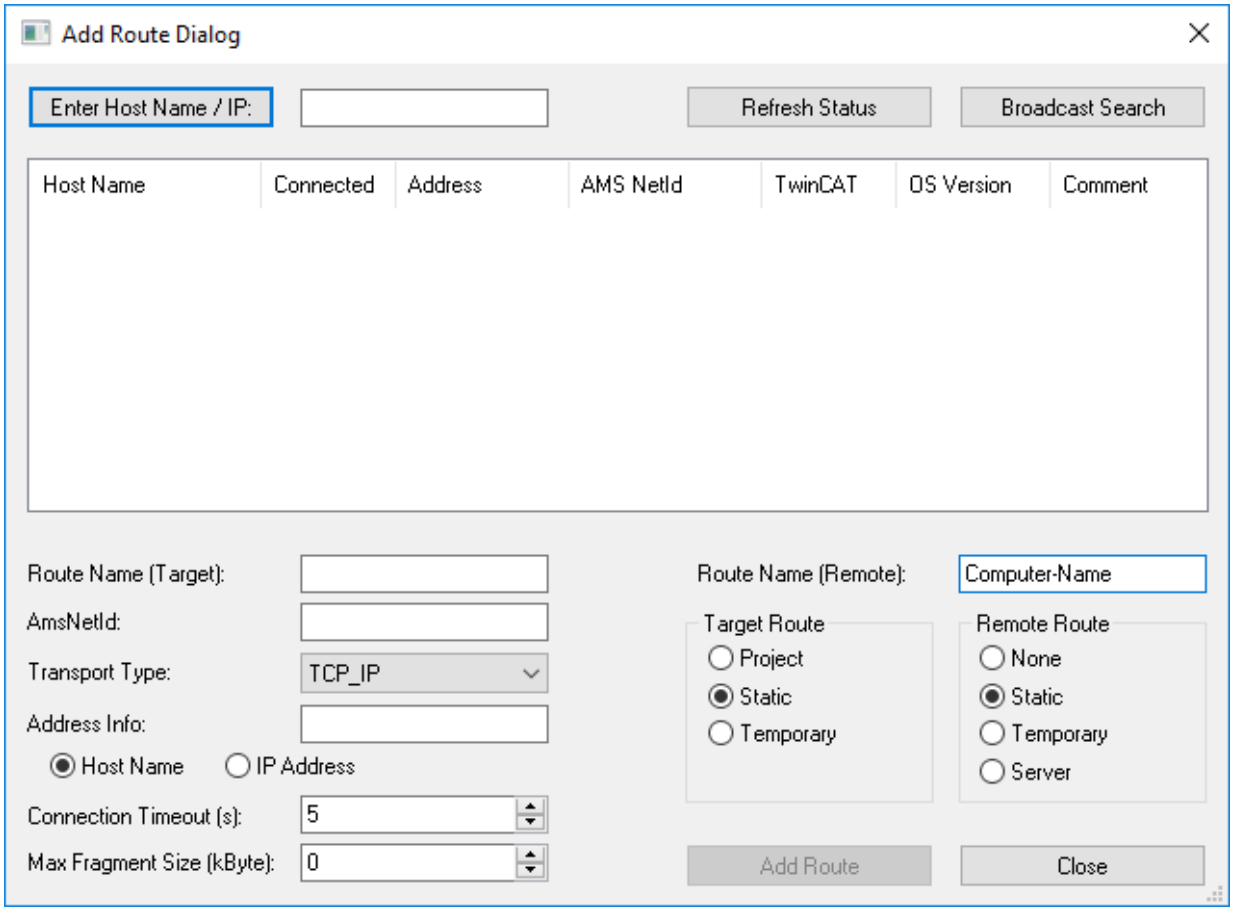


2. If the target system is not yet listed, that is there is no route to the system, click **Search**.

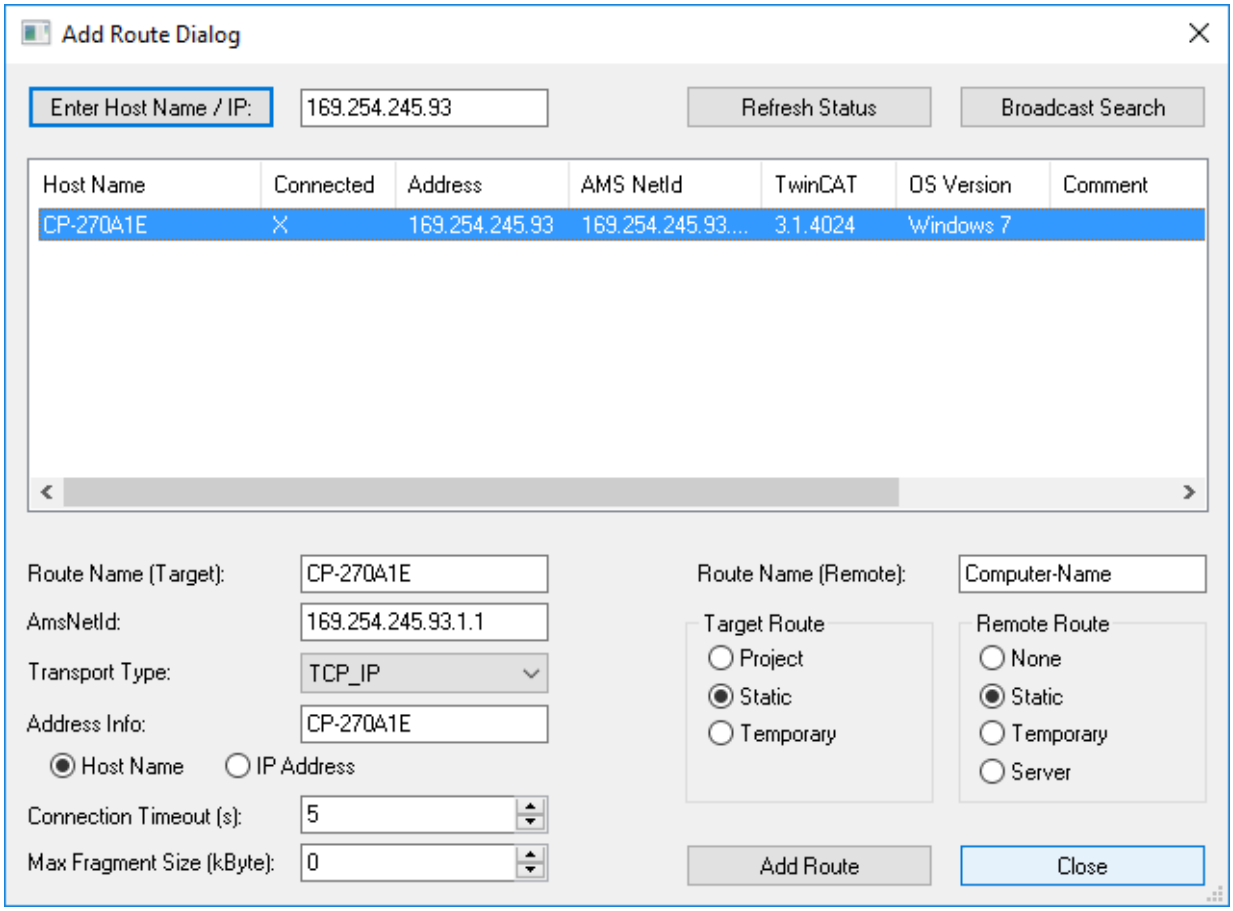


The Add Route Dialog opens, where you can add more routes.

3. In the **Add Route Dialog** click **Broadcast Search** to search for TwinCAT systems in the network. Alternatively, search for the device name or IP address using **Enter Host Name / IP**. Any systems that are found are then displayed.



4. Select a system from the list and click **Add Route** to create a route to it. The displayed information and setting options are described in detail in the chapter Add Route Dialog. The new route now appears in the **Choose Target System** dialog.



5. Select the target system in the **Choose Target System** dialog and confirm with **OK**.

Next step

System configuration [▶ 31]

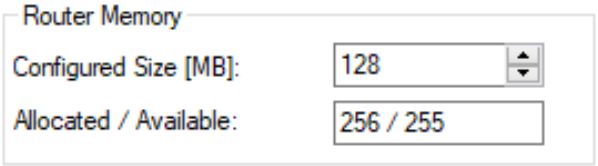
4.1.3 System configuration

The system configuration depends on the target system and is described in detail in chapter System configuration [▶ 54].

Increase router memory

TwinCAT Vision requires dynamic memory allocation when creating an image, for example. This takes place in the router memory [▶ 54]. Accordingly, the router memory must be configured with adequate size.

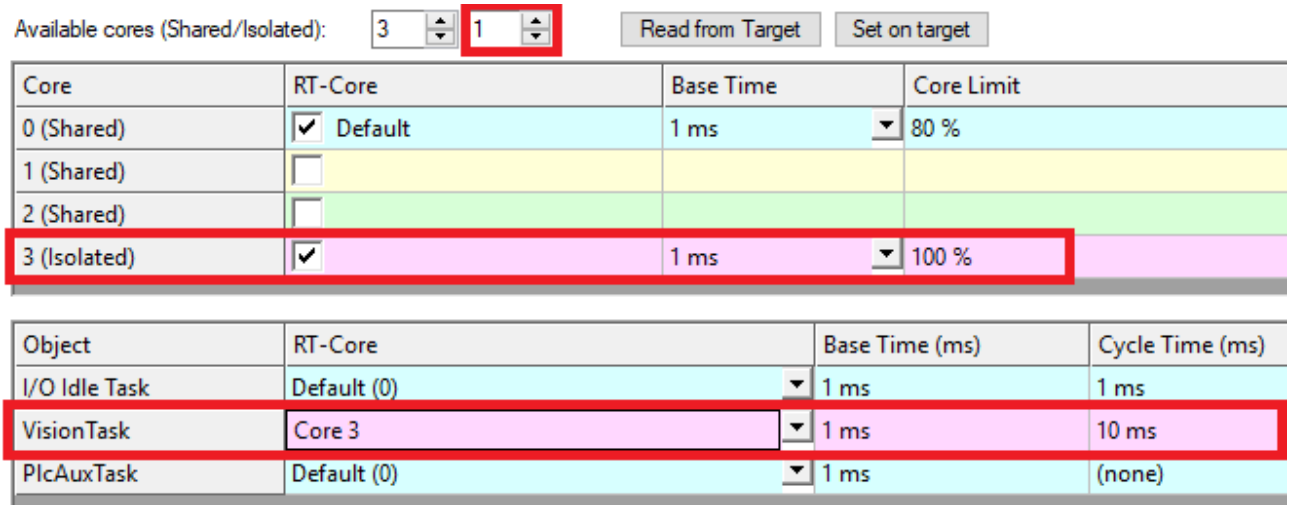
This is done in the TwinCAT project under **SYSTEM > Real-Time > Settings > Router Memory (Mbyte)**.



The configuration must then be activated . If the router memory is reduced, it is necessary to restart the target system.

Selection of a CPU core and a task

In multi-core systems, TwinCAT enables a distinction between Windows cores and isolated cores. With Windows cores, the operating system and the TwinCAT application share the processor time. The real-time portion can be limited to between 10% and 90%. In contrast, isolated cores are fully available to the TwinCAT application. For this reason, the use of isolated cores is recommended for Vision applications. Divided cores can also be used without problems for the initial testing.



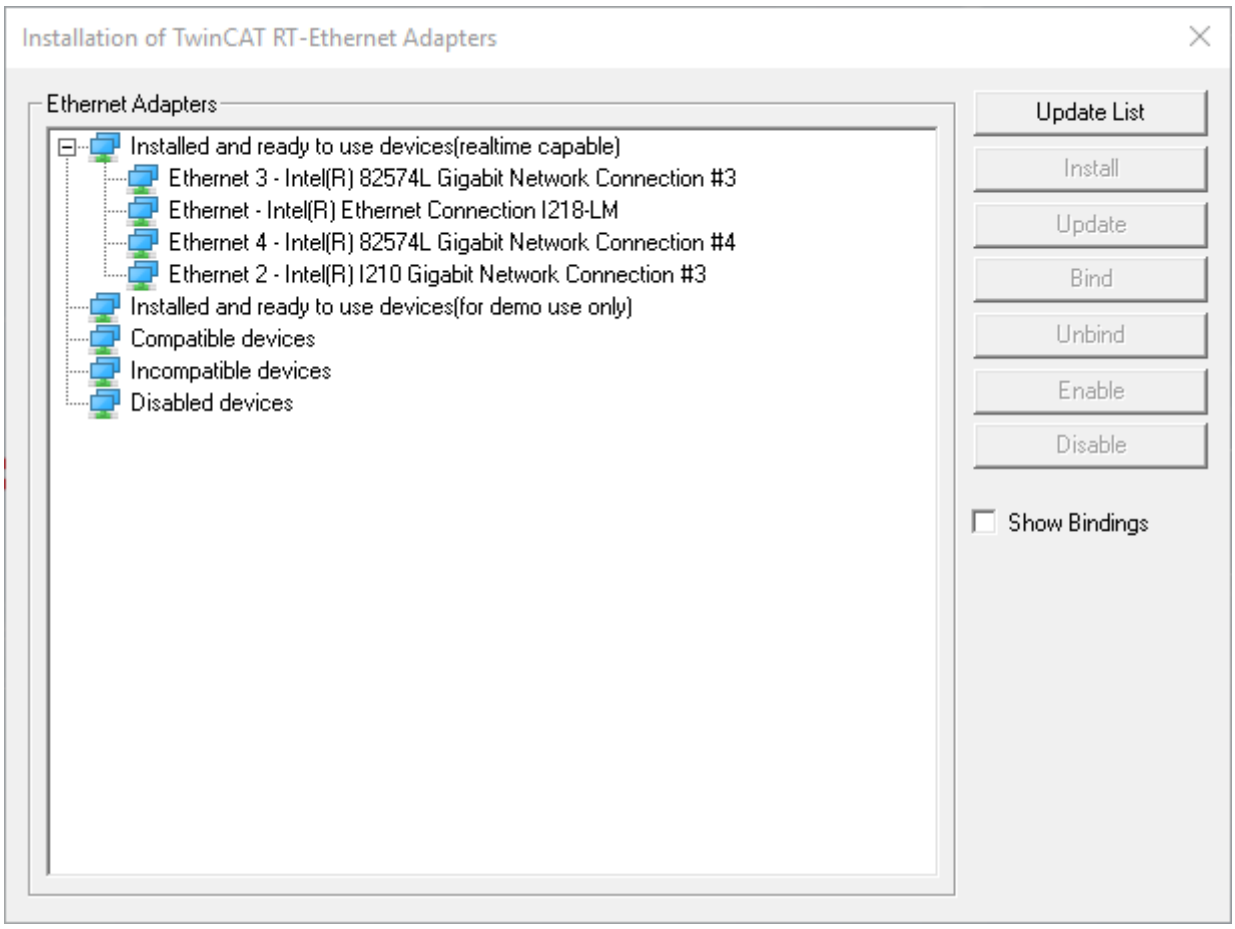
The core configuration is carried out in the TwinCAT project tree under **SYSTEM > Real-Time > Settings**. Detailed configuration instructions can be found in the chapter CPU cores and tasks [▶ 61]. This chapter also describes how to change the cycle time of the executing task.

The configuration must then be activated  . In addition, the target system must be restarted when redefining divided and isolated cores.

Installing a real-time Ethernet driver

To connect GigE Vision cameras to TwinCAT Vision, the TwinCAT RT Ethernet driver must be installed on the network adapters used. The driver ensures that TwinCAT can use the network adapter during runtime. Details can be found in chapter Network adapter [▶ 56].

In the development environment, an installation assistant can be called up via **TWINCAT > Show Realtime Ethernet Compatible Devices**... On the target system the assistant can be started via the file *C:\TwinCAT\3.1\System\TcRteInstall.exe*. The RT Ethernet driver can be installed for the network adapters listed under **Compatible devices** in the dialog that opens.



Make sure that all network adapters you want to connect a camera are now in the list of **Installed and ready to use devices (realtime capable)**.

Next step

Displaying a Vision node and creating an application [▶ 33]

4.2 Setting up a Vision device

Video example for working with File Source Control:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/5931182859.mp4

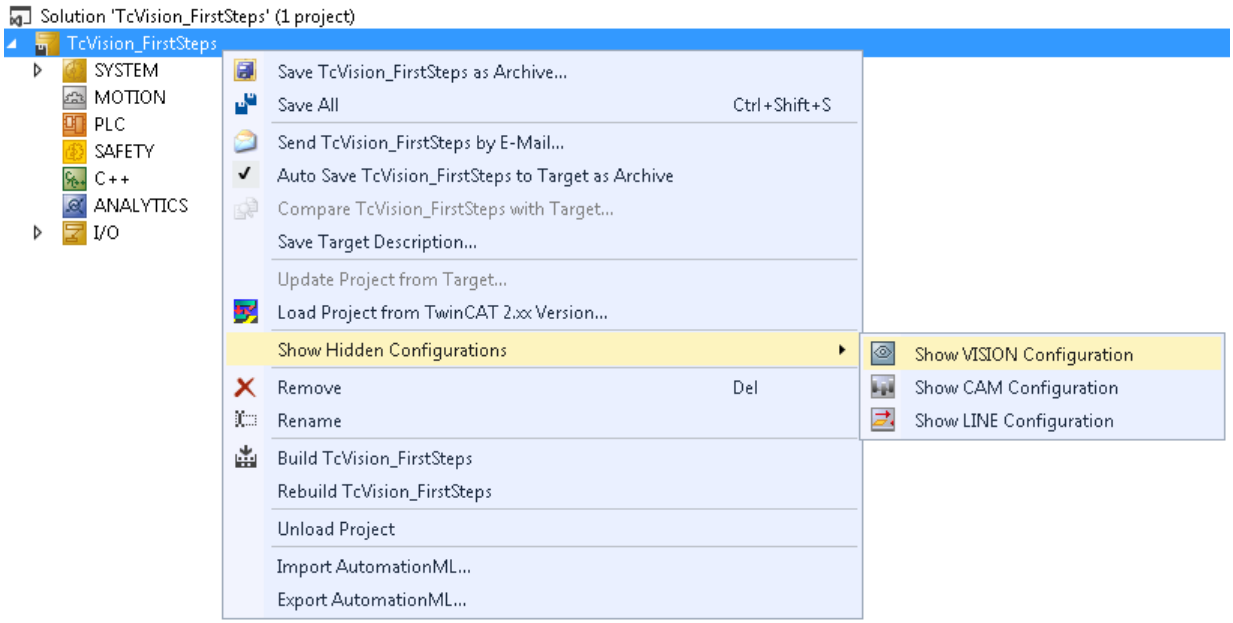
Video example for creating a camera instance:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/5931185163.mp4

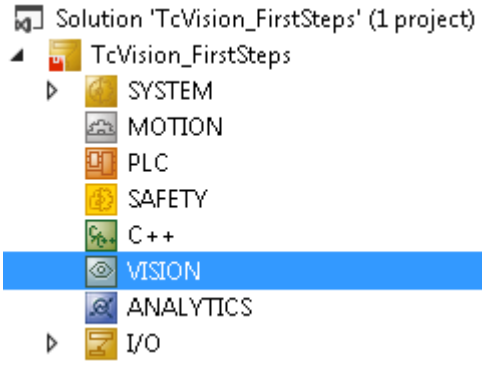
4.2.1 Displaying a Vision node and creating an application

There is a Vision node within the TwinCAT project for the Vision configuration.

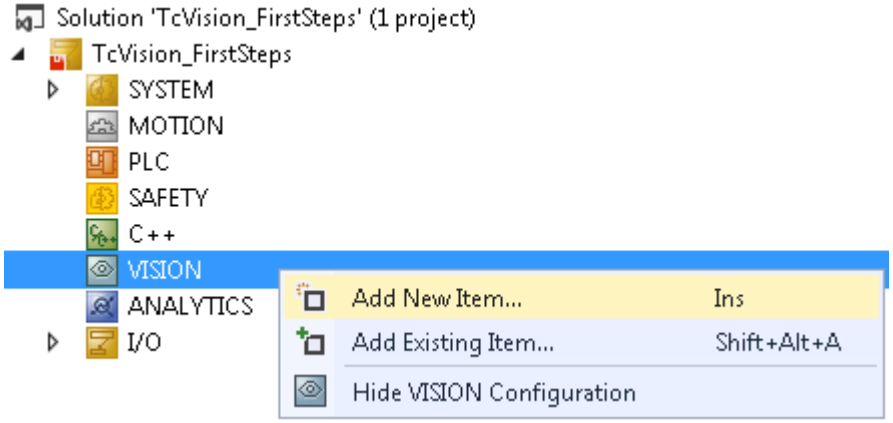
1. If the Vision node is not displayed in the TwinCAT project, right-click on the TwinCAT project in the Solution Explorer >**Show Hidden Configurations** > **Show VISION Configuration.**



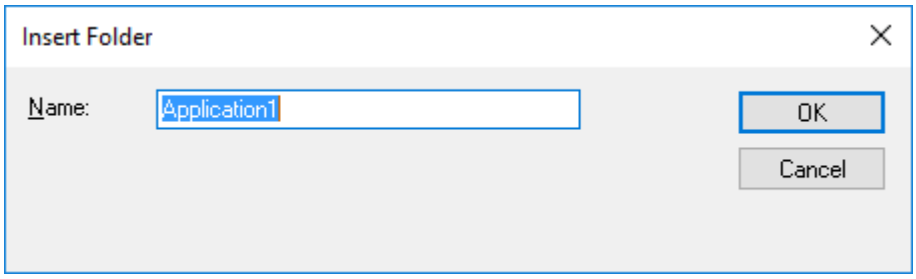
⇨ The Vision node is now visible.



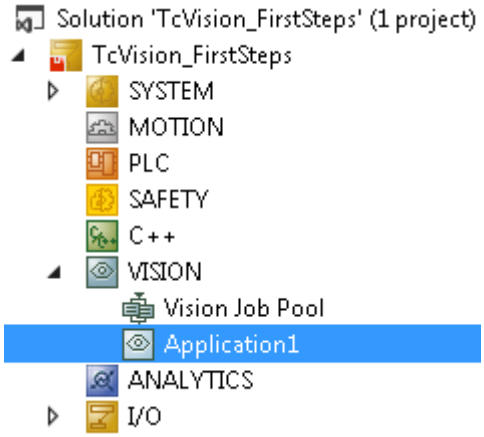
2. Within the Vision node, folders (also referred to as applications) are used to group the image processing devices.
Right-click on **Vision Node** > **Add New Item.**



3. Select a name and confirm with **OK.**



⇨ The folder is created



In addition, you can find further information here:

- VISION node [▶ 65]
- Application node [▶ 68]

Next step

In the next step, a Vision device is created which can be used to load images into the PLC. This can be either a real GigE Vision camera or a File Source Control for loading existing image files:

Creating a GigE Vision camera [▶ 38] or Creating a File Source Control [▶ 35]

4.2.2 Creating a File Source Control

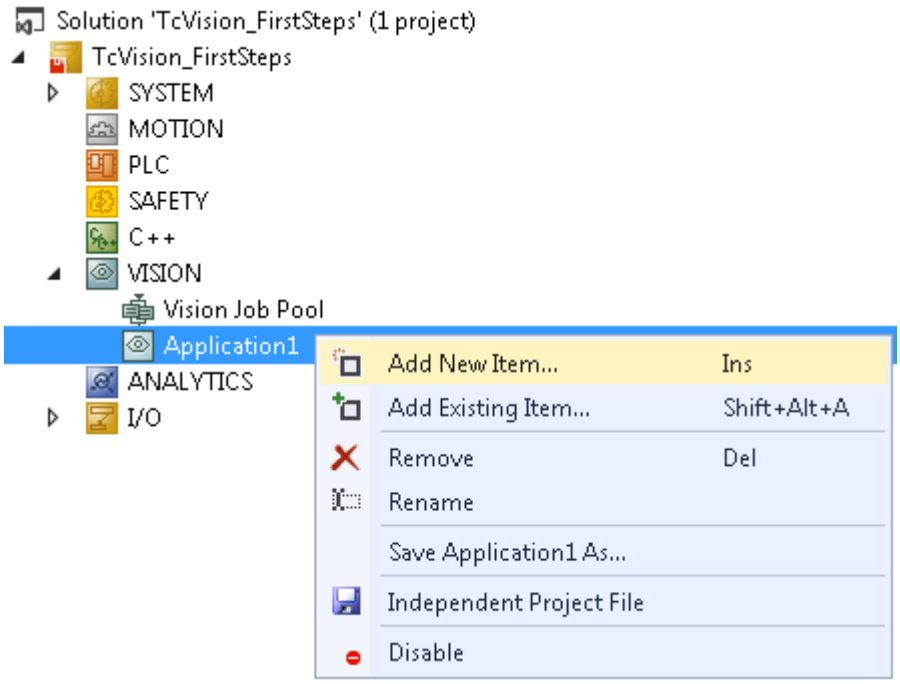
The creation of a File Source [▶ 125] object is described in the following steps. This can be used to load stored images into the PLC. To do this, the target system must be in Config Mode. Click on the  symbol in the TwinCAT XAE Base toolbar or on **Restart TwinCAT (Config Mode**) in the **Visual Studio menu under TWINCAT**.

Alternatively, you can Creating a GigE Vision camera [▶ 38].

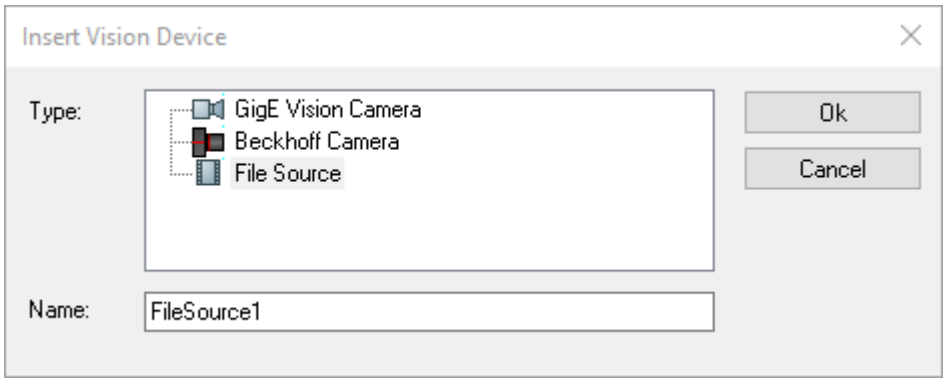
Proceed as follows to create the File Source object:

1. Click an application under the VISION node you created earlier.

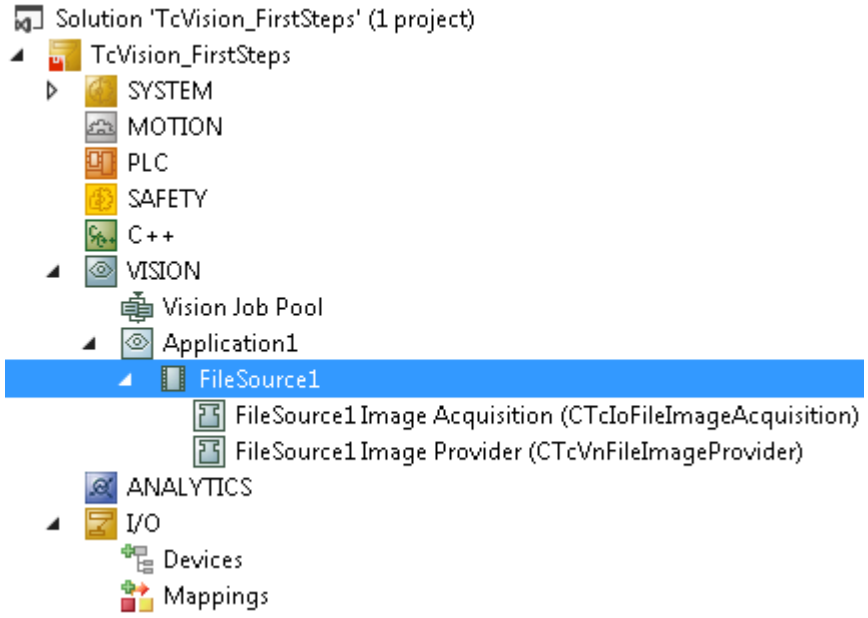
2. Right-click and select **Add new item...**



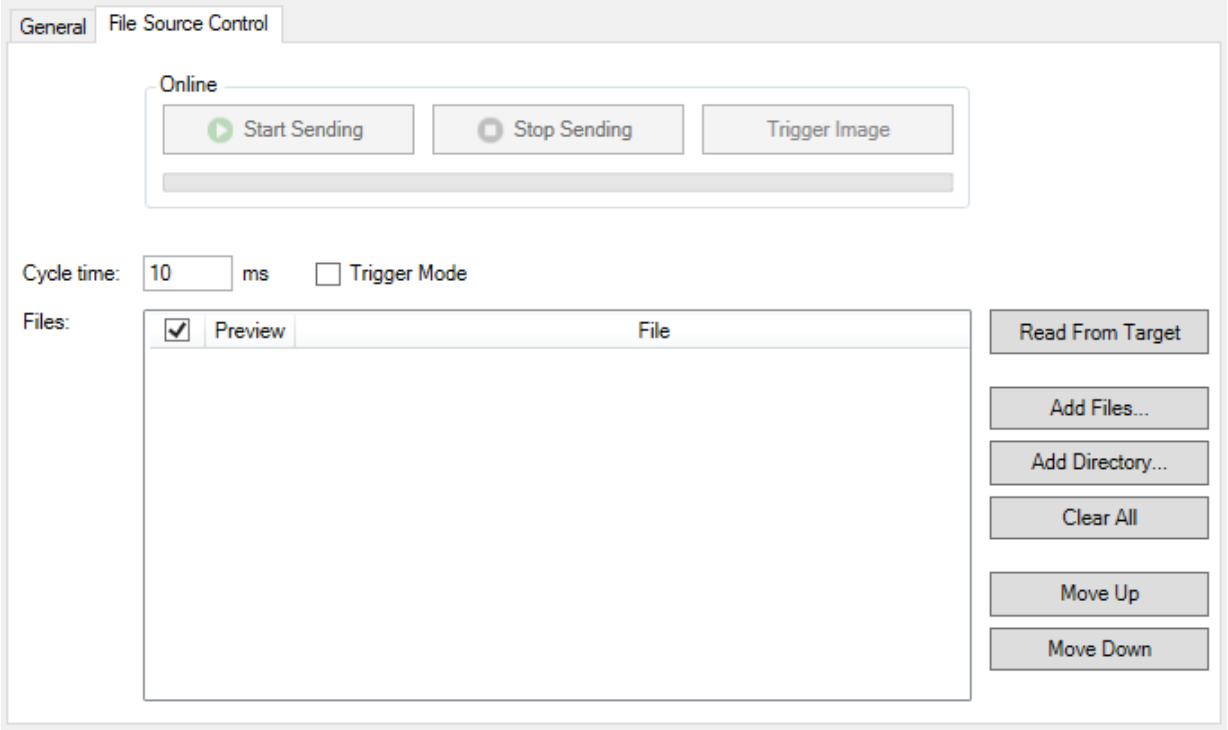
3. Select the device type **File Source**, adjust the name if necessary, click **OK**.



⇨ The File Source Control is now created below the application and is ready to use.

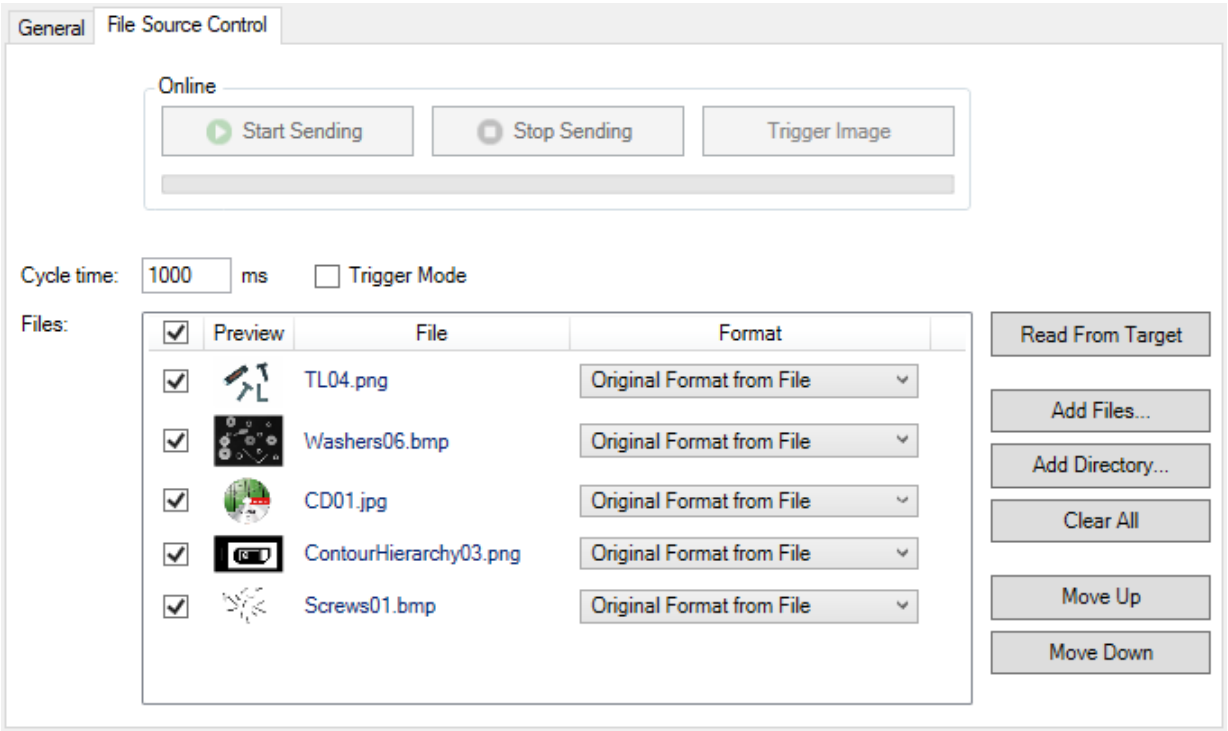


4. Double-click the **File Source Object** underneath the application node in the project tree. In the window that opens, select the **File Source Control** tab and implement the relevant settings.



5. Click **Read From Target**. The File Source Control [▶ 126] in the project is compared with any images stored on the selected target system.
6. Add images to the File Source Control via **Add Files...**, **Add Directory**... or by drag and drop. If necessary, you can download sample images with the project download: First steps [▶ 27]
7. If necessary, adjust the cycle time in which the File Source Object sends images to the PLC.

⇨ The File Source Object can be used like this:



Next step

Creating a PLC project [▶ 45]

4.2.3 Creating a GigE Vision camera

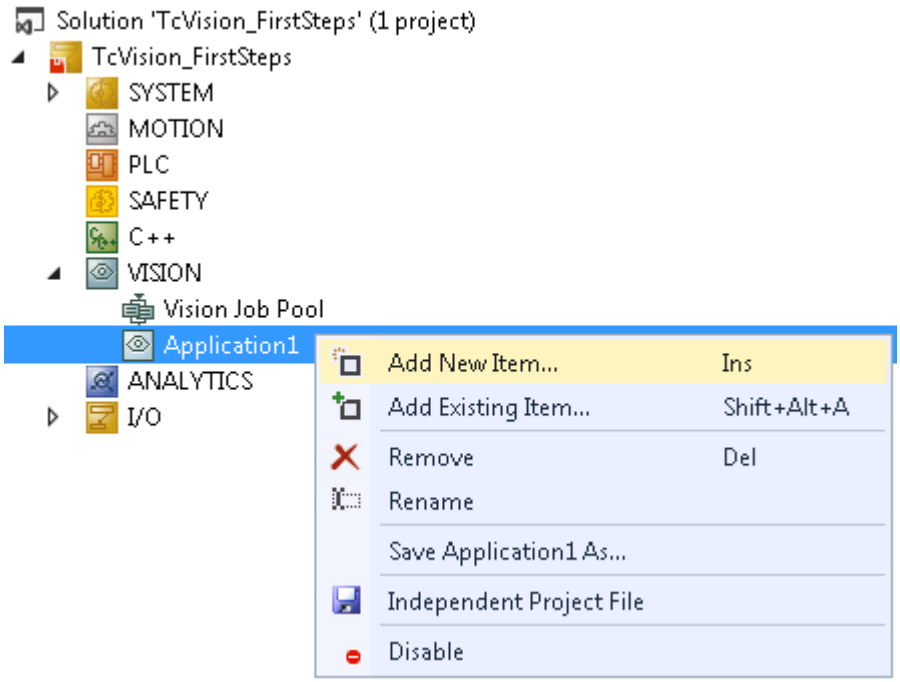
The following steps describe how to create a GigE Vision camera object [▶ 69], which can also be used to create a Beckhoff camera [▶ 125] object. This requires a GigE Vision camera to be connected to the PC and the TwinCAT RT Ethernet adapter to be installed for the corresponding network port, as described in System configuration [▶ 31].

In addition, the target system must be in Config Mode. Click on the  symbol in the TwinCAT XAE Base toolbar or on **Restart TwinCAT (Config Mode)** in the **Visual Studio menu under TWINCAT**.

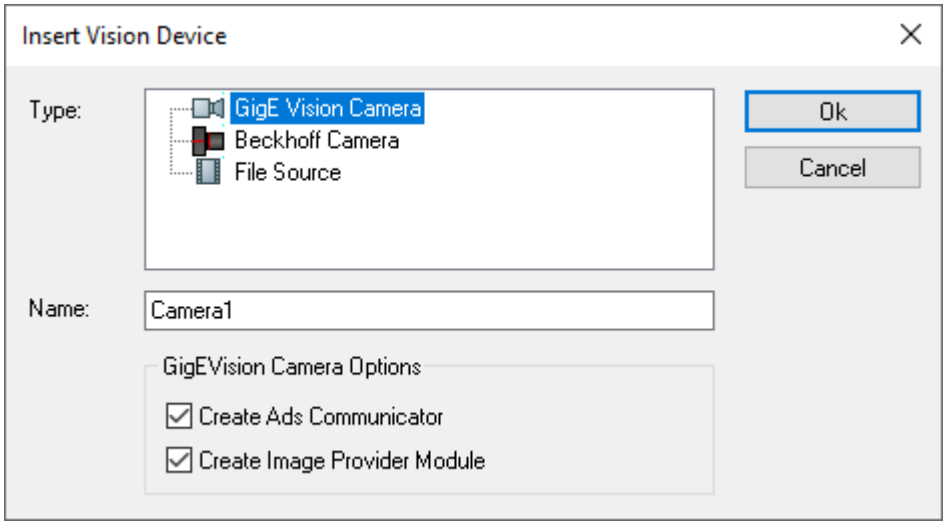
Alternatively, you can create a File Source Control [▶ 35].

To create a camera, proceed as follows:

1. Click an application under the Vision node you created earlier.
2. Right-click and select **Add new item...**



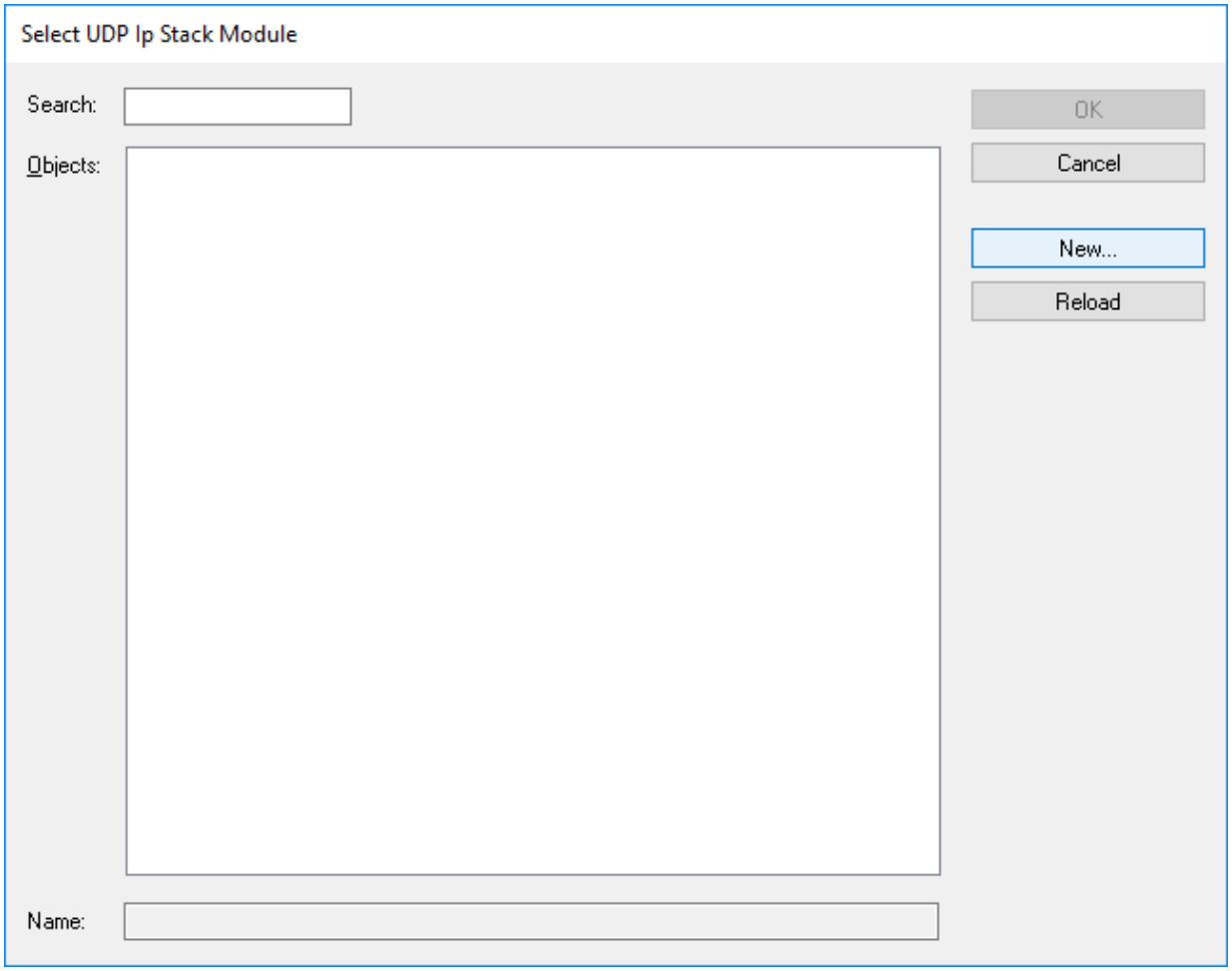
3. Select the device type **GigE Vision Camera or Beckhoff camera**. If required, adjust the camera name and the GigEVision Camera Options and click **Ok**.



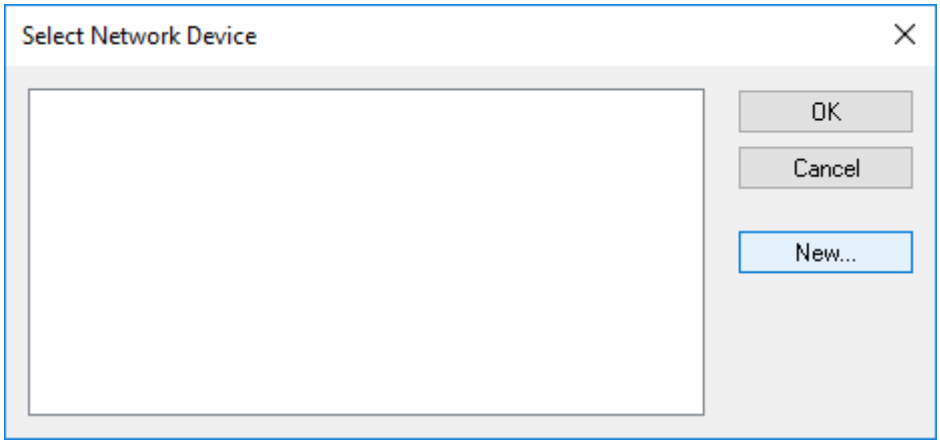
An Image Provider is the interface to the PLC function blocks and is therefore normally always required (see Camera details [▶ 123]).

An Ads Communicator is only required for recording streams via the Record/Playback [▶ 101] tab in the camera object.
If no streams are to be recorded, this option can be disabled. This eliminates the need for the Ads Communicator module and the corresponding VnTask ADS.

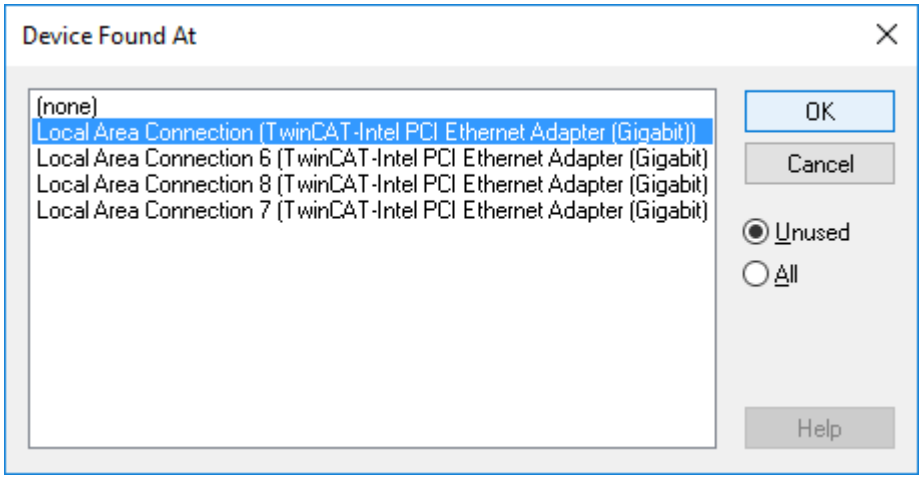
4. GigE Vision cameras communicate via UDP/IP. A suitable network device with a corresponding IP stack in the I/O node is required for the network port used. Create this by clicking **New...**.
   Only one network device with one IP stack may be created for each physical port of a network card.



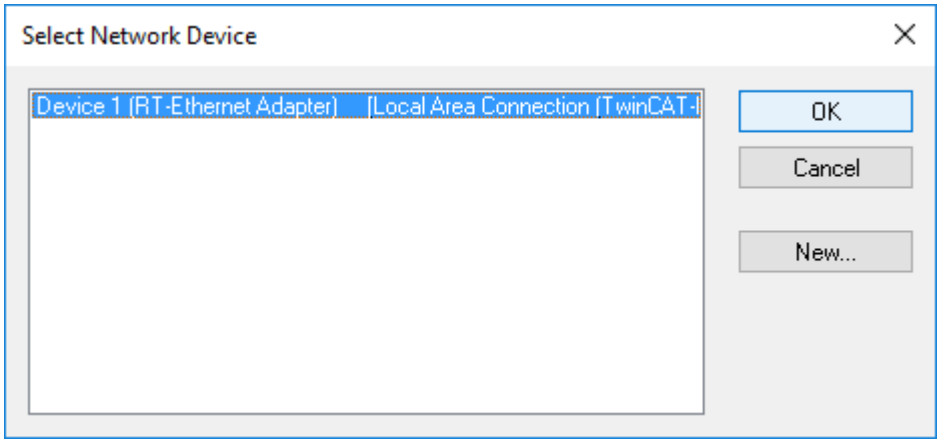
5. To create a new network device, click **NEW...** in the **Select Network Device dialog.**



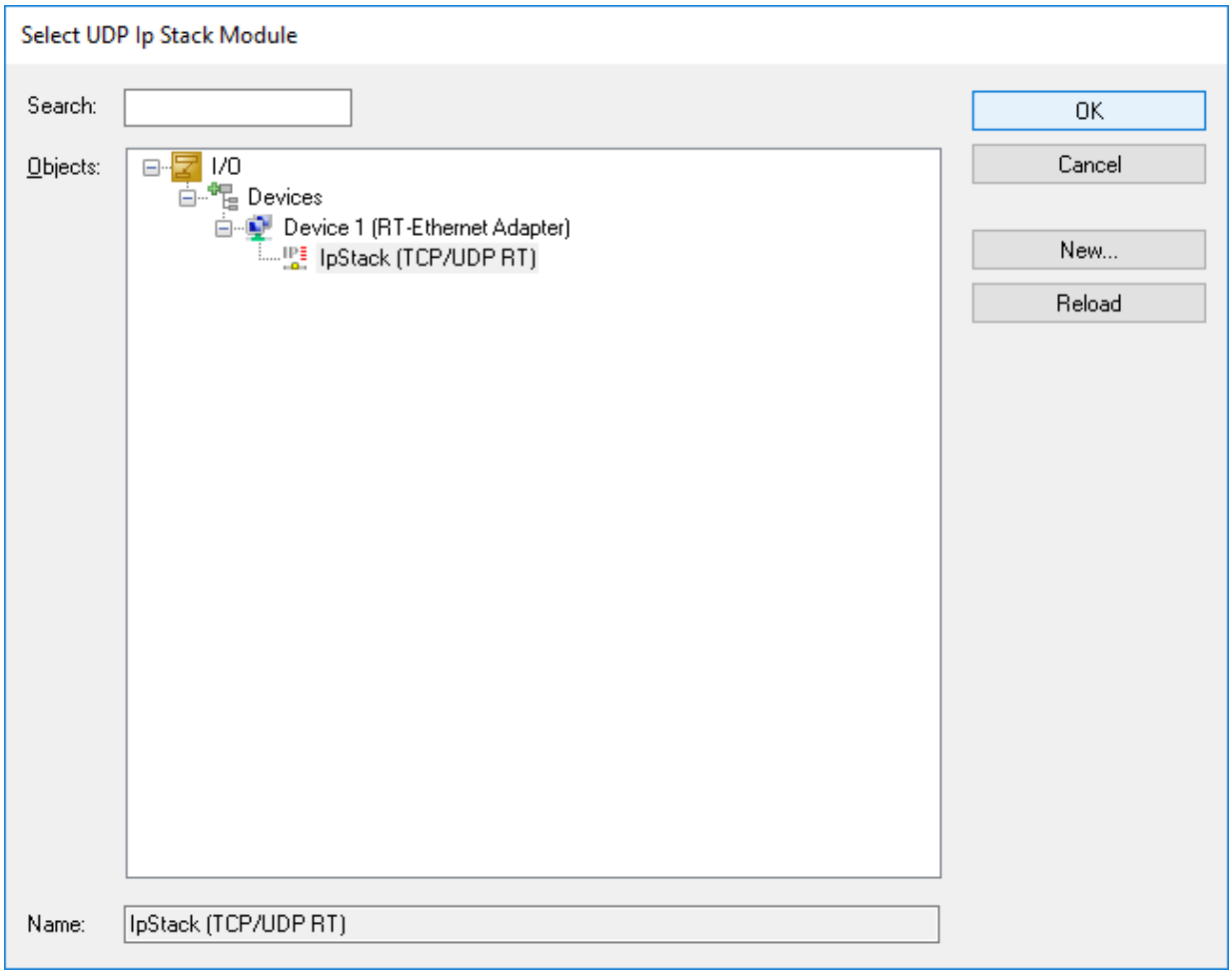
6. In the **Device Found At** dialog, select the network port to which the camera is connected and click **OK**. **Only network ports for which the TwinCAT RT Ethernet driver is installed are displayed.**



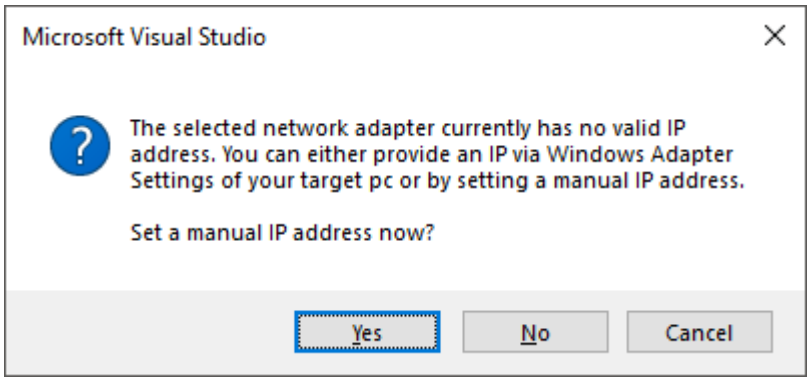
7. The network device is now created in the I/O configuration. Select it in the **Select Network Device** dialog and click OK. This creates an IP stack for the device.



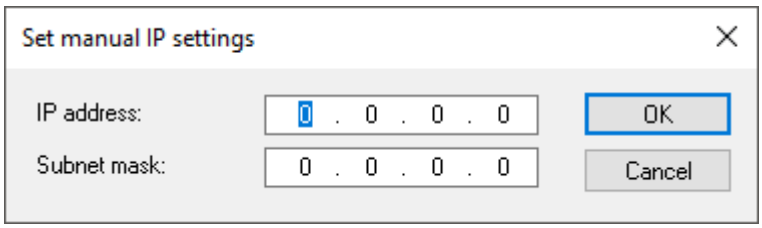
8. Select the IP stack and click **OK**.



9. If no IPv4 address has been assigned to the network port yet, answer the question whether you want to assign a fixed IP address.

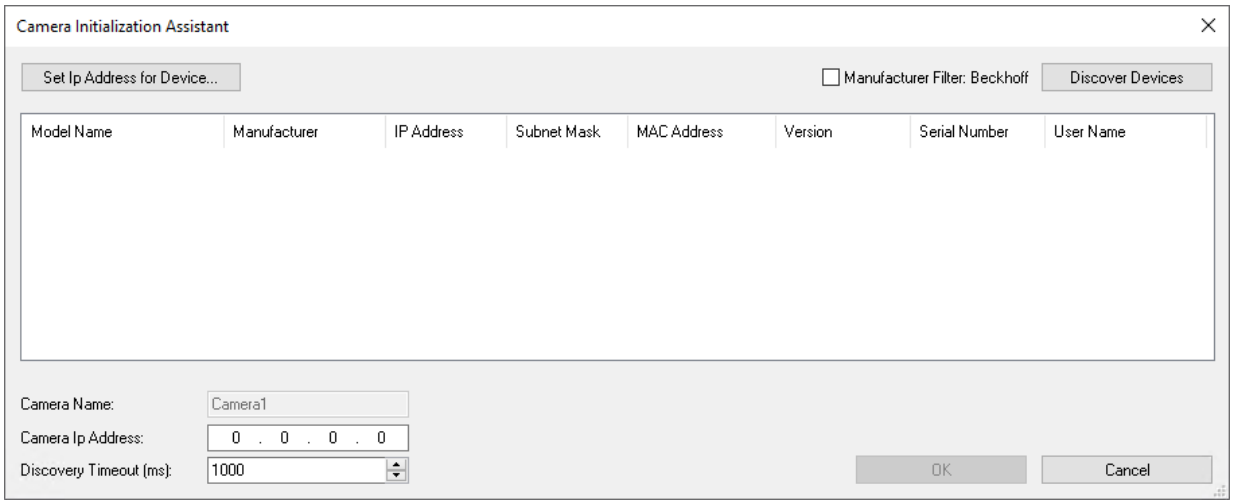


If Yes, set the IP address in the following dialog.

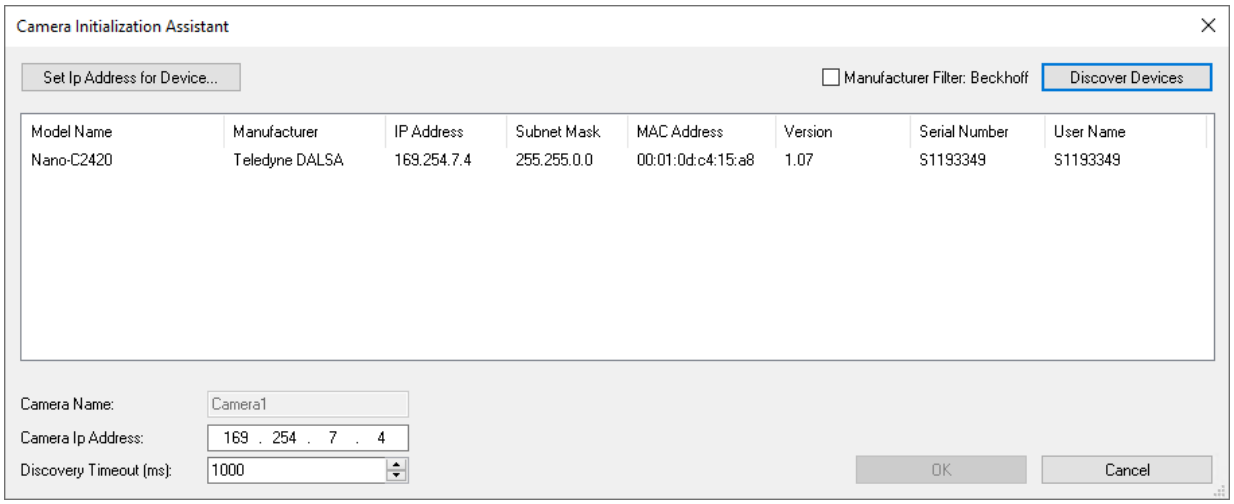


The IP address and subnet mask are then saved in the TwinCAT IP configuration of the network adapter (see Network configuration [▶ 59]).

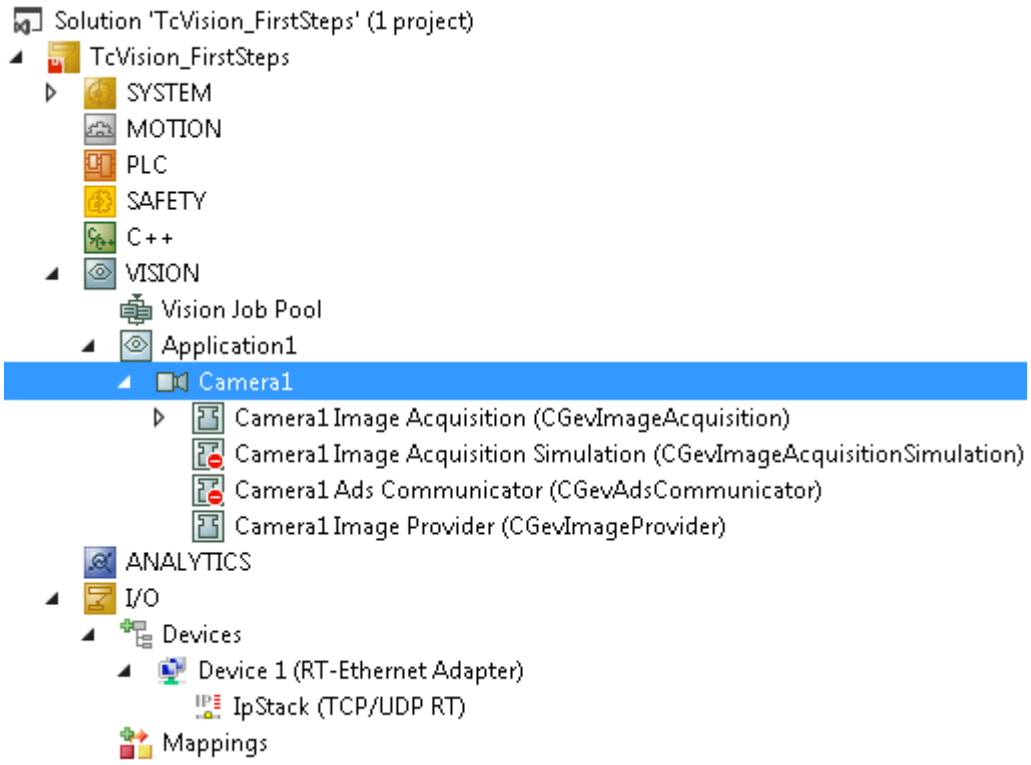
10. In the **Camera Initialization Assistant** click **Discover Devices**. The selected network port is searched for connected cameras.



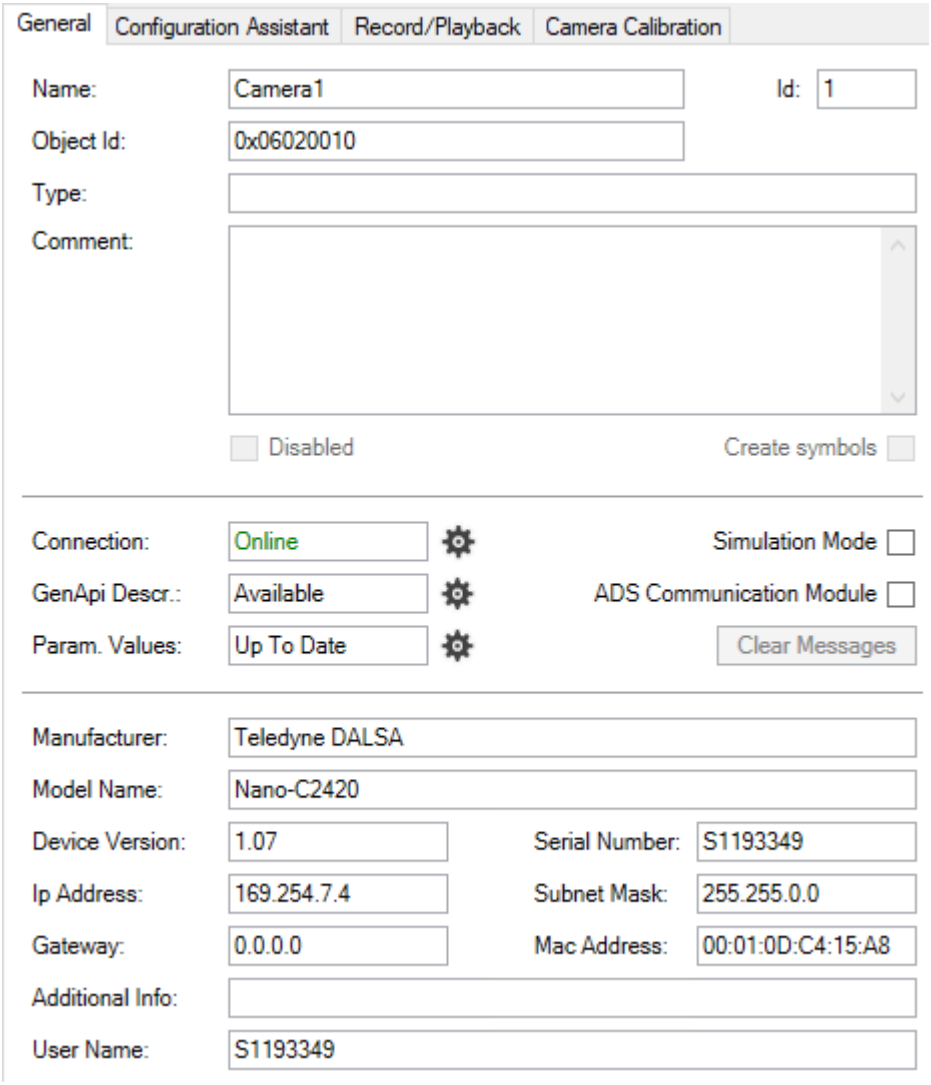
11. All cameras found are displayed depending on the Manufacturer Filter setting. Select the desired camera and click **OK**.



⇨ The camera is now created under the application and can be used.



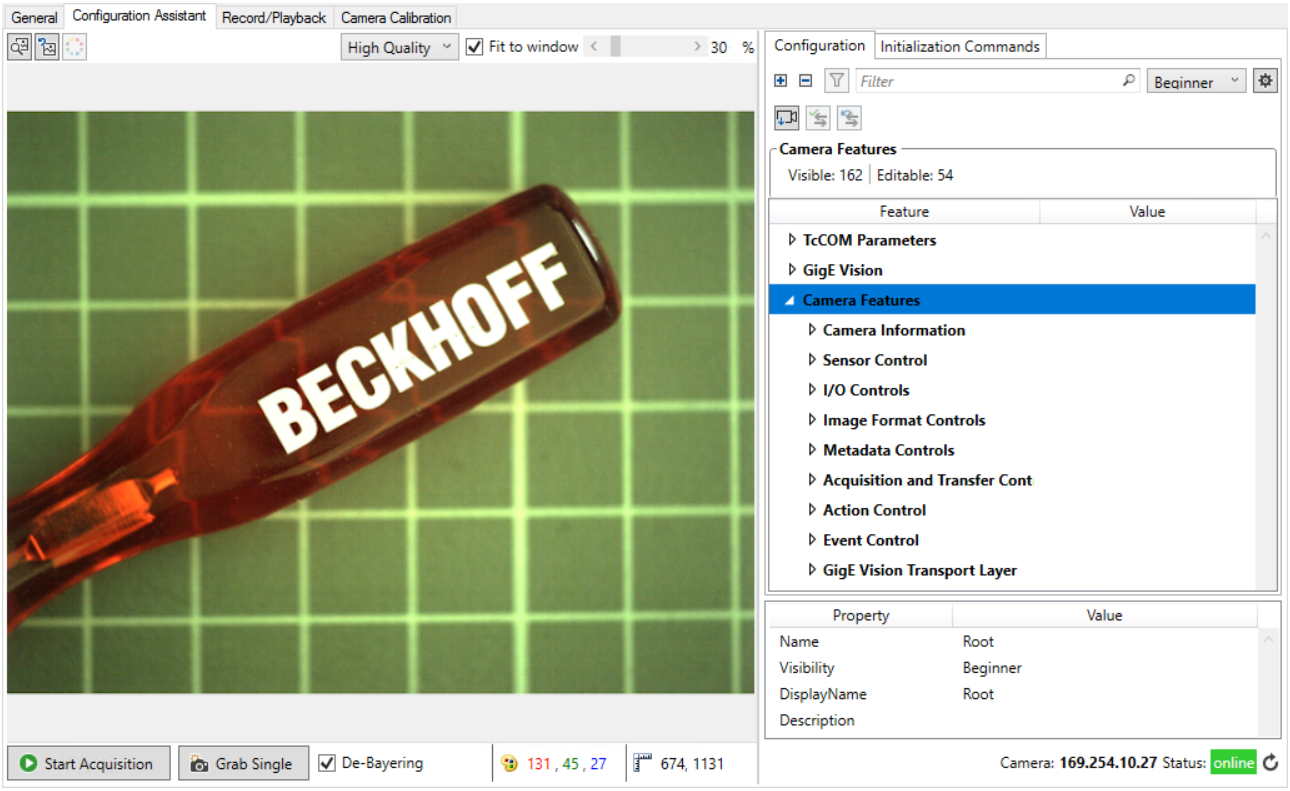
Double-clicking the camera node [▶ 69] opens a window in which the connection to the camera can be verified. The General [▶ 71] tab provides general information about the camera.



In the Configuration Assistant [▶ 75] you can use **Start Acquisition** to view the live image of the camera and make configuration settings on the camera.

See also: Camera configuration samples [▶ 2845]

If you want to apply the changed parameters and write them to the camera at every startup, create the Initialization Commands [▶ 87] at the end.



Next step

Creating a PLC project [▶ 45]

4.3 Creating a PLC project

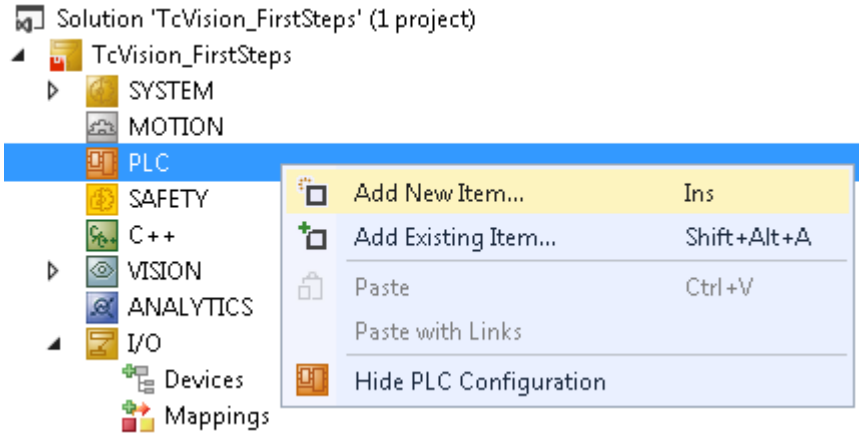
Video example for creating a PLC project with TwinCAT Vision:

Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/5977084043.mp4

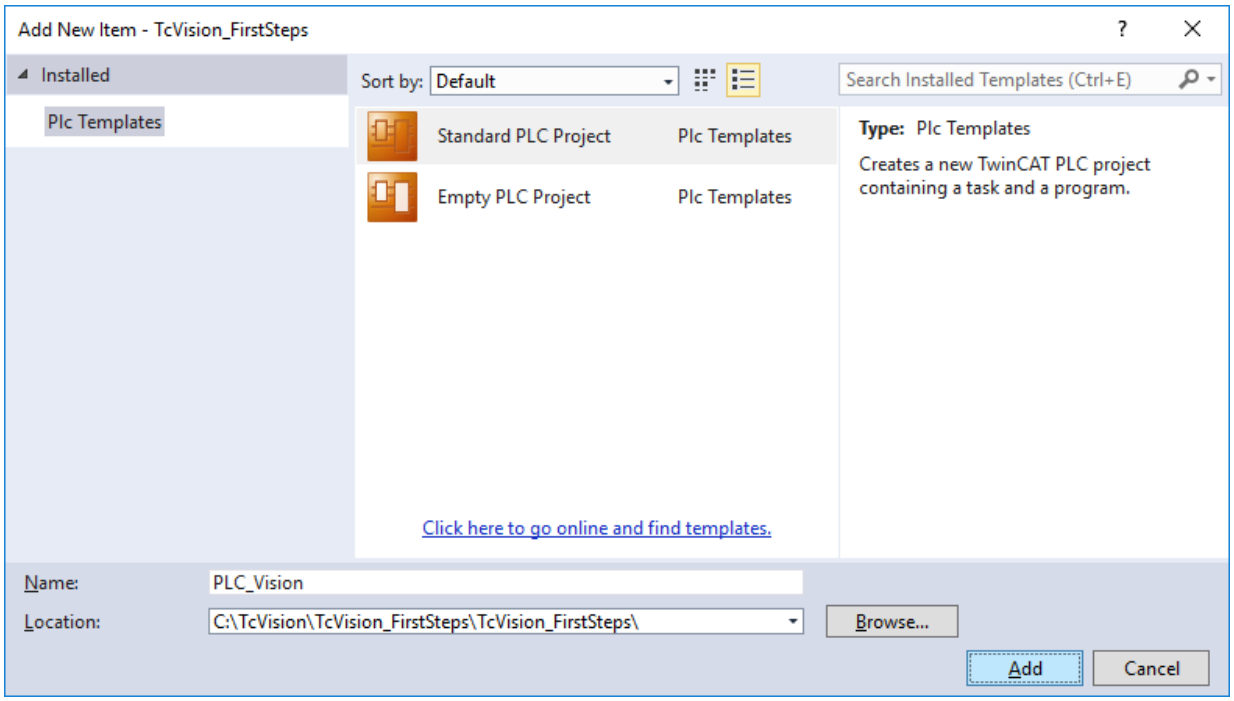
4.3.1 Creating a PLC project

Like the programming of the general machine control, programming of the image processing also takes place in the PLC. To this end, the TwinCAT project must be extended by a PLC project.

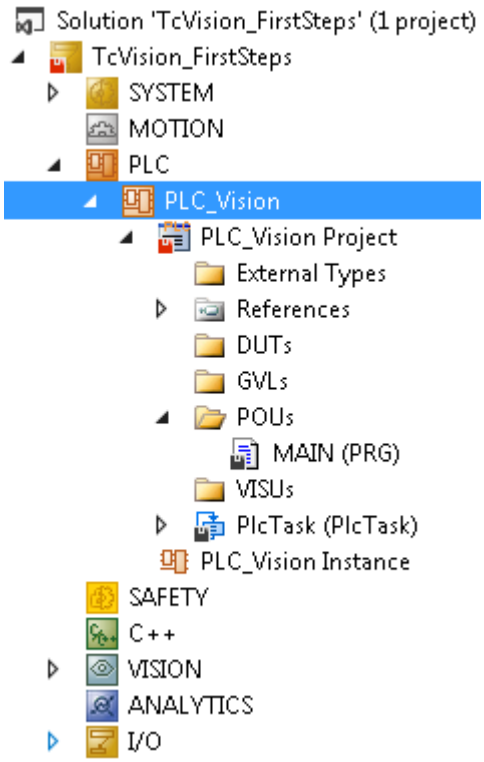
1. Right-click the PLC node in the Solution Explorer and select **Add New Item...**



2. Select the name and location in the **Add New Item** dialog.



⇨ The PLC project was created.



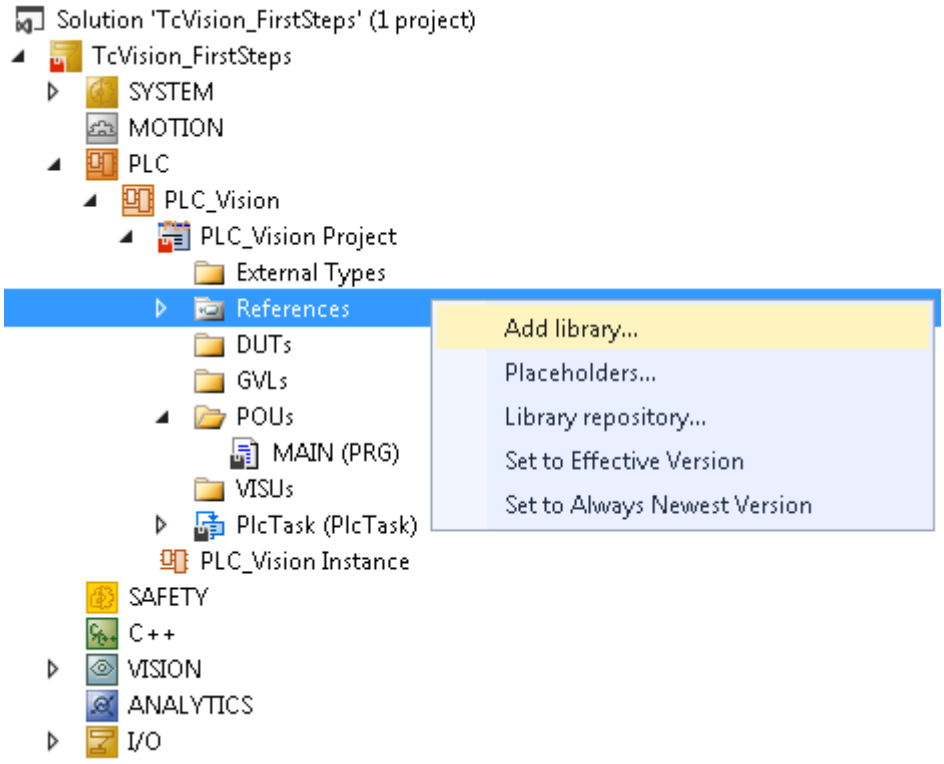
Next step

Integrating a PLC library [▶ 46]

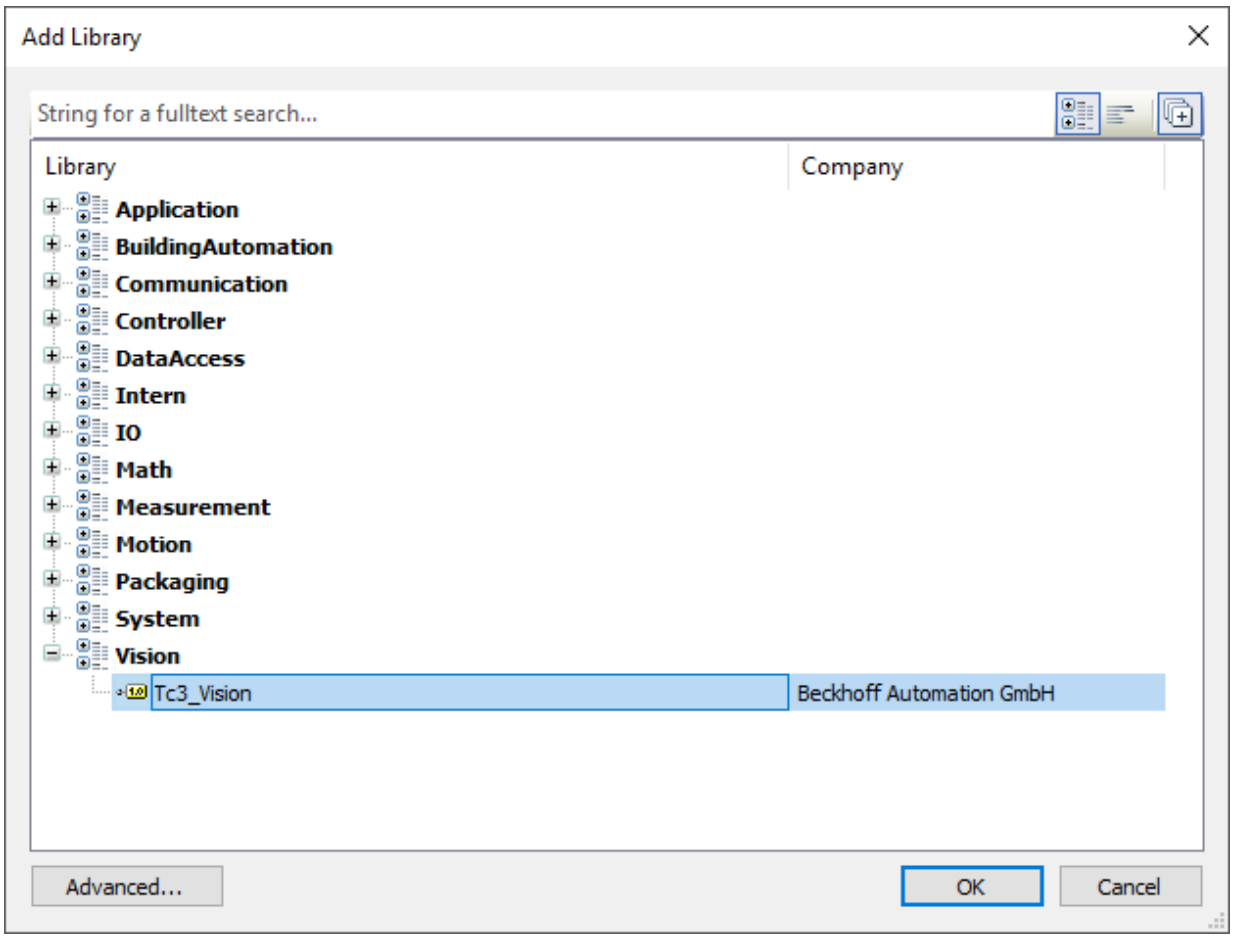
4.3.2 Integrating a PLC library

The Tc3_Vision library contains the data types, functions and function blocks required for image processing.

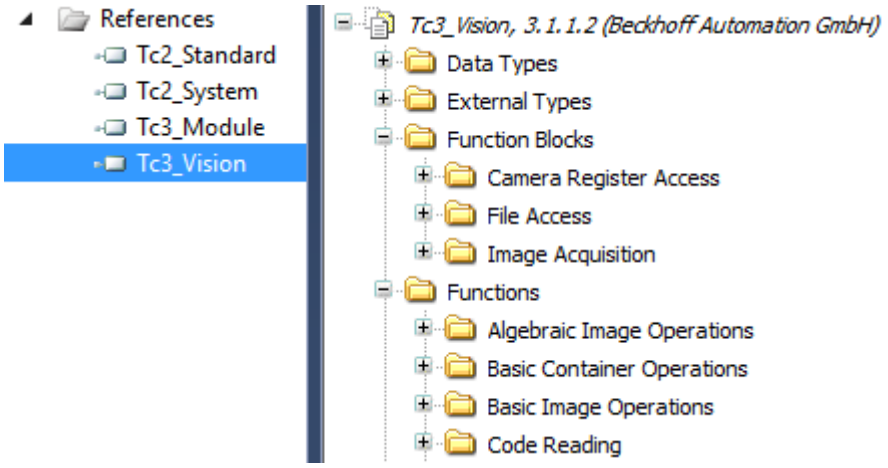
1. To add them to the PLC project, right-click on **References** and select **Add library...**



2. Select the **Tc3_Vision** library and confirm with **OK**.



⇨ The library is now added and can be viewed by double-clicking it.



Next step

Writing a PLC program [▶ 48]

4.3.3 Writing a PLC program

A simple PLC program for continuous image acquisition from a camera or a file source includes the following:

Variables

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

Name	Type	Description
hr	HRESULT	All functions and many methods of the Tc3_Vision library return a result code of type HRESULT for error handling. Further information about the concept behind it and the meaning of individual codes can be found under HRESULT [▶ 141].
fbCamera	FB_VN_SimpleCameraControl	Each camera and File Source Control is represented by a function block instance. In the simplest case this is an instance of the FB_VN_SimpleCameraControl [▶ 1646], which covers the common properties of camera and File Source Control and can therefore be linked with both device types. Alternatively, FB_VN_GevCameraControl [▶ 1621] can be used for cameras and FB_VN_FileSourceControl [▶ 1613] for File Source Controls.
eState	ETcVnCameraState	Each Vision device has a state of type ETcVnCameraState [▶ 168].
ipImageIn	ITcVnImage	Images for processing are managed using ITcVnImage [▶ 411] variables. In this sample, access to the camera input image is gained via ipImageIn.
ipImageInDisp	ITcVnDisplayableImage	Images for display are managed via ITcVnDisplayableImages [▶ 411].

Program code with state machine

Each Vision device has an internal state, `TCVN_CS_INITIAL` being the initial state. From this, the device can be brought into the `TCVN_CS_ACQUIRING` state using the `StartAcquisition` method. In this state, images can be captured and transferred. If an image is completely stored in the memory, it can be retrieved via `fbCamera.GetCurrentImage(ipImageIn)`.

To check if an image was received completely and could be retrieved, the IF query queries the `HRESULT` of the method and whether the pointer to the memory area of the image is not 0. After that, you can start analyzing the image. To check and indicate that an image could be retrieved, the counter variable `nNewImageCounter` is incremented as an example.

F_VN_TransformIntoDisplayableImage [▶ 812] then moves the image pointer to `ipImageInDisp` to display the image. The image can then no longer be retrieved via `ipImageIn`. Alternatively, a copy can be created with F_VN_CopyIntoDisplayableImage [▶ 781] to display and edit the image.

```
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
```

Further information on the state machine can be found in the API chapter Image Acquisition [▶ 1611].

Further information

- API reference [▶ 137]
- Function blocks for image acquisition [▶ 1611]
- Images [▶ 149]

Next step

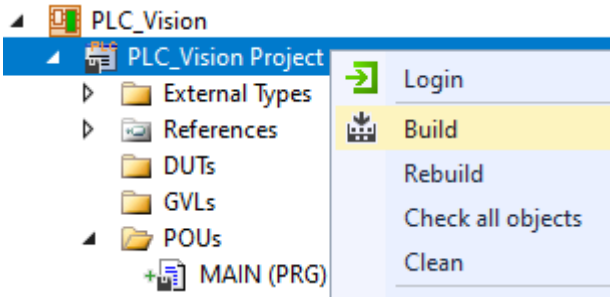
Initializing a function block [▶ 49]

4.3.4 Initializing a function block

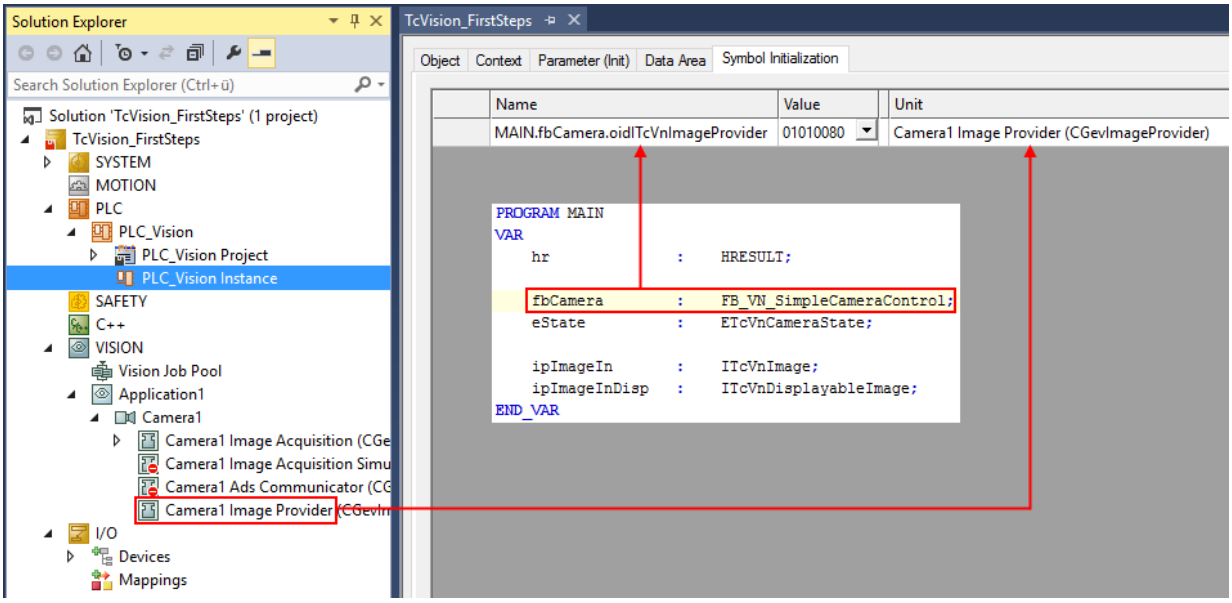
Following the configuration of a Vision device and the creation of a PLC program, the Vision device must now be linked with appropriate camera variables in the PLC.

The instance `fbCamera` of the function block FB_VN_SimpleCameraControl [▶ 1646] from the PLC program must be linked with the image provider of the Vision device. Please note:

1. Build the PLC program [▶ 48] by right-clicking the PLC project and selecting **Build**. Make sure that no errors occur when doing this. If this is the case, the **Symbol Initialization** tab appears in the PLC instance and the `fbCamera` instance below it.



2. Double-click the **instance of the PLC project** and open the **Symbol Initialization** tab.
3. Assign the Image Provider of your Vision device to the symbol `MAIN.fbCamera.oidITcVnImageProvider` via the drop-down in the Value column. This is shown for a camera object in the image, but it works in exactly the same way for File Source objects.



Next step

Activate TwinCAT project and start PLC project [▶ 50]

4.4 Run

Video example for executing a TwinCAT Vision PLC project:

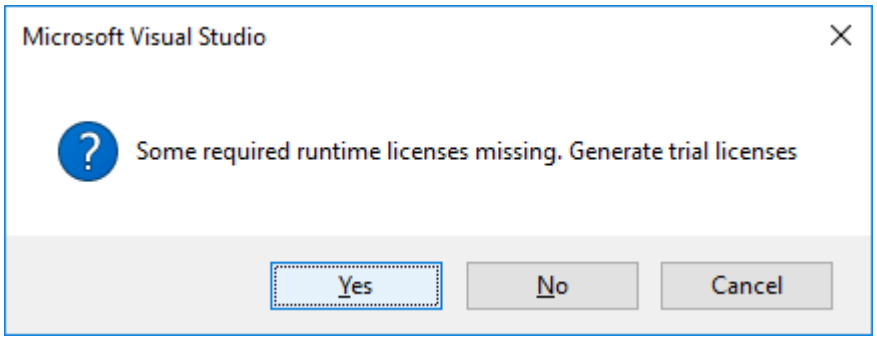
Video:: https://infosys.beckhoff.com/content/1033/tf7xxx_tc3_vision/Resources/5977086347.mp4

4.4.1 Activate TwinCAT project and start PLC project

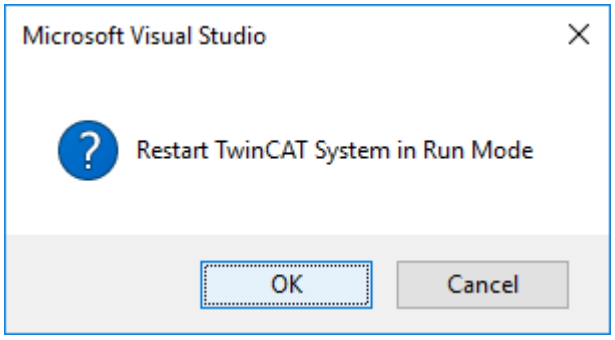
You can execute the project by performing the following steps:

1. Activate the configuration via  or via the Visual Studio menu under **TwinCAT** > **Activate Configuration**.

2. If no licenses have been activated yet on the target system, the following dialog appears. For an initial test you can create a trial license with **Yes** and enter a security code.



3. You will then be asked whether you want to restart TwinCAT. Confirm with **Yes**. Alternatively, a restart can be triggered via  or via the Visual Studio menu under **TwinCAT** > **Restart TwinCAT System**.



4. If the PLC is not set to autostart, start it manually. To do this, log into the PLC via  or via the Visual Studio menu under **PLC** > **Login**.

5. Then start the PLC via  or via the Visual Studio menu under **PLC** > **Start**.

Next step

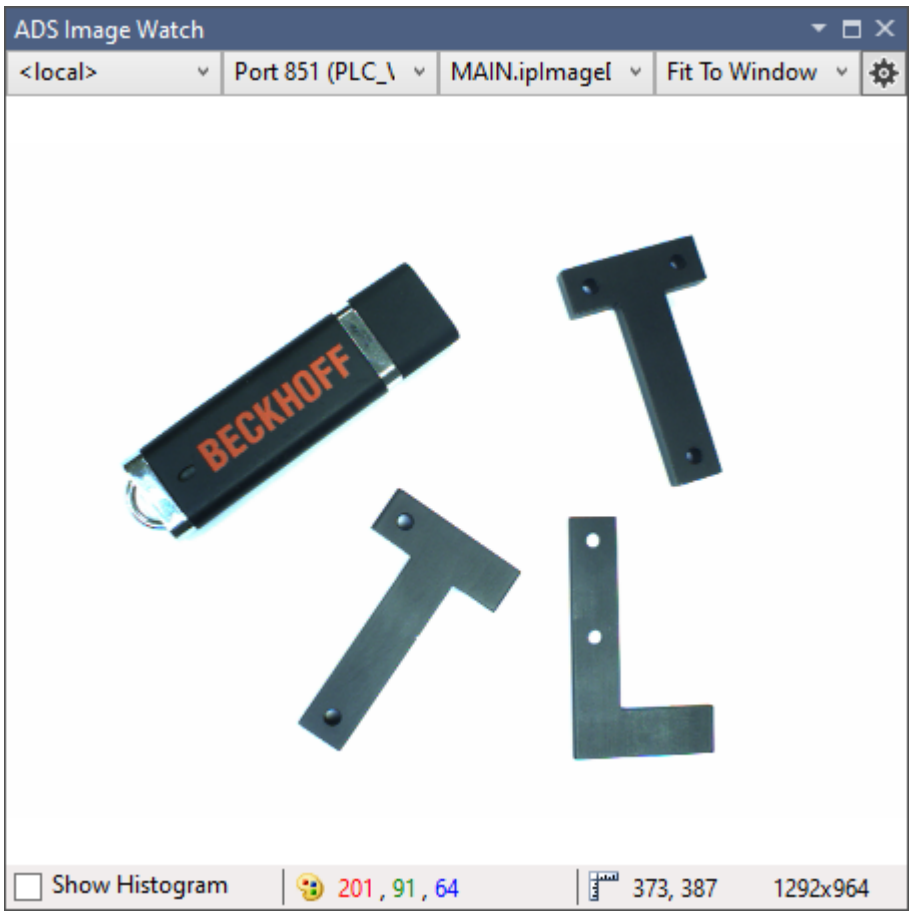
Image display in ADS Image Watch [▶ 51]

4.4.2 Image display in ADS Image Watch

Images of type `ITcVnDisplayableImage` in the PLC can be displayed in ADS Image Watch. You can find the window in the Visual Studio menu **TwinCAT > Windows > ADS Image Watch**.

To select the image to display, set the following parameters from left to right:

- Target system
- Port
- ADS symbol of the image
- Scaling of the image

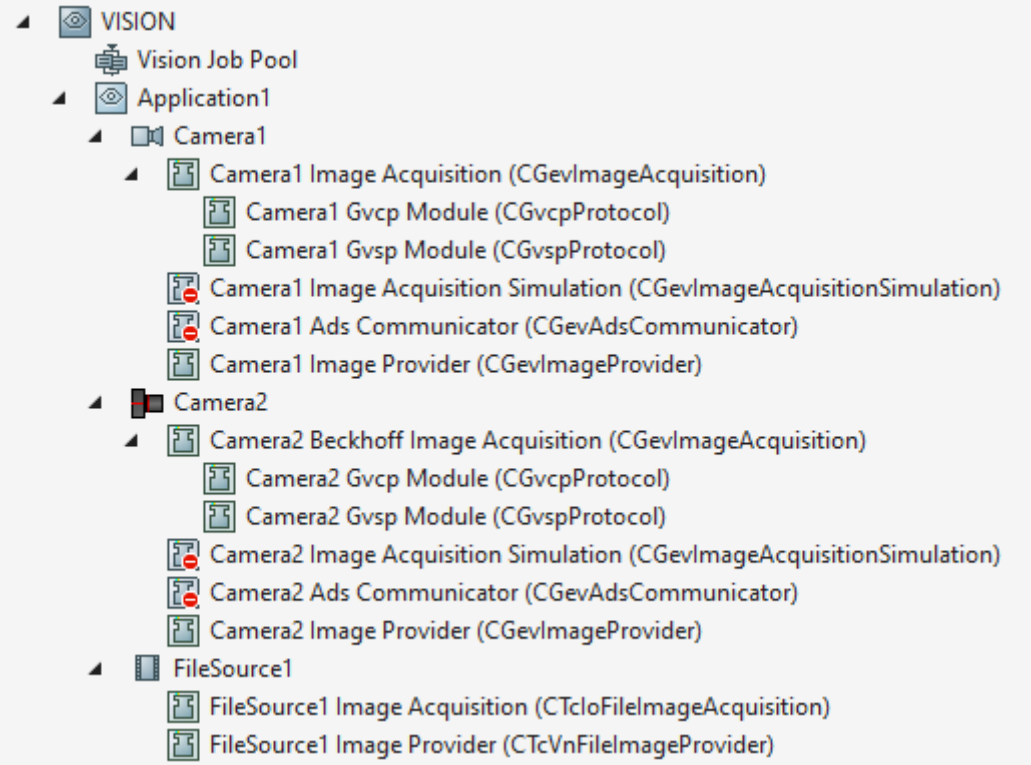


In addition to the image content, the following information can be read in the ADS Image Watch:

- Histogram of the image
- Color/intensity values at the current cursor position
- Pixel coordinates of the current cursor position
- Image size

5 Development environment

TwinCAT Vision is fully integrated in the TwinCAT 3 development environment. Essentially, TwinCAT Vision extends the TwinCAT system by a further node, below which the Vision components can be created and configured. The following configuration elements are used:



- The VISION node [▶ 65] provides general configuration options that affect all Vision components on the system. It includes all other configuration elements.
- The Vision Job Pool [▶ 131] can be used to define job tasks for parallel execution of Vision functions. Each system has a single Vision Job Pool.
- An Application node [▶ 68] encapsulates several Vision devices (GigE Vision cameras and File Sources) to form an applicative unit. The entire configuration of an application can be exported and imported. Each TwinCAT project can have several applications.
- A GigE Vision camera node [▶ 69] contains the configuration of a physical GigE Vision camera. Each application can have several GigE Vision camera nodes.
- A Beckhoff camera node [▶ 125] contains the configuration of a physical Beckhoff camera. Each application can have several Beckhoff camera nodes.
- A File Source node [▶ 125] contains the configuration for an interface between images on the hard disk of the target system and an image acquisition module within the PLC. Each application can have several File Source nodes.

Communication with system components

GigE Vision camera nodes, Beckhoff camera nodes and File Source nodes each contain TcCOM modules for communication with various system components.

Furthermore, the development environment features an ADS Image Watch [▶ 131] window element for displaying images from the PLC via ADS. In addition, a general system configuration [▶ 54] is necessary for the correct functioning of the Vision components.

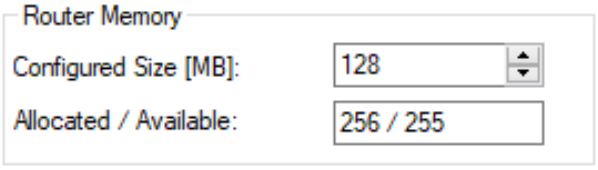
In order to be able to display images from the PLC live in a TwinCAT HMI, a corresponding HMI extension [▶ 2704] is available.

5.1 System configuration

The following sections describe the settings specific to TwinCAT Vision, which should be carried out or checked in relation to the system and project.

5.1.1 Router memory

Compared to conventional control systems, image processing applications require a lot of memory, because even a single image can contain several megabytes of data. In TwinCAT Vision, all dynamic data (see Interface Pointer [▶ 137]) are allocated in the router memory. For this reason, it is particularly important to configure the size of the router memory accordingly. It can be set in the TwinCAT system configuration under the Real-Time node in the **Settings** tab:



Configured Size	Setting of the router memory required for the current project.
Allocated	Router memory currently allocated on the system.
Available	The part of the router memory currently allocated on the system that is still freely available. (The rest is already used within TwinCAT.) **If the available memory is not fully restored following a TwinCAT restart, this points to a memory leak. In that case, restart your system and check your program with regard to memory leaks.**

The maximum adjustable size of the router memory in TwinCAT 3.1 4024 is 1024 MB. From TwinCAT 3.1 4026, the usable size essentially depends on the available main memory and is limited to 65535 MB. The actually required memory size depends among other things on the use of dynamic memory (e.g. with images [▶ 149] and containers [▶ 152]) in the PLC program.

NOTICE

Size of the router memory

When configuring the router memory, it must be noted that the router memory will be subtracted from the main memory and is thus no longer available to the operating system. Therefore we recommended avoiding the size of 1024 MB in systems with a main memory size of less than 4 GB. With external projects, always check the router memory setting before activating the configuration and reduce it if necessary.

Estimating the required memory size

The required size of the router memory depends on many factors. A rough estimate for the Vision part of the TwinCAT project can be obtained by calculating the memory dynamically allocated (by Interface Pointers [▶ 137]).

The storage space `M` (in bytes) of an image can be calculated by incorporating the image size (width `w` and height `h`), number of channels `C` and the number of bytes of the element type: `M = w * h * C * sizeof(ElementType)`.

For a container of the type String or Vector, the storage space `M` (in bytes) can be calculated through the number of allocated elements `N` and the number of bytes of the element type: `M = N * sizeof(ElementType)`.

Attention must be paid to how the data are copied and released. If an image is copied or a new one is generated, the storage space for both instances must of course be included in the calculation. Moreover, the memory requirement of the router itself and other project parts must be included in the calculation. When retrieving the images or other data via ADS, these are copied, so additional storage space must also be taken into account for this.

NOTICE

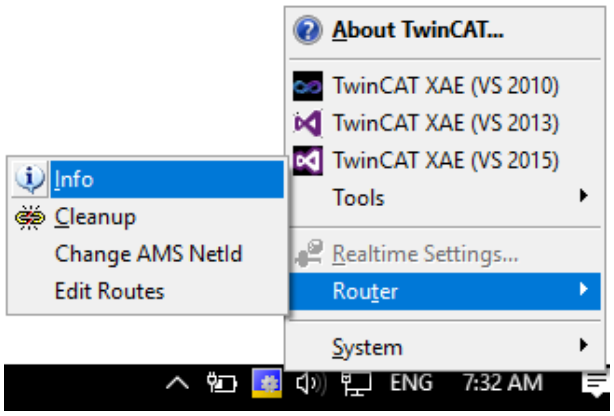
Memory value with safety factor

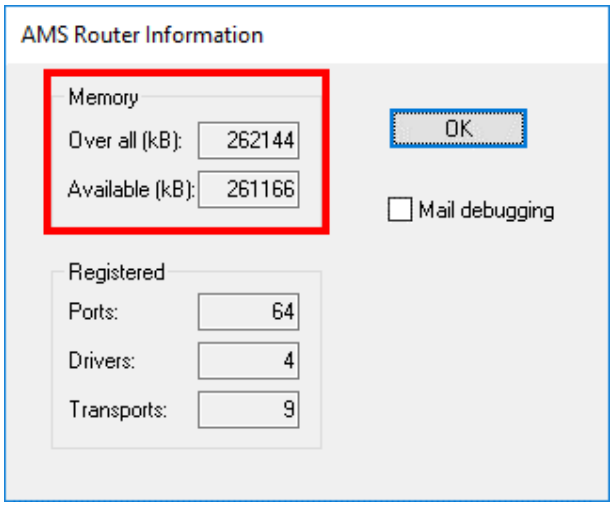
Always extend the calculated memory value with a safety factor. Also take into account the memory requirement of the other modules in the project.

So as not to use an unnecessarily large amount of space in the router memory, you should take care not to make any unnecessary deep copies and to release objects again that are no longer required (see Interface Pointer [▶ 137]). In order to find out how much memory is actually being used, the router memory can be monitored live as follows.

Monitoring the router memory

Even if the router memory is set in the project, this is a system-wide setting. Therefore, the router memory set in the project and currently used on the system can be viewed on the target system. To do this, right-click the TwinCAT icon in the Windows taskbar and select **Router > Info**.





NOTICE

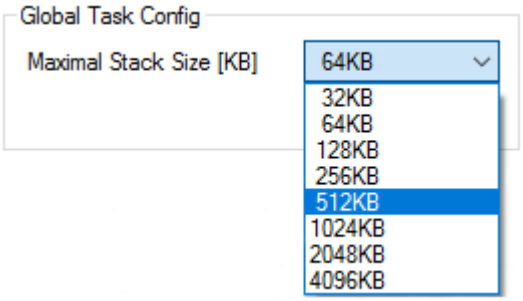
Reduction of the available router memory

The available router memory may be reduced through Exceptions or unreleased Interface Pointers. Therefore, the memory should be checked during the development, in particular in the case of an error. If not enough memory is available, restart the complete system and then check your program.

5.1.2 Stack size

The default size of 64 KB is usually sufficient for image processing applications. When using larger image formats or special functions such as the F_VN_Clahe [▶ 1239] this value is too small and the stack size must be increased. Then it is recommended to use 512 KB or larger directly.

It can be set in the TwinCAT system configuration under the Real-Time node in the **Settings** tab:



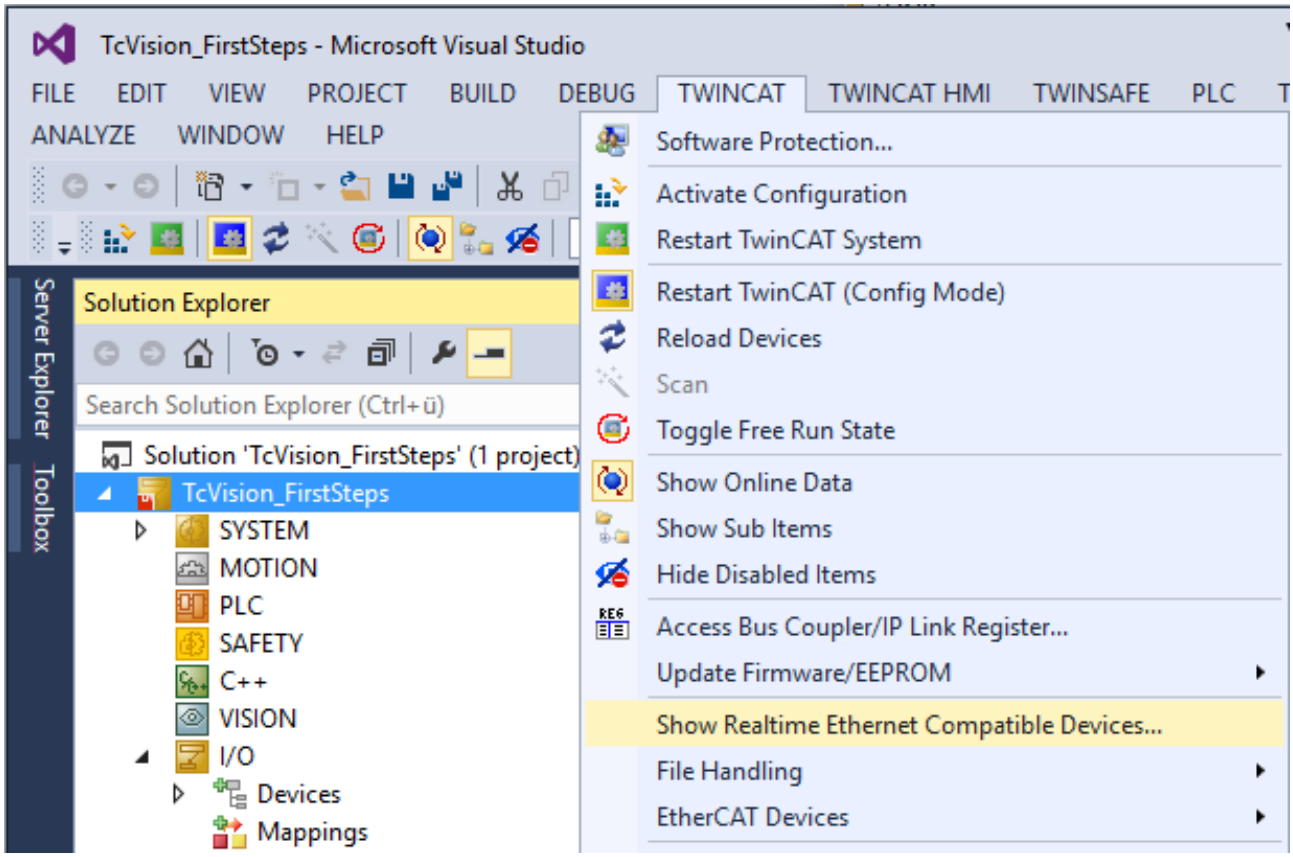
5.1.3 Network adapter

There are several things to consider when selecting, configuring and using a network adapter, which are listed and described individually below:

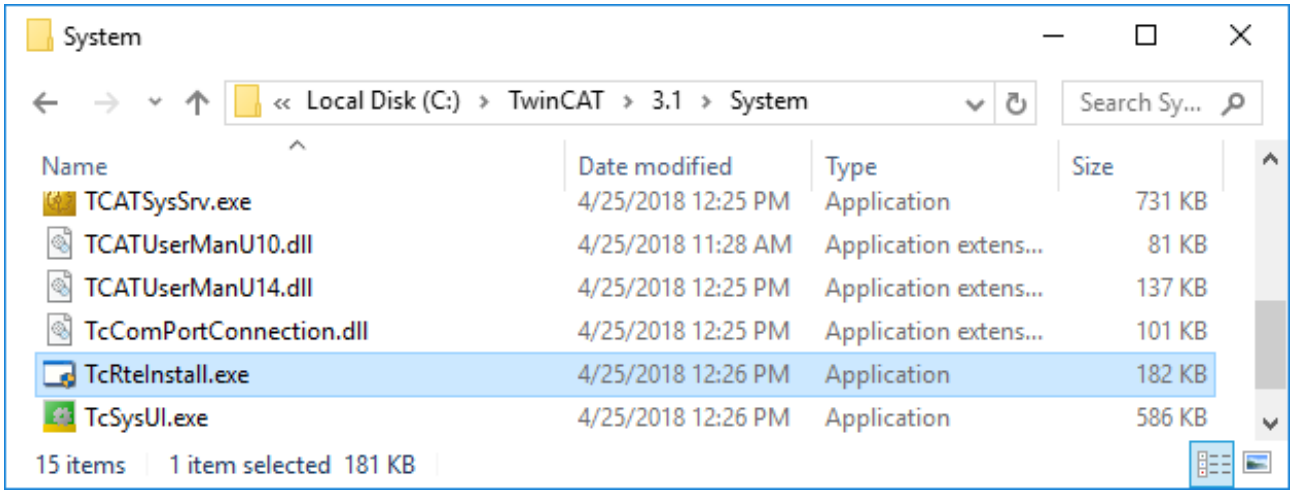
- First, a suitable network adapter that is supported by our TwinCAT Realtime Ethernet driver must be chosen so that the driver can be installed and used in real-time operation.
- All other network protocols that are not required should then be disabled for this network adapter in order to prevent interferences in communication.
- After setting up a camera for the first time, a persistent IP address should be set in the IP-Stack of the TwinCAT project.
- If several cameras are to be operated on the same port of the network adapter, additional settings must be made in the IP-Stack.
- Finally, you will find a few notes on the diagnostic options.

TwinCAT RT Ethernet adapters

Similar to the connection of EtherCAT devices, the TwinCAT Realtime Ethernet driver must be installed on the corresponding network adapter to enable connection to GigE Vision devices. To open a corresponding installation dialog, click **TWINCAT > Show Realtime Ethernet Compatible Devices** in the Visual Studio menu.



Alternatively you can call the file *C:\TwinCAT\3.1\System\TcRteInstall.exe* on the target system; this also opens the corresponding installation assistant.

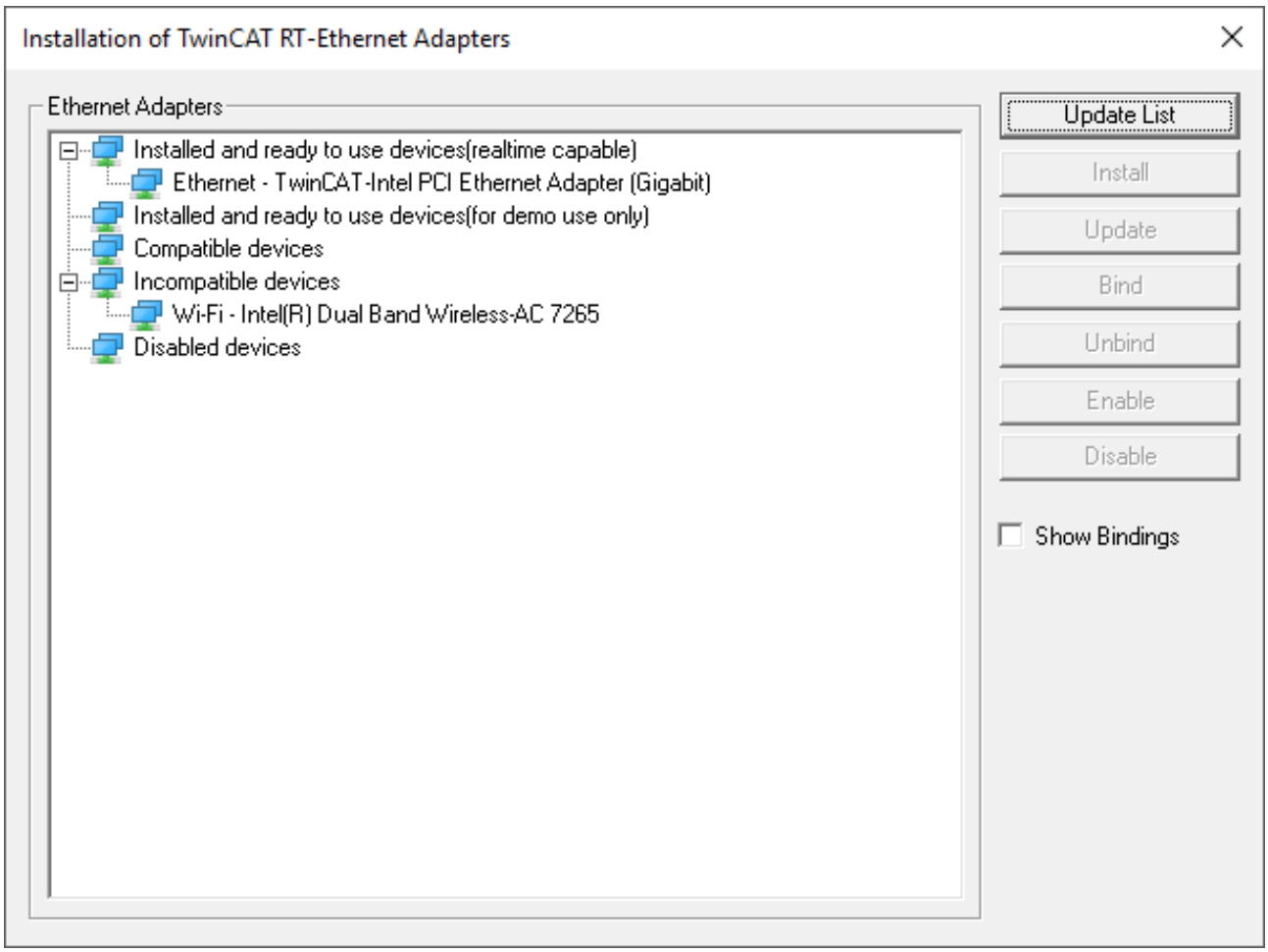


In the installation assistant, make sure that the corresponding network adapter is listed under **Installed and ready to use devices (realtime capable)**. If this is not the case, select an appropriate adapter from the list of **Compatible devices** and click **Install**.

NOTICE

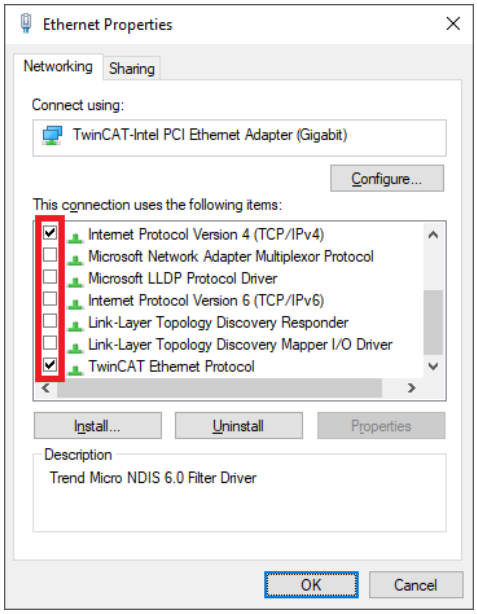
For demo use only

This status is not sufficient for a stable connection and operation of GigE Vision devices. If the network adapter is only listed under this group after installation, a different adapter must be used.



Other network protocols

It is recommended to restrict the use of the network adapter to the TwinCAT Ethernet protocol and to assign a persistent IP address via the IP-Stack in the TwinCAT project. Because if several protocols have access to the network adapter, they could send packets that could cause errors in the camera communication. The setting for this can be found in the Windows properties of the network adapter.



Internet Protocol version 4 (IPv4) is only required if DHCP is to be used to set up the camera connection. When setting up a camera for the first time, the automatic assignment of a link-local address (LLA) can be helpful. In all other cases or after completing the initial settings, you can also disable the IPv4 protocol and configure the IP address manually in TwinCAT in the IP-Stack [▶ 59].

NOTICE

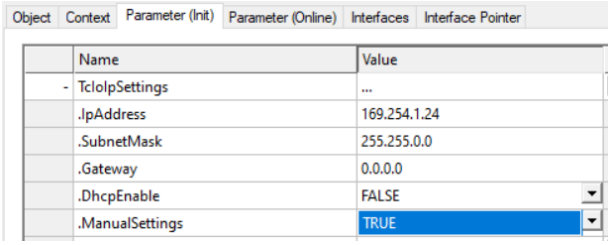
Disabling other network protocols

If possible, disable all network protocols on your Ethernet adapter except for the TwinCAT Ethernet Protocol. Otherwise, errors and delays may occur in the communication with GigE Vision cameras.

Specifying a persistent IP address

If the PC should not automatically specify the IP address of the network adapter, but instead you wish to manually specify a persistent IP address, you can do this in TwinCAT as follows:

1. Open the **IP-Stack** of the RT-Ethernet adapter in the I/O configuration.
2. Select the **Parameter (Init)** tab and display further parameters by clicking the **Plus (+)** in front of `TcIoIpSettings`.
3. Specify the parameters `.IpAddress` and `.SubnetMask`.
4. Set the parameter `.ManualSettings` to `TRUE`.



The same setting is automatically assigned if you assign a manual IP address when creating a camera object [▶ 38]. Alternatively, a persistent IP address can also be defined via the Windows Network Adapter properties.

NOTICE

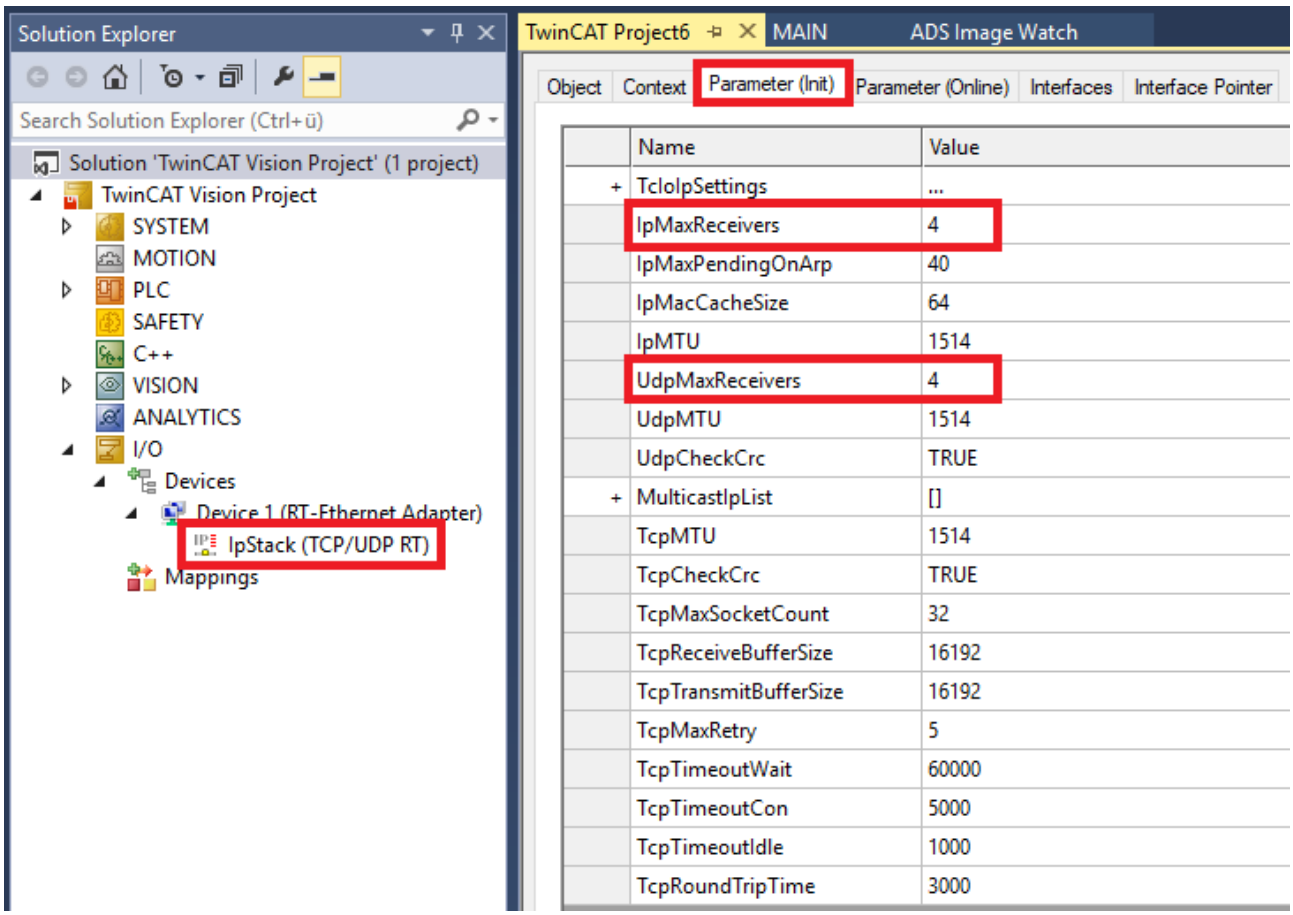
Persistent IP addresses

To prevent the IP address of the network adapter from changing, specify a persistent IP address in the IP-Stack of the TwinCAT project.

Operating multiple cameras on one port

If you want to operate several cameras on the same RT Ethernet adapter, you must make sure that the UDP/IP settings allow this. Navigate to the UDP/IP stack under the RT Ethernet device and select the **Parameters (Init)** tab.

In the parameter list that appears, set `IpMaxReceivers` and `UdpMaxReceivers` such that they correspond to twice the number of cameras connected via this adapter. This is because there are two receivers per camera (GVCP and GVSP modules). The standard setting allows two camera connections, for which reason the parameters are set to `2 * 2 = 4`.



Advanced Settings

Maximum Transmission Unit (MTU)	The maximum permitted size of the frame to be transmitted is 1514. Thus, jumbo frames are not currently supported.

Diagnostics

In case of problems with the connection, especially with sent or received frames, you will find a tab with statistics information in the RT Ethernet adapter and in the IpStack the **Parameter (Online)** tab with individual counter information. Details can be found in the appendix/troubleshooting at Camera communication [▶ 2877].

Promiscuous Mode

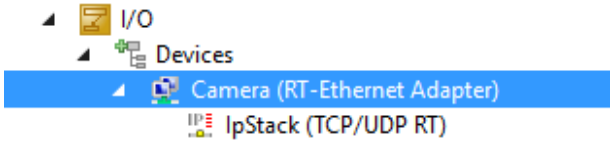
Promiscuous Mode is a setting in TwinCAT RT Ethernet adapters. It is required for debugging purposes with the Wireshark tool. This mode can cause problems when communicating with GigE Vision devices.

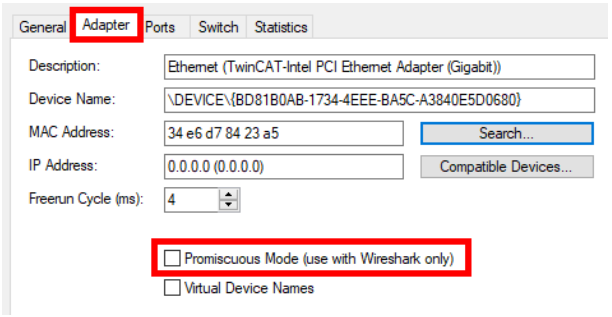
NOTICE

Avoid Promiscuous mode

This mode can cause problems when communicating with GigE Vision devices. GigE Vision devices should therefore be operated without Promiscuous Mode!

You can check this setting by opening the corresponding RT-Ethernet adapter in Visual Studio and switching to the **Adapter** tab. The checkbox for **Promiscuous Mode (use with Wireshark only)** must be deactivated.





5.1.4 CPU cores and tasks

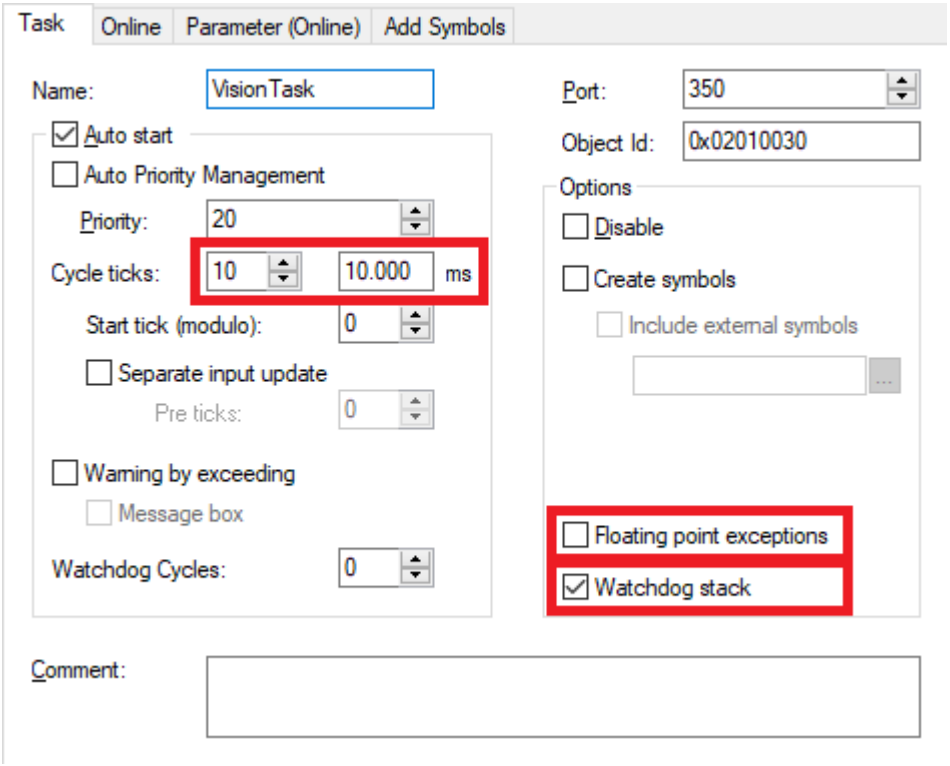
The configuration of tasks and CPU cores is not specific to TwinCAT Vision, but is the same for all TwinCAT components. For instructions on how to do this, see Real-time settings and Task settings. The issues and considerations relating to TwinCAT Vision are described below.

A TwinCAT Vision application requires tasks for various purposes:

- Cyclic task for executing the PLC
- Cyclic task for operating Vision devices
- Job Task for parallelized PLC execution
  (Optional; this type of task is described separately in the chapter Job tasks [▶ 64]).

Configuring PLC tasks

Basically, all cyclic tasks required for TwinCAT Vision are created automatically. When a PLC is created, a task with a cycle time of 10 ms is created. This means that image processing sequences in the PLC are repeated every 10 ms. You can change the task settings under **SYSTEM > Tasks > [TaskName]**.



Cycle ticks	The cycle ticks multiplied by the base time of the selected CPU core (by default 1 ms) produces the above-described cycle time of the PLC.
Floating point exceptions	This setting specifies whether or not TwinCAT checks for floating point exceptions. For Vision applications it is recommended to deactivate this option. Besides that, these error messages do not indicate the exact reason for the exception.
Watchdog stack	This setting must be activated if watchdogs [▶ 146] are to be used in the assigned PLC program.

⚠ WARNING

Floating point exceptions

For PLC tasks that essentially execute Vision commands, it is urgently recommended that you deactivate the option "Floating point exceptions". Otherwise, unnecessary errors could occur with certain TwinCAT Vision API functions and lead to a system abort. However, this option must be active if Motion Control or Safety components are also executed on the same PLC task.

NOTICE

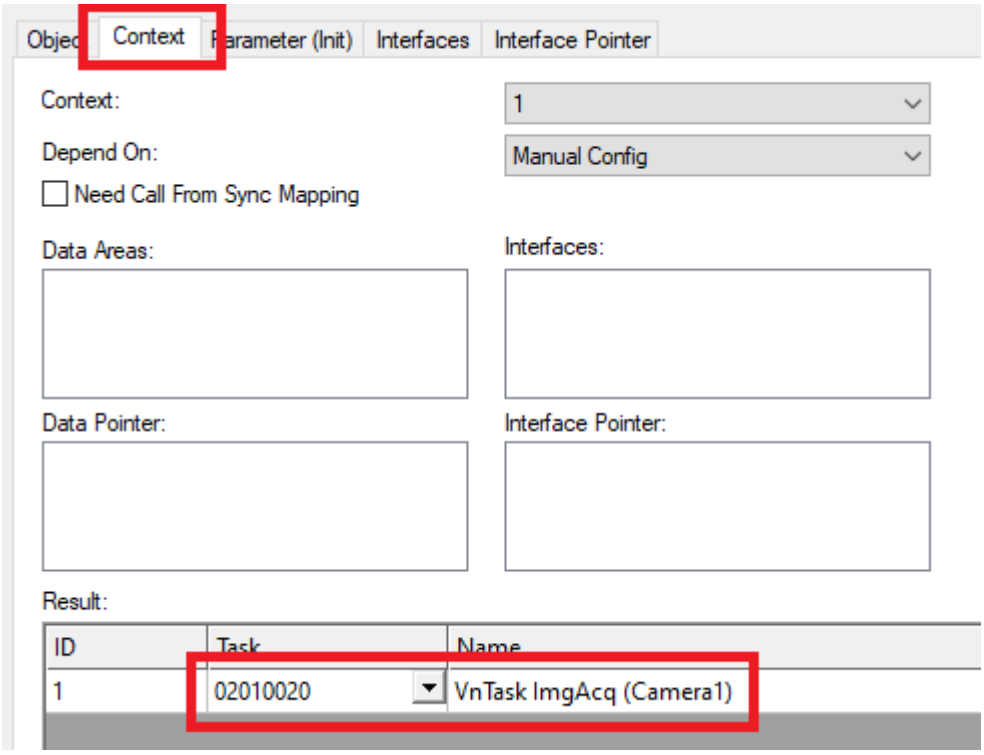
Floating point exceptions in Job Tasks

If you also use Job Tasks [▶ 64] for Vision applications and have disabled the "Floating point exceptions" in the PLC task, you must also disable this option in the assigned Job Tasks. Otherwise exceptions and system crashes can still occur.

Configuration of device tasks

The required tasks are generated automatically when creating Vision devices. The cycle times of these tasks can be adjusted as described above. For most cases it is recommended not to change the cycle times of these tasks. This is only necessary in special cases.

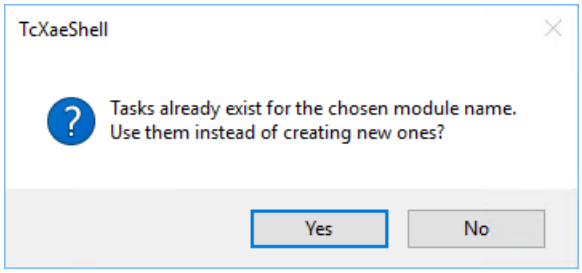
If the link of a Vision device with a task should have to be adjusted, this is done on the **Context** tab of the respective TcCOM objects of a GigE Vision camera [▶ 123] or File Source [▶ 130] object:



The following TcCOM objects must be linked to a cyclic task, if they are used:

- Camera Image Acquisition [▶ 123]
- Camera Image Acquisition Simulation [▶ 124]
- ADS Communicator [▶ 124]
- File Image Acquisition [▶ 130]

If suitable tasks already exist under the selected device name when creating a Vision device, they can optionally be reused. In this case the following window appears; click **Yes** to confirm the reuse:



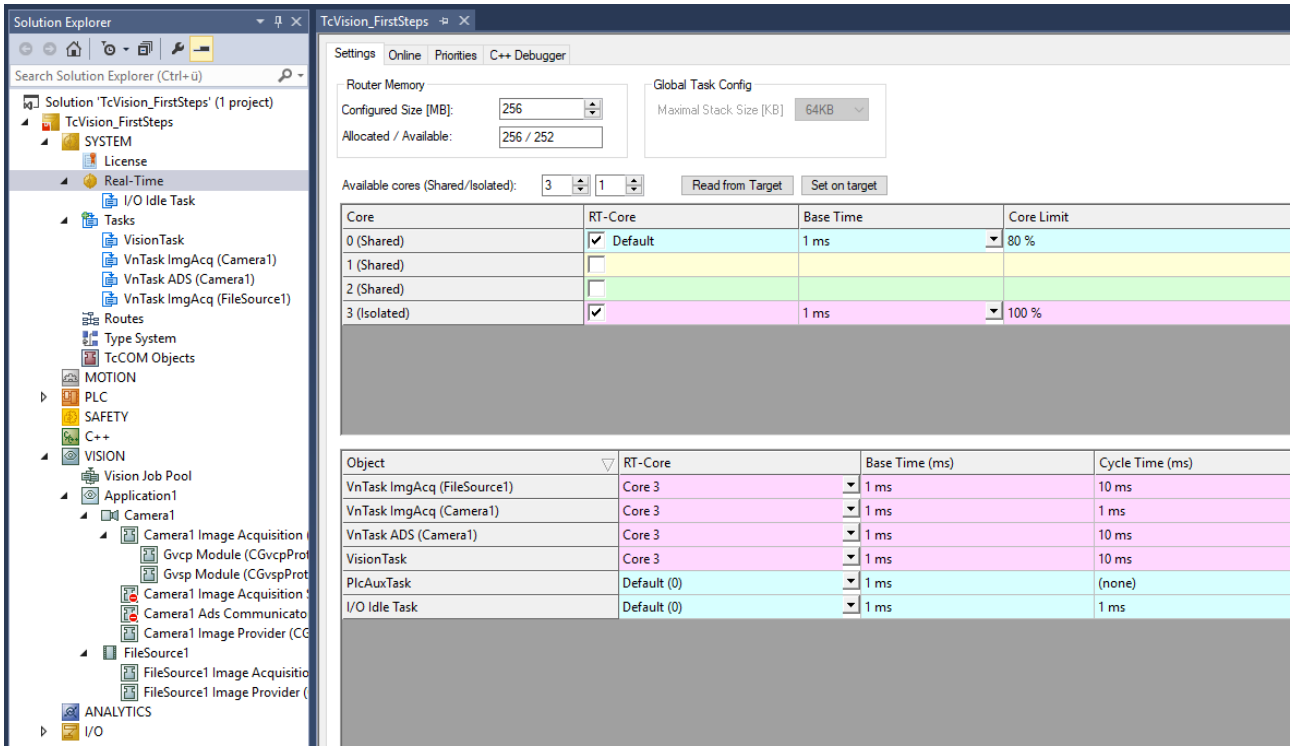
Reuse of device tasks

When reusing device tasks, it should be ensured that they are not used by several Vision devices at the same time.

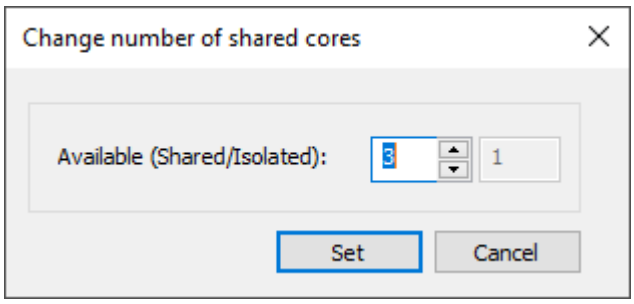
Assignment to CPU cores

The assignment of tasks to CPU cores takes place in the TwinCAT project under **SYSTEM > Real-Time > Settings**. In multi-core systems, TwinCAT enables a distinction between Windows cores and isolated cores. With Windows cores, the operating system and the TwinCAT application share the processor time. The real-time portion can be limited to between 10% and 90%. In contrast, isolated cores are fully available to the TwinCAT application. For this reason, the use of isolated cores is recommended for Vision applications.

To display the current CPU core configuration of the target system, click **Read from Target**.



To change the CPU core configuration of the target system, click **Set on Target** and select the desired distribution of divided and isolated cores.



After that the target system must be restarted.

Recommendation

We recommend using at least one dedicated, isolated core for the Vision tasks (device and PLC). If there is a large load on the TcVnService due to the saving or loading of images (by camera streams, a File Source Control or direct from the PLC), we recommend reserving one core entirely for Windows or severely limiting the real-time portion of a core.

If multiple Tasks are assigned to a core, priority must be considered. As a rule, tasks with shorter cycle times should be assigned a higher priority.

5.1.5 Job tasks

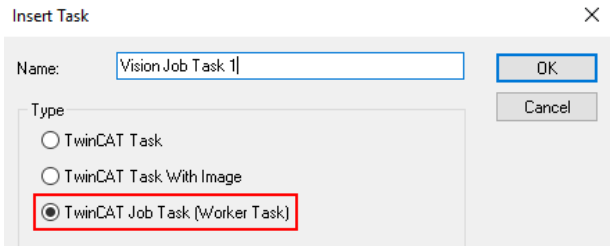
Job Tasks are used by TwinCAT Vision to parallelize functions. Depending on the system in use, its configuration, the API functions in use, the image content and the number of job tasks, this can lead to a shortening of the execution time. The supported functions are marked in the API reference [▶ 137] by the comment “Can use available TwinCAT Job Tasks for executing parallel code regions.” The Vision Job Pool [▶ 131] is used to assign which Job Tasks are used by TwinCAT Vision. Job Tasks only need to be created and configured once; thereafter the parallelization takes place automatically in the runtime.

Benefit of Job Tasks

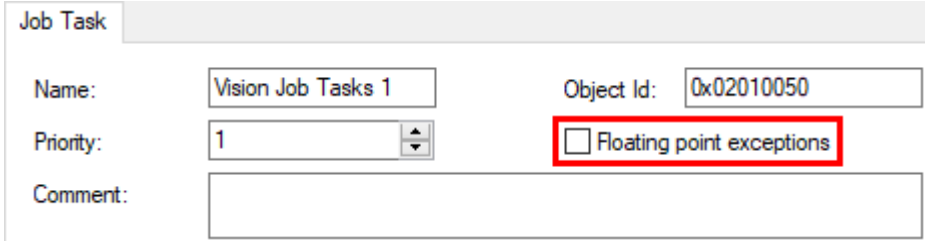
The actual benefit of Job Tasks must be checked in the individual case. This is dependent on the system in use, its configuration, the API functions in use, the image format, the image content and the number of Job Tasks.

Use

Like other tasks, Job Tasks can be created via the **Solution Explorer > SYSTEM >**, right-click **Tasks > Add new item...**. In the following **Insert Task Dialog**, select the type **TwinCAT Job Task (Worker Task)**.



A corresponding core assignment can then be made via **Solution Explorer > SYSTEM > Real-Time**, like for all other tasks. Observe the notes under Restrictions and recommendations [▶ 65].



NOTICE

Floating point exceptions

This setting specifies whether or not TwinCAT checks for floating point exceptions. The same option should be selected here as in the Configuration of the corresponding PLC tasks [▶ 61]. Otherwise exceptions and system crashes can occur.

The assignment of which job tasks are used by TwinCAT Vision is made via the parameter list of the Vision Job Pool [▶ 131]. The Vision Job Pool can be found in the Solution Explorer below the Vision node, as soon as an application has been created there.



Prioritization

Prioritization of the Vision Job Pool access
If several programs that are executed in parallel share the Vision Job Pool, the program with the highest task priority is generally granted access to the Job Tasks. However, this assignment is dynamic, i.e. if the program with the highest priority is currently in a sequential processing section, lower-priority programs can use the Job Tasks. As soon as a higher-priority program can process tasks in parallel, the tasks of lower-priority programs are displaced. However, the master task of the lower-priority program is able to terminate a task started in parallel on its own, so that there is no need to wait for a new availability of Job Tasks.

Multiple tasks on one core
As indicated in the section on restrictions and recommendations, a Job Task should not share a core. Otherwise, the task priorities apply.

Restrictions

- Job Tasks can currently only be used for TwinCAT Vision.
- More than one Job Task may not be assigned to the same core.
- A Job Task should not be assigned to the same core as a Vision PLC task.
- A Job Task should not be assigned to the same core as the PlcAuxTask.

Recommendations

- Each Job Task should be assigned to a dedicated isolated core.
- However, if another cyclic task runs on the same core as a Job Task, the priority of the cyclic task should be higher than that of the Job Task. It should be noted that this restricts parallelization and, in the worst case, make it unavailable.
- No job task should be assigned to the default core.

If cyclic TwinCAT tasks or the Windows operating system claim computing time on the core at the same time, an analysis is hampered due to the varying load of the Job Task.

5.2 VISION node

Components that are relevant for image processing are configured via the VISION node and its sub-elements. It has tabs for Service Configuration [▶ 66] and Logging [▶ 68].

Displaying the VISION node

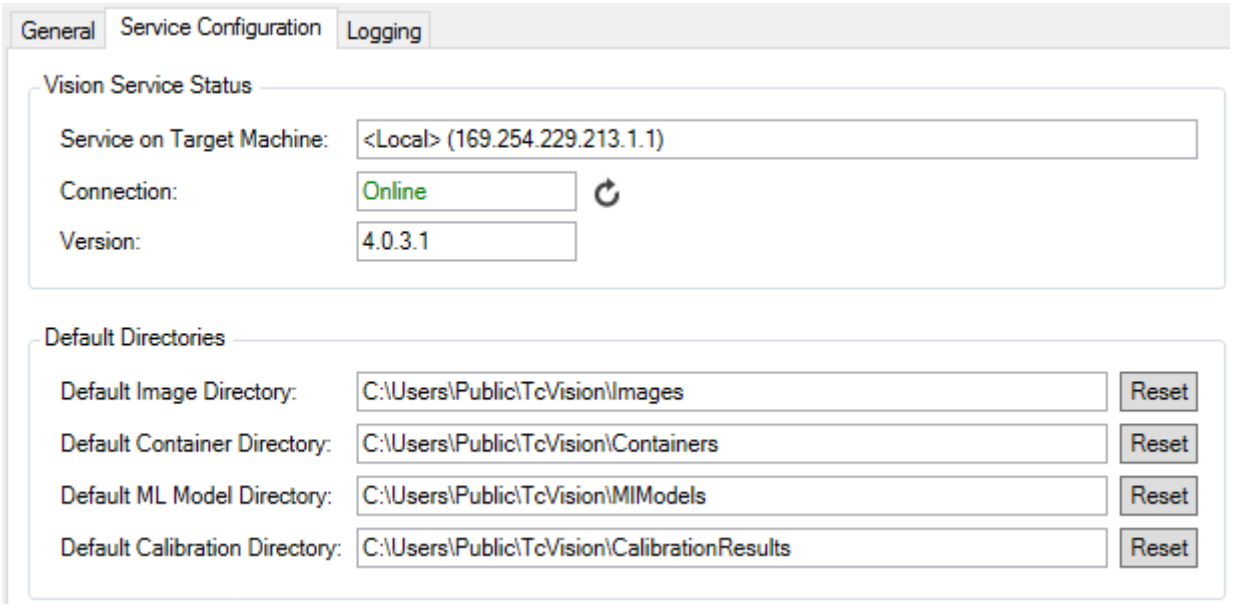
If the VISION node is not displayed in your TwinCAT project, right-click on your TwinCAT project and select **Show Hidden Configurations > Show VISION Configuration** to show the Vision node. The next step is to create an Application node [▶ 68].

5.2.1 Service Configuration

The Vision Service offers a non-real-time counterpart for some real-time components (e.g. function blocks). For this purpose the Vision Service implements an ADS server, which enables access from both the real-time and development environments.

The Vision Service enables the TwinCAT Vision API function blocks to indirectly access the hard disk from the real-time environment, e.g. to load or save images.

The Vision Service is configured under **VISION > Service Configuration**.



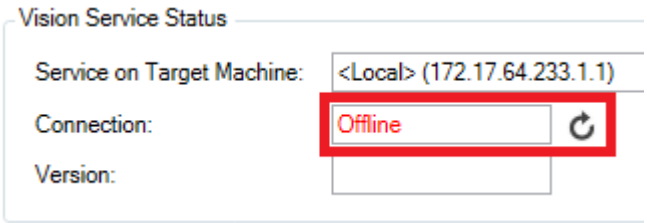
Vision Service status

Service on Target Machine	Target system on which the TwinCAT Vision Service is running.
Connection	Status of the connection between the development environment and Service. This is a good indicator of whether the Service is running properly.
Version	Version number of the TwinCAT Vision Service.

Before it is used for the first time, the Vision Service must be installed on the target system, see Setup instructions [▶ 17].

Manually starting the Vision Service

The Vision Service is normally started automatically together with TwinCAT. However, if the Vision Service is offline, it can be reactivated with a TwinCAT restart (click  or ). If this does not work, or if the Service repeatedly fails to start, it should be re-registered with TwinCAT; see: Vision Service doesn't start [▶ 2864].



Default Directories

These paths define the point on the target system at which TwinCAT Vision, by default, should search for files and save them.

Default Image Directory	This is the default path used by the function blocks to read and write images: FB_VN_ReadImage [▶ 1598] FB_VN_WriteImage [▶ 1605]
Default Container Directory	This is the default path used by the function blocks to read and write containers: FB_VN_ReadContainer [▶ 1596] FB_VN_WriteContainer [▶ 1603]
Default ML-Model Directory	This is the default path used by the function blocks to read and write ML models: FB_VN_ReadNeuralNetwork [▶ 1609] FB_VN_ReadMlModel [▶ 1600] FB_VN_WriteMlModel [▶ 1607]
Default Calibration Directory	This path is used by default by the function blocks for reading and writing calibration results: FB_VN_ReadCalibrationResult [▶ 1594] FB_VN_WriteCalibrationResult [▶ 1601]

A complete list of all important paths can be found in the section Important paths [▶ 2860].

Vision Service Log

All important service events are written to the log file TcVnService.log on the target system, which can be found under *C:\ProgramData\Beckhoff\TcVnService*. In case of support questions regarding the TwinCAT Vision Service, the log file and the time at which the behavior occurred must be sent as well. The development system also features a logger [▶ 68] for all components under the Vision node.

Vision Service Cache settings

By default, the Vision Service caches the images it reads in order to provide the requested images more quickly the next time they are retrieved. This optimized mode of operation covers the requirements of most applications and is therefore recommended.

In case the image content changes but the file name remains the same, image caching can be disabled. To do this, open the Vision Service settings at *C:\ProgramData\Beckhoff\TcVnService\settings.json* on the target system, change the default value of the updateCacheOnReadSize parameter from 0 to 1 and save the file.

Exit Vision Service

To change the settings, the TwinCAT Vision Service (TcVnService.exe) must first be terminated via the task manager, otherwise the changes will not be applied. After that, the service can be restarted by restarting in Config Mode.

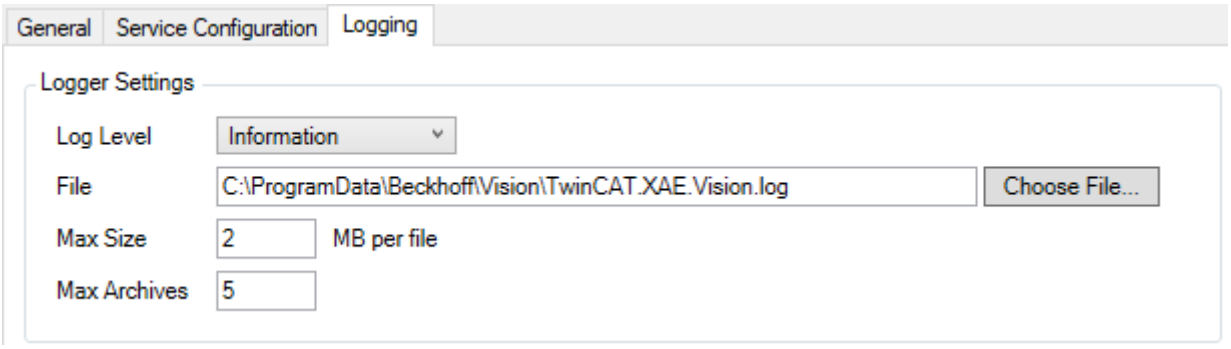
NOTICE

Inconsistent data

Ensure that the same file is not accessed simultaneously by multiple processes, which can lead to file corruption or inconsistent data.

5.2.2 Logging

The Vision node has a logger which, depending on the Log Level, logs all important events of the TwinCAT Vision components underneath the VISION node. The settings can be found under **VISION > Logging**.



The following settings are available:

Log Level

Level	Description
Error	Only error messages are logged.
Warning	Error messages and warnings are logged.
Information	Error messages, warnings and important events are logged. [default]
Verbose	Error messages, warnings and all important events are logged.

Log file

By default, the log file with the name *TwinCAT.XAE.Vision.log* is saved in the directory *C:\ProgramData\Beckhoff\Vision*. This can be changed in the **Logging** tab. The same applies to the maximum size of a log file and the maximum number of files. If the limits are reached, old data is overwritten.

In case of support enquiries relating to behavior within the Vision node, please include the log file and the time at which the behavior occurred.

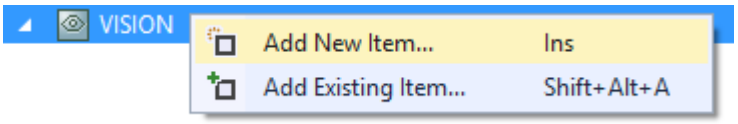
Other loggers

The Vision Service [▶ 66] also has a logger on the target system. In addition, there is a log file for the TwinCAT Vision installation. All relevant logging paths are listed in the section Important paths [▶ 2860].

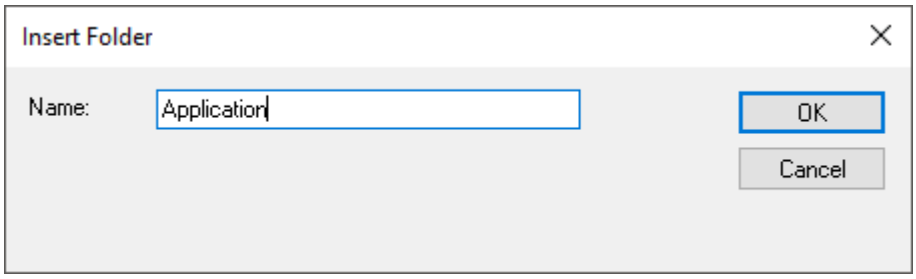
5.3 Application node

A TwinCAT Vision configuration can contain several application nodes, all of which are located inside the VISION node. An Application node is created as follows:

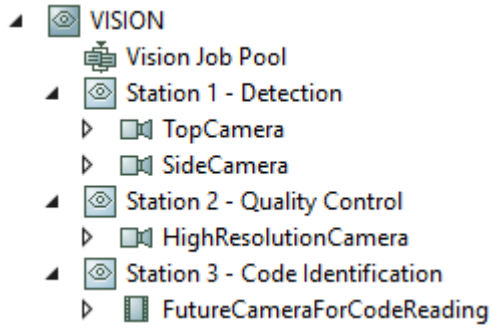
1. Right-click on **VISION node** → **Add New Item…**



2. Enter a name for the Application node and click **OK**.



GigE Vision Camera nodes [▶ 69] and File Source nodes [▶ 125] are created under the Application node. The purpose of Application nodes is consolidation of thematically related Vision devices. The following figure shows an example structure with several Vision applications:



In addition, application nodes enable the common export/import [▶ 135] of several Vision device configurations in a single XTI file.

5.4 GigE Vision Camera objects

Each GigE Vision Camera object represents a camera within a project. A project can contain several camera objects.

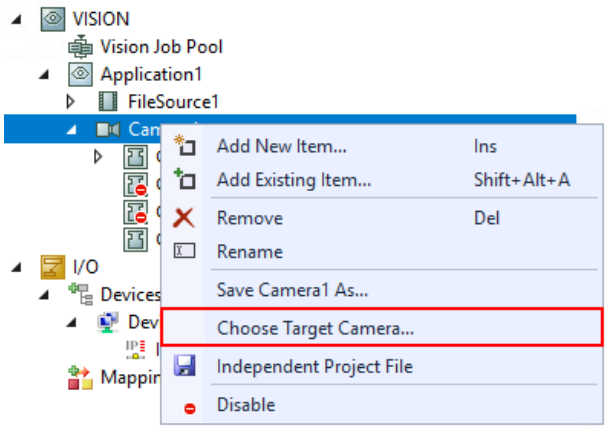
Creating a GigE Vision Camera object

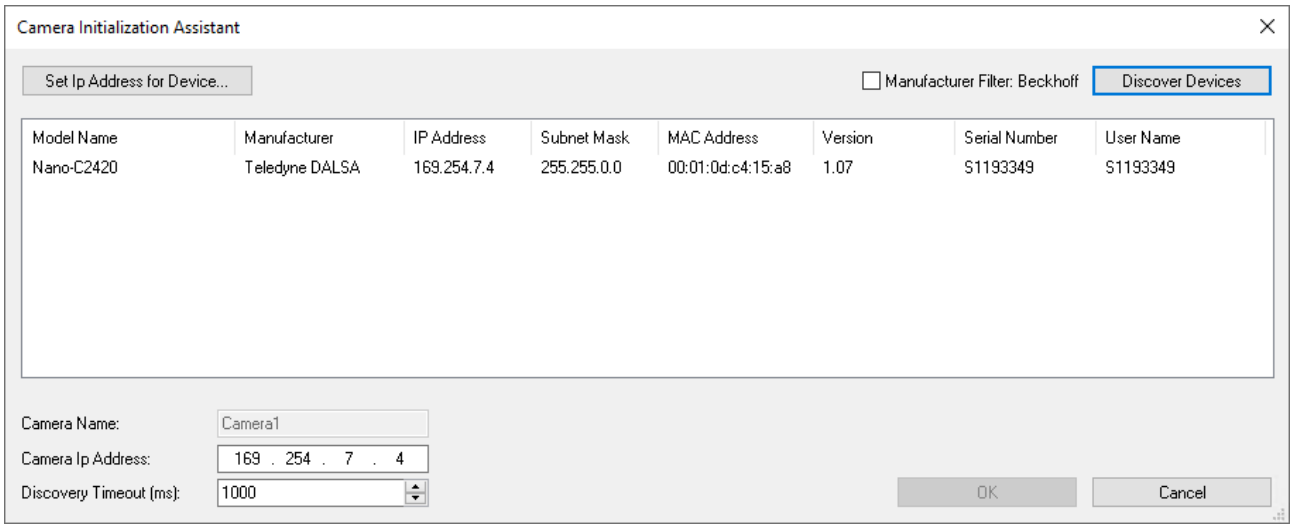
Creating a camera is described in First Steps [▶ 27] under Creating a GigE Vision camera [▶ 38]. Refer here to the notes on the configuration of network adapters [▶ 56] and device tasks [▶ 61].

After creating a camera, the system automatically tries to read the GenAPI file from the camera. This file contains the complete description of the parameter tree of the camera (see GigE Vision Standard [▶ 122]). On the basis of this description, the configuration tree is then made available in the Configuration Assistant [▶ 75] and the current parameter values are read by the camera.

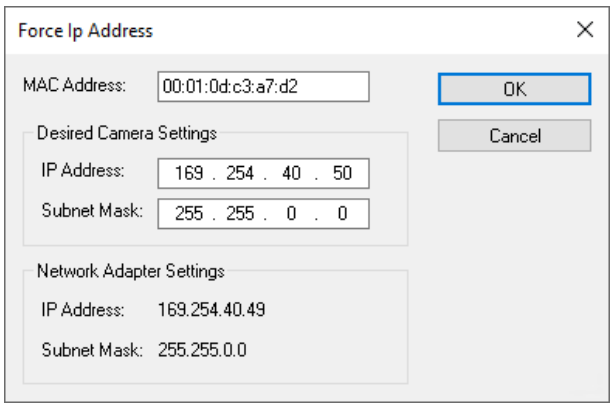
Selecting a connected camera

To link a camera connected to the target system with a GigE Vision Camera object, the **Camera Initialization Assistant** can be opened by right-clicking the corresponding camera object and selecting **Choose Target Camera**. Alternatively, this dialog can also be opened via the Connection Options on the General [▶ 71] tab.





Camera Name	Display of the name of the selected camera object.
Camera Ip Address	IP address that is saved in the TcCOM parameters. The entry will be overwritten on selecting a camera and also updated in the module with **OK**.
Discovery Timeout	The Discovery Timeout describes the time allowed for the response from the camera that was connected to the respective network port. Unit: milliseconds (ms) Default: 1000
Set Ip Address for Device	Opens the following Force Ip Address dialog in order to set the IP address and subnet mask of a camera. The specified IP address and subnet mask are saved in the TcCOM parameter `CameraIPAddress` in the Image Acquisition Object [▶ 99].
Manufacturer Filter: Beckhoff	Enables a filter so that only cameras from the manufacturer Beckhoff are displayed in the results table.
Discover Devices	The search procedure is started with **Discover Devices**.
Results table of the camera connections found	List of the cameras found and display of the following camera information read out: model name, manufacturer, IP address, subnet mask, MAC address, version, serial number and user identification



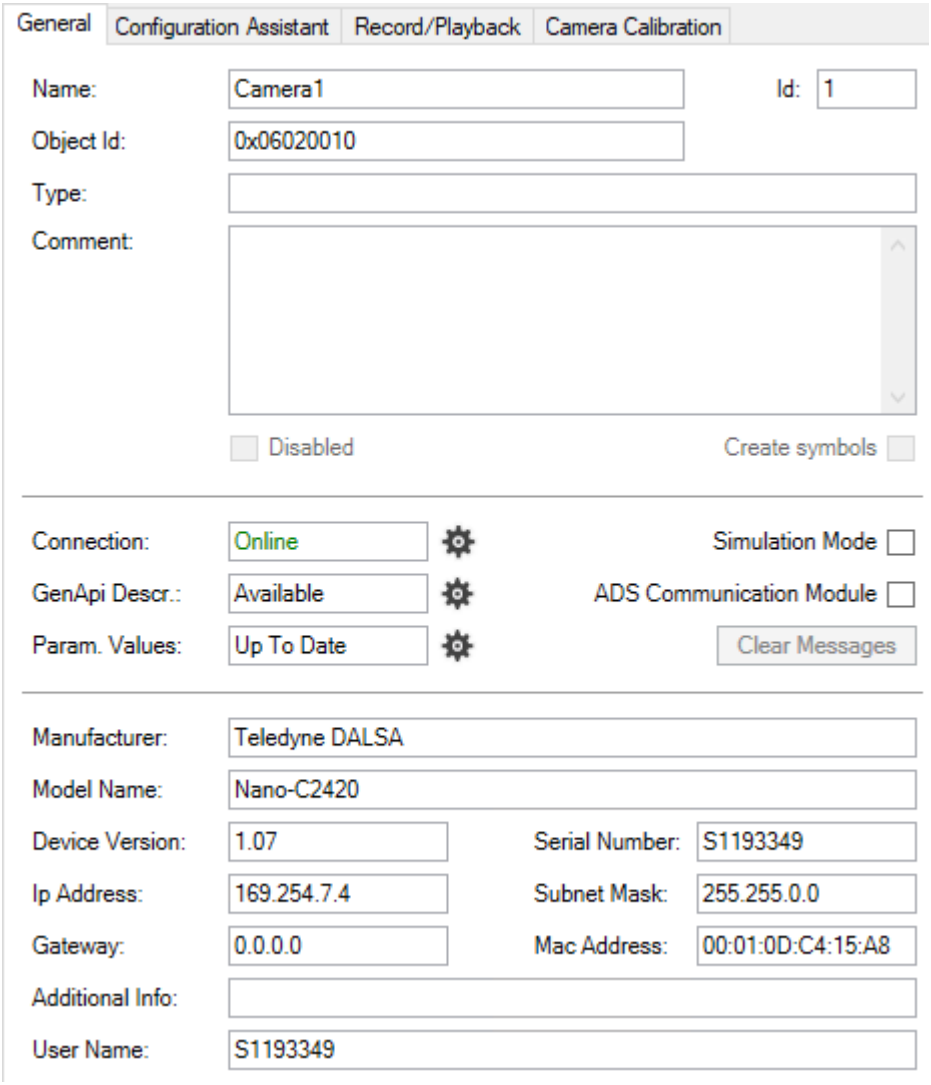
MAC Address	MAC address of the selected camera.
Desired Camera Settings	Input of the desired IP address and subnet mask to which the camera is to be set.
Network Adapter Settings	Current IP address and subnet mask of the network adapter with which the camera is connected.

Exporting/importing GigE Vision Camera objects

If you wish to save the configuration of a camera or import it into a project, follow the instructions in the chapter Exporting/importing configurations [▶ 135].

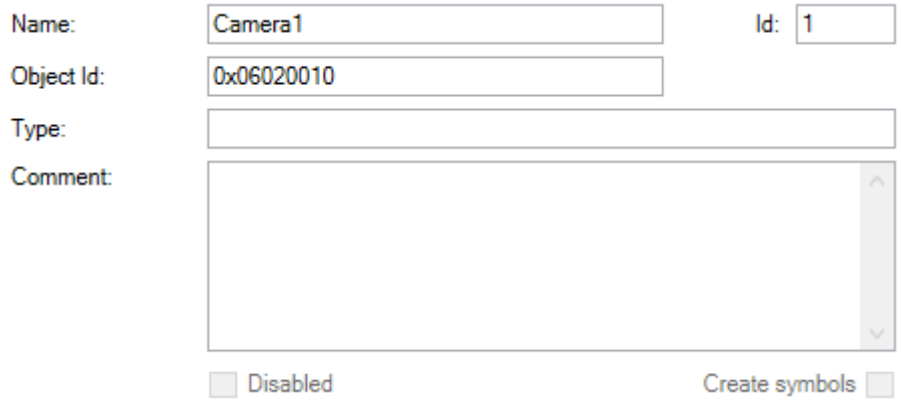
5.4.1 General

The General tab provides an overview of the camera object in TwinCAT and the connected camera.



General object information

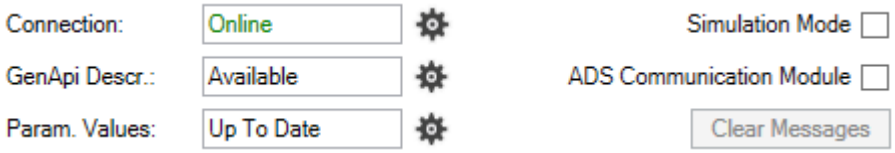
The upper part contains general information about the camera object:



Name	Selected name of the object. **Do not assign an object name more than once!**
Object Id	32-bit identification number that is automatically assigned and unique within a project.
Type	Object type.
Comment	Freely editable field for user notes.
Id	Identification number of the camera modules, which is assigned consecutively. If a camera module is deleted, the number becomes free again and can be assigned to a new camera module.

Status of the camera instance

The middle part contains status information for the camera instance:



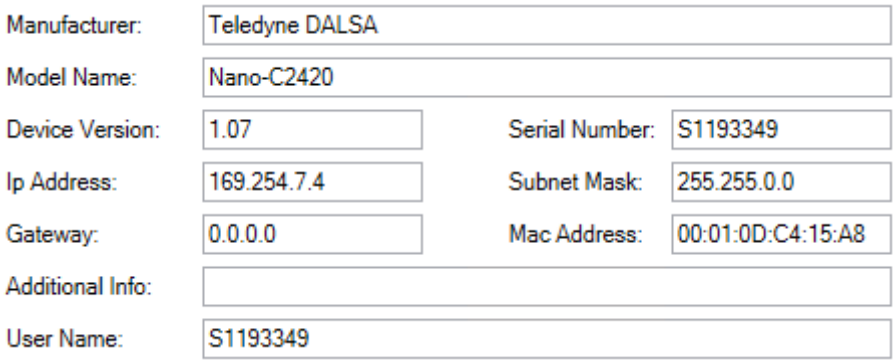
The following table explains the individual elements:

Connection	Displays the connection status to the camera. Via the button  the display can be updated with **Refresh Camera Status**. In addition, it is possible to open the dialog for selecting a camera with **Choose Target Camera** Possible status: • **Unknown**: The connection to the camera cannot be checked. The camera module may not be in the operating state or creation of the camera module may have been incomplete. • **Online**: The connection to the camera is established. • **Offline**: There is no connection to the camera. • **Disabled**: The camera node is disabled. There is no communication with the camera. • **Not in Config Mode**: The connection status can only be displayed here in TwinCAT Config Mode. • **Loading**: The camera module is initialized.
GenApi Descr.	This indicates whether the GenApi structure (feature tree in the Configuration Assistant [▶ 75]) was successfully created. Status options: • **Available**: GenApi structure was successfully created. • **Not Available**: GenApi structure was not created. No features can be displayed. The button  also allows the following actions: • **Read GenApi Description from Camera**: The GenApi description of the camera is read. At the same time, all features in the configuration tree [▶ 78] are reset. • **Export GenApi Description**: The GenApi description of the camera is saved locally.
Param. Values	Status options: • **Up To Date**: All features of the camera have values and are up-to-date (camera online). • **Complete**: All features of the camera have values, but these may be outdated (camera offline). • **Incomplete**: Not all features of the camera have values. The button  also allows the following actions: • **Write Local Changes:** All local parameter values that deviate from the camera values (marked in orange in the configuration tree [▶ 78]), are written to the camera. • **Revert Local Changes**: Any local changes in the feature tree that have not yet been written to the camera will be reset.

Simulation Mode	In simulation mode, images from recorded camera streams can be sent to the PLC instead of a real camera supplying images (see Record/Playback [▶ 101]). If the simulation mode is activated, the Image Acquisition Object is automatically deactivated and the Image Acquisition Simulation Object is activated. If the simulation mode is deactivated, the Image Acquisition Object is automatically activated and the Image Acquisition Simulation Object is deactivated. The corresponding object links (see Camera details [▶ 123]) will each be adjusted automatically. Default: disabled.
ADS Communication Module	Required for the recording of a camera stream, see Record/Playback [▶ 101]. This checkbox merely activates or deactivates the Ads Communicator object of the Camera object. **Please note that if Ads Communicator object is activated, images captured by the camera will still be internally referenced for a short while so that they cannot be transformed into displayable images.** For further details see Displayable images [▶ 151]. Default: disabled.
Clear Messages	Use this button to delete the messages written by this camera object. The button is only active if messages are present.

Camera information

In the lower part, general camera information is displayed as far as it could be read from the camera (e.g. manufacturer and model name of the camera). This information can also be found in the configuration tree [▶ 78].

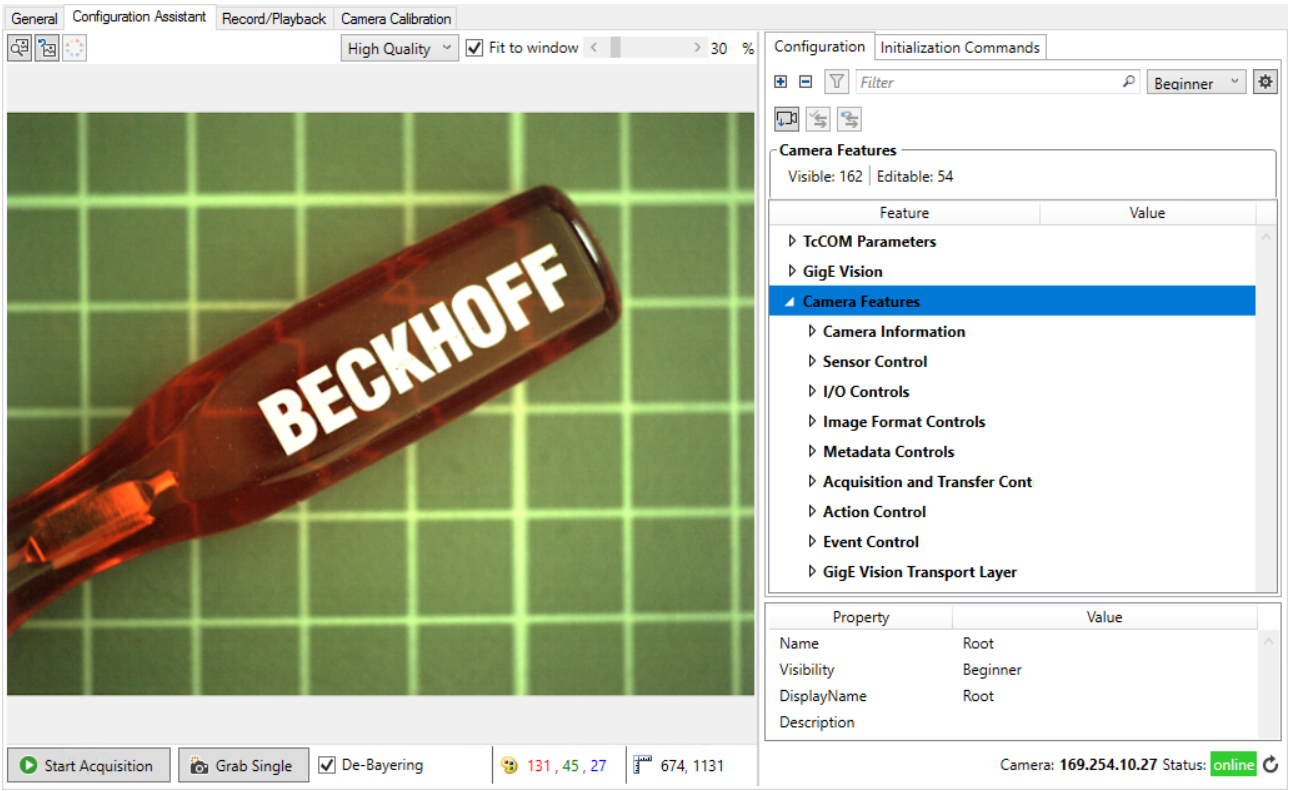


5.4.2 Configuration Assistant

Both the camera parameters and the communication behavior can be configured on the **Configuration Assistant** tab. The Configuration Assistant has four sections:

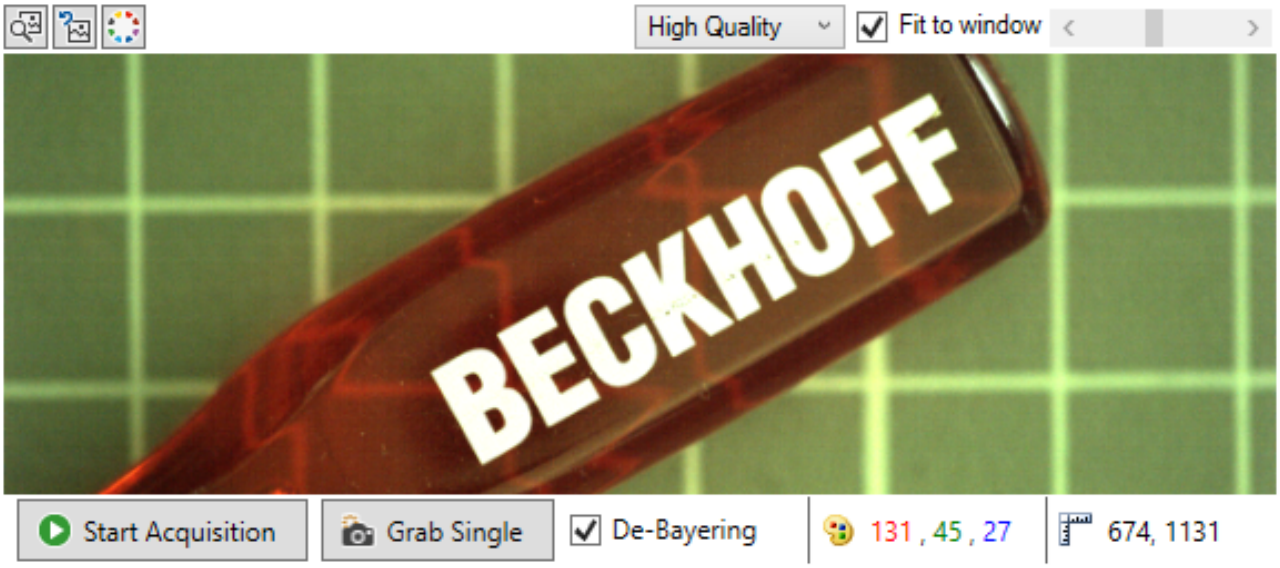
- Image preview [▶ 76]
- Configuration tree [▶ 78]

- Initialization Commands [▶ 87]
- Actions & settings [▶ 84]
- TcCOM parameters [▶ 99]



5.4.2.1 Image preview

The area on the left in the Configuration Assistant features a preview window and some operating elements.



In this image preview, only 8-bit integer images are correctly displayed.

Lower bar

Below the preview window are the controls for starting / stopping the preview and information about the cursor position on the image:

Start/Stop Acquisition	Button for starting or stopping image acquisition.
Grab Single	Button for triggering a single image.
De-Bayering	Images in Bayer format can alternatively be previewed as RGB images. **If this option is activated, the next image from the camera will be converted, not the present one.**
Pixel Value	The color or intensity value of the pixel at the mouse cursor position on the image is indicated here.
Cursor Position	Shows the mouse cursor position on the image. Please note that the zero point for images is always at the top left.

Image acquisition and trigger settings

Make sure that the image acquisition and trigger settings of the camera are selected in such a way that the camera reacts to the commands.
If, for example, the Trigger mode of the camera is active, the camera will not automatically send images at the start of the image acquisition. See the example Image acquisition and trigger [▶ 2849] for more details.

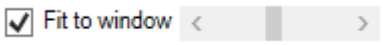
Upper bar

Above the preview image there are some controls for facilitating the camera configuration and adjustment of the image display.

Camera configuration

 **Set ROI**	After clicking this button, an ROI can be drawn directly in the preview image by holding the left mouse button down. This ROI is written to the camera parameters, so that the next image is sent with the set ROI.
 **Reset ROI**	After clicking this button a previously set ROI will be deleted, i.e. image size and offset will be set to the default values of the camera.
 **Whitebalance**	Clicking will perform a white balance on color cameras. The camera must support the following GenAPI parameters for this: *BalanceRatioSelector* and *BalanceRatio* (or *BalanceRatioAbs* for older cameras). In addition, the camera image must contain a white (not overexposed or reflective) object, such as a sheet of paper, for white balance. Following a white balance, the necessary values are saved in the named GenAPI parameter and can be adopted into the Initialization Commands [▶ 87] or UserSets. Automatically persistent storage on the camera may be specific to the camera concerned, but is not a standard feature. This button is only active if there is a valid color format (3-channel images as RGB or 1-channel images in the Bayer format).

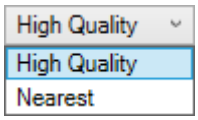
Image size



Adjustment options for the image display size. The image can be scaled between 20% and 6400% of the original size. Click **Fit to window** to automatically set the display size such that the preview makes the best use of the available space on the screen.

Interpolation

Here you can choose between different interpolation methods to scale the image from the original size to the display size:



High Quality	High-quality interpolation with smooth edge and pixel transitions.
Nearest	Interpolation by adopting the locally nearest pixel value, whereby the individual pixels are displayed visibly and delimited. This method can be used to check pixel values.

Right-click the image

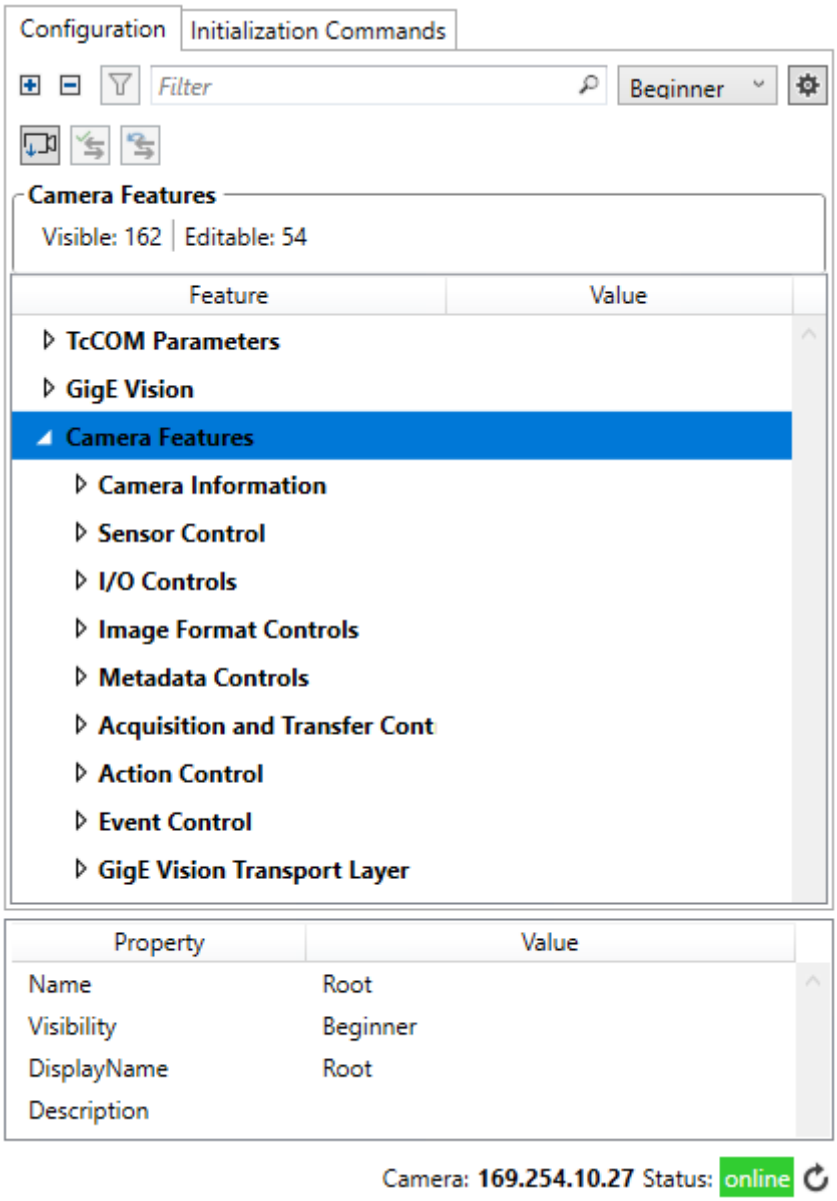


Start Acquisition	Start image acquisition.
Stop Acquisition	Stop image acquisition.
Save Current Image	Save image as bitmap, JPEG or PNG.

5.4.2.2 Configuration

The Configuration tab consists of 3 areas, which are described individually below:

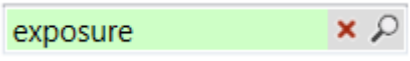
- Filter and controls
- Configuration tree
- Properties window



Filter elements

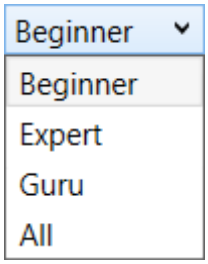


With the help of filter elements, features can be found quickly and easily in the configuration tree and in the Initialization Commands list:

- Click on the  symbol to expand all tree elements. All parameters are visible.
- Click on the  -symbol to collapse all tree elements. Only the main groups are visible.
- Clicking the  icon will filter the tree items and display only the lines with a change. The button is only enabled when changes are displayed.
- The search line 

filters the parameters by specific character strings or (hexadecimal) addresses. When entering a string, the background of the text box is white and an additional button for deleting or resetting the filter appears. If an entry has been confirmed by pressing Enter or clicking the filter button, the text box is also highlighted in green. This indicates whether the current string corresponds to the active filter and therefore the view of the tree elements.

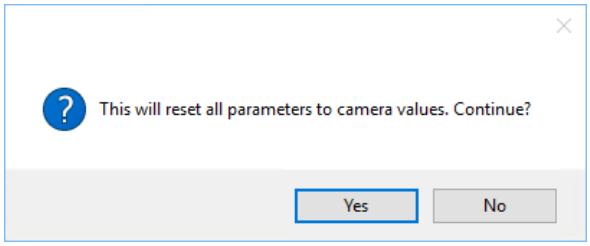
- In accordance with the GenICam standard, the four visibility levels **Beginner**, **Expert, Guru** and **All** can be selected. The visibility level of each feature is stored in the GenApi file of the camera.



Controls



The functions of the buttons from left to right are:

Read Camera Parameters	All parameter values are read from the camera and written to the configuration tree. This must first be confirmed by the following dialog: 
Write Local Changes	All local parameter values that deviate from the camera values (marked in orange in the configuration tree), are written to the camera.
Revert Local Changes	Any local changes in the feature tree that have not yet been written to the camera will be reset.

Configuration tree

The configuration tree contains three main groups:

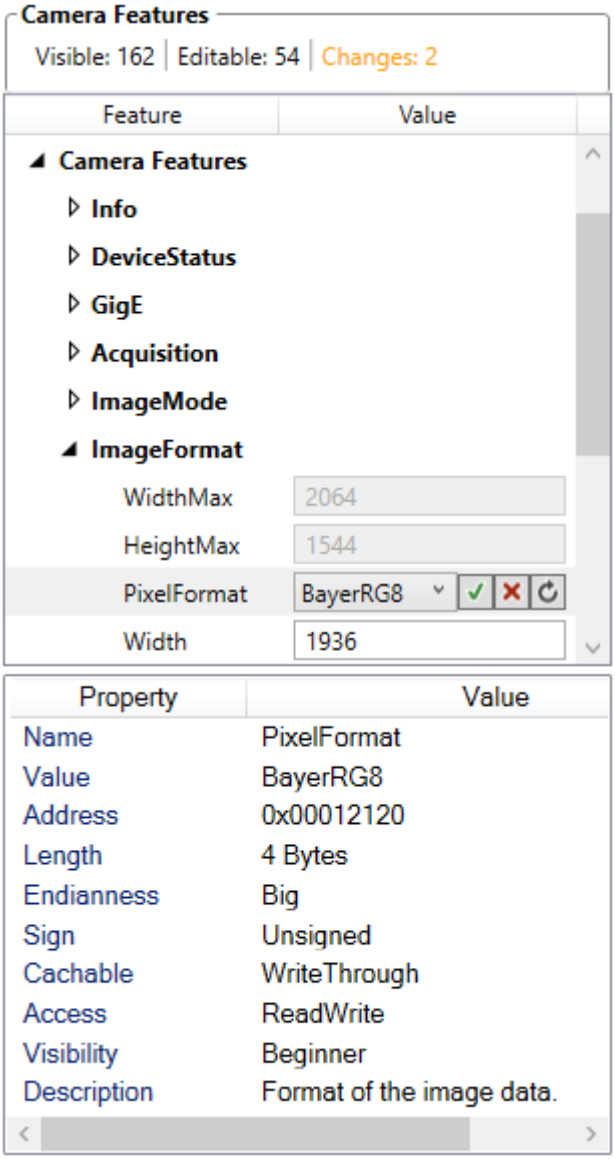


- **TcCOM Parameters**: Parameterization of the communication behavior between TwinCAT and the connected camera. An explanation of these parameters can be found in chapter TcCOM parameters [▶ 99].
- **GigE Vision**: Information about the camera according to the GigE Vision standard, and IP configuration option. Depending on the camera, the settings may also be available in the Camera Features tree.
- **Camera Features**: Camera parameterization based on the GenICam standard. The description of all camera features is read automatically from the GenAPI file by the camera. A feature can be an executable command, a (changeable) parameter or an organizational element. For some settings (e.g. image acquisition/trigger, binning, exposure time, etc.), see Camera configuration samples [▶ 2845]. For details on the parameters of a particular camera, please refer to the corresponding vendor documentation. In addition, the GenICam standard [▶ 122] describes how the Camera Feature tree is structured.

Camera Features Information

Three pieces of information about the Camera Features are displayed above the configuration table:

- Visible: Number of all currently visible or displayed features, depending on the active filter settings.
- Editable: Number of "Visible" features that can be edited. The number generally depends on the current camera parameterization and therefore varies.
- Changes: Total number of changes within the Camera Features tree. This number is only displayed if there are changes, regardless of the active filter settings.



If you click a feature in the configuration tree, the properties of the feature will be displayed underneath the tree.

Setting of parameters

Changeable features can be changed via a text box, a slider, a combo box or a checkbox. Read-only features are grayed out. The following buttons are available for every changeable configuration element:

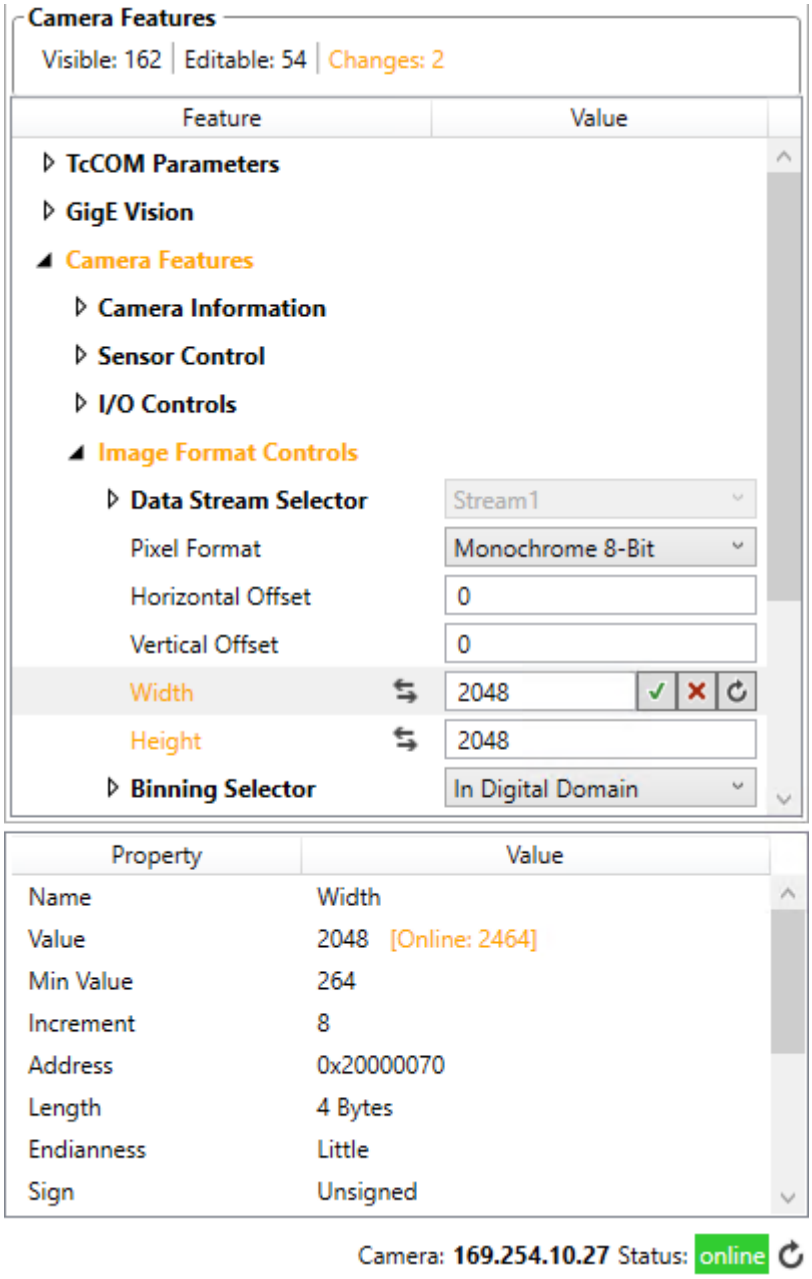
Write Value
Writes the local change to the camera.

Reset Value
Deletes the local change if there is a difference between the local and camera value.

Load Value

Loads the current value of the parameter from the camera.

If the set parameter value deviates from the actual value on the camera, the parameter and higher-level elements in the configuration tree will be marked in orange. In addition, the lines with changes are marked with an icon and the deviation is displayed in the properties window. An orange "Changes" note above the table generally indicates changes in the list.

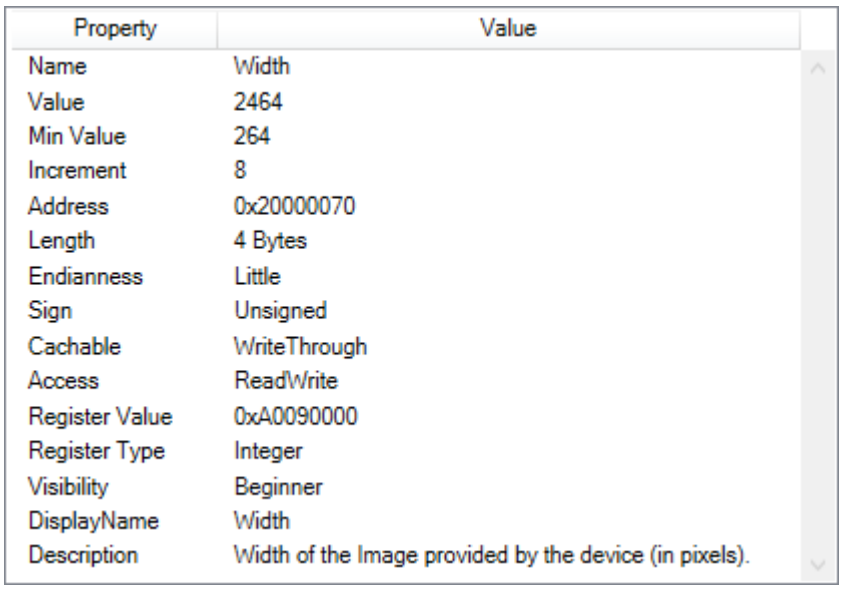


Offline changes

The parameters in the configuration tree are also changeable if the camera is offline or if the target is in Run mode. In this case the parameters are not written directly to the camera, but only saved in the project. As a result of this, it is possible to make offline parameter changes that can be synchronized with the camera at a later time.

Feature properties

The properties of the individual features are displayed underneath the configuration tree. This includes the size and address of the respective camera register, access rights and a description.



Property	Value
Name	Width
Value	2464
Min Value	264
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
Description	Width of the Image provided by the device (in pixels).

Property	Description
Name	Name of the parameter.
Value	Local value of the parameter. If the local value deviates from the camera value, the camera value is additionally displayed in orange.
Address	Hexadecimal address of the parameter on the camera
Length	Size of the parameter in bytes
Endianness	Byte sequence of the parameter. This property must be observed when using the function blocks FB_VN_ReadRegister [▶ 1574] and FB_VN_WriteRegister [▶ 1582].
Sign	• **Signed**: Parameter is signed • **Unsigned**: Parameter is unsigned
Cachable	Specifies whether the parameter value is cached when no camera is connected. • **WriteThrough**: The value is written both to the camera and to the project cache and project memory. • **NoCache**: According to the GenAPI description of the camera, the value of this camera parameter should not be saved locally. Therefore, this parameter may be empty if there is no camera connection.
Access	Access rights for reading and writing this parameter on the camera.
RegisterValue	Register value of the parameter. This can deviate from the displayed value if it is a known type. Example: Name: PixelFormat Value: BayerRG8 RegisterValue: 13
RegisterType	Data type of the register. This type indicates which of the read / write function blocks is to be used.
Visibility	Visibility level of the parameter. Possible values: **Beginner**, **Expert, Guru** and **All** The visibility of parameters can be set with the filter elements [▶ 79].
DisplayName	Name of the feature to be displayed.
Description	Meaning of the parameter

These properties are required, for example, if a parameter is to be read or changed from the PLC. Essentially, attention must be paid to Address, Length, Endianness and RegisterValue. The corresponding function blocks can be found in the API reference under Camera Register Access [▶ 1571].

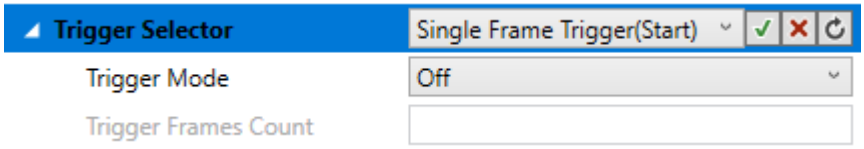
Depending on the data type of a register, further properties such as Min/Max/Increment may be added.

Selectors

Selectors are parameters that are used for the grouping and selection of several sub-features. In order to use a sub-feature, the Selector parameter must first be set to suit (both in the configuration assistant and in the PLC).

If a Selector parameter has no address property, it does not point to a camera register. It then serves merely to provide a clearer display in the configuration assistant. In this case the sub-parameters can be changed directly from the PLC.

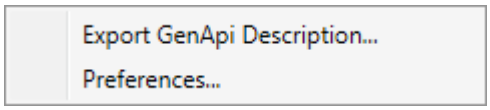
In order to take into account all sub-parameters of a Selector when creating the Initialization Commands, iteration takes place automatically over all sub-features. A corresponding setting can be found in Actions & settings [▶ 84]. An example of a Selector is the Trigger Selector:



5.4.2.3 Actions & settings

Actions

At the top right is the action button  , which can be used for further actions or additional settings:

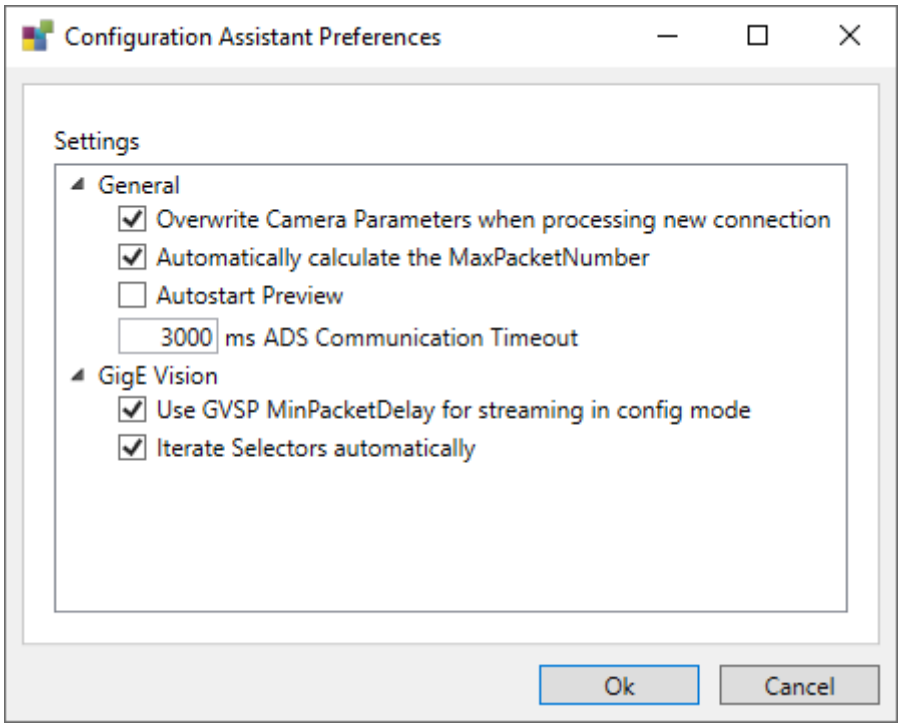


Export GenApi Description...	The camera's GenAPI description is stored locally in a path to be specified.
Preferences...	The settings window described below is opened.

Loading of saved GenAPI file not possible

The loading of a saved GenAPI file is not possible because a connection to the camera is needed in order to read some camera features.

Settings


Configuration Assistant Preferences
Settings
General
Overwrite Camera Parameters when processing new connection
Automatically calculate the MaxPacketNumber
Autostart Preview
3000 ms ADS Communication Timeout
GigE Vision
Use GVSP MinPacketDelay for streaming in config mode
Iterate Selectors automatically
Ok
Cancel

Overwrite Camera Parameters when loading Project	This setting controls whether saved camera values should be written directly to the camera when the project is loaded. This ensures that the camera is automatically in the state in which the project was last saved when it was configured. This only works when the camera objects are initialized for the first time after the project has been loaded, provided that the camera is connected at the time and TwinCAT is in Config mode. Default: activated
Automatically calculate the MaxPacketNumber	If this option is active, the maximum number of required (and allowed) packets for an image is automatically determined after reading out all camera parameters. The calculation is based on image size and pixel format. The calculated value is only written to the responsible GVSP object if it is greater than the current value. This functionality automatically adjusts the reserved space for an image to be received to the actual amount of image data. Default: activated.
Autostart Preview	If this option is active, the live image of the camera is automatically displayed when the Configuration Assistant [▶ 75] is opened, if this is possible. Default: disabled.
ADS Communication Timeout	The Communication Timeout relates to communication within the camera instance in TwinCAT. It only has to be increased if other timeouts in the GVCP or GVSP module also have to be increased. Unit: milliseconds (ms) Default: 3,000 (adapted to the camera)
Use GVSP MinPacketDelay for streaming in config mode	If this option is active, the MinPacketDelay from the GVSP module is used to reduce the bandwidth in Configuration Mode. If you switch this option off, the frame rate of the camera will be reduced instead. Default: activated.
Iterate Selectors automatically	When this option is active, camera enumeration parameters that act as selectors are automatically iterated. The value of a selector parameter is used as an index for addressing other parameters. By changing a selector value, the dependent parameters can be changed accordingly, thus achieving consistent parameterization of the camera. Without this iteration, only the parameters of the currently selected selector would be described. This applies to the settings in the configuration tree [▶ 78] as well as when creating the Initialization Commands [▶ 87]. Default: activated.

NOTICE

Effect on Initialization Commands

If the **Iterate Selectors automatically** option is changed, the Initialization Commands [▶ 87] must then be recreated. Re-create the Initialization Commands by clicking the Create button on the Initialization Commands tab.

5.4.2.4 Initialization Commands

The purpose of the Initialization Commands is to save a specific parameter configuration for a camera and, if required, to set the camera to this configuration. This functionality is necessary because the camera always reflects the delivery state after a restart. The initialization can take place automatically when starting TwinCAT or manually by means of a method call in the PLC.

The initialization commands are comparable with the startup list of EtherCAT devices or with UserSets. The UserSet is a GenAPI feature that allows saving and automatically loading a configuration in the camera. UserSets are not required by the GigE Vision standard and are therefore not natively provided in all cameras.

In order to be able to provide such functionality uniformly for all GigE Vision cameras, TwinCAT Vision offers the possibility with the initialization commands. Of course, UsetSets can also be used as an alternative if available.

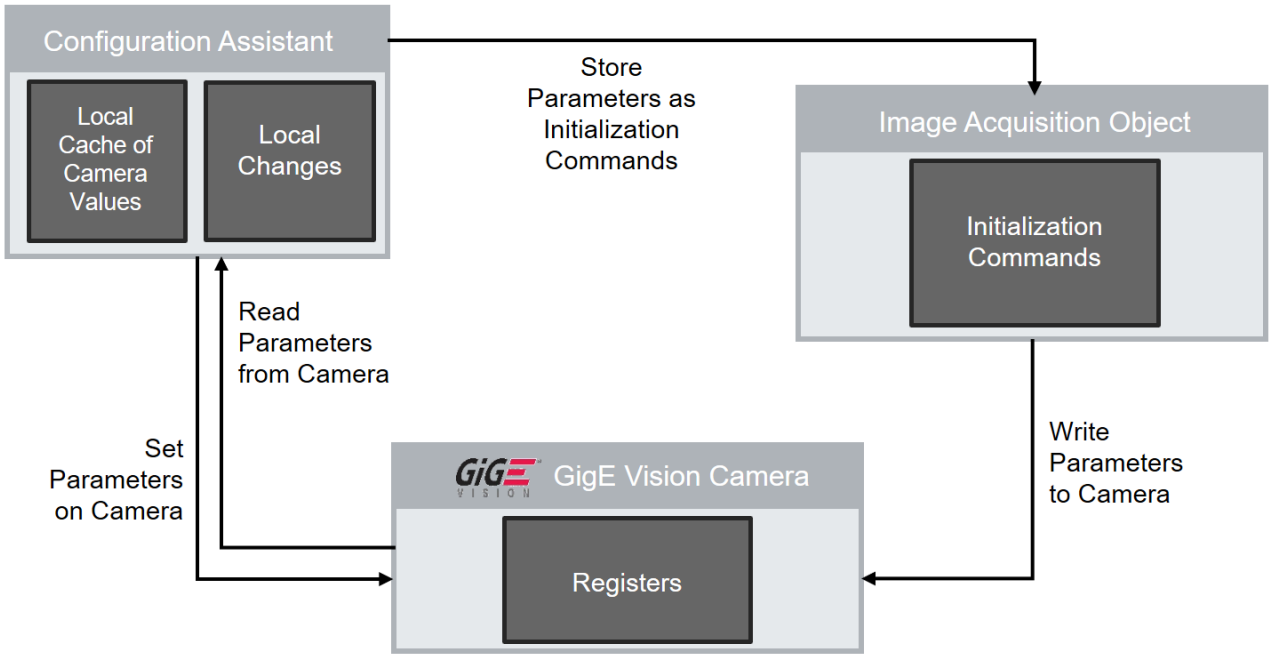
In addition to this general information, the initialization commands can be divided thematically into the following areas:

- Configuration [▶ 88]
- Creation [▶ 93]
- Troubleshooting [▶ 95]

Camera state after a boot process

After a GigE Vision camera has been booted, it contains by default the factory settings in the configuration. In order to restore a previous configuration status, the respective parameters must initially be written to the camera via UserSets, via Initialization Commands or in the background by the camera assistant

The role of the Initialization Commands can be described as follows:



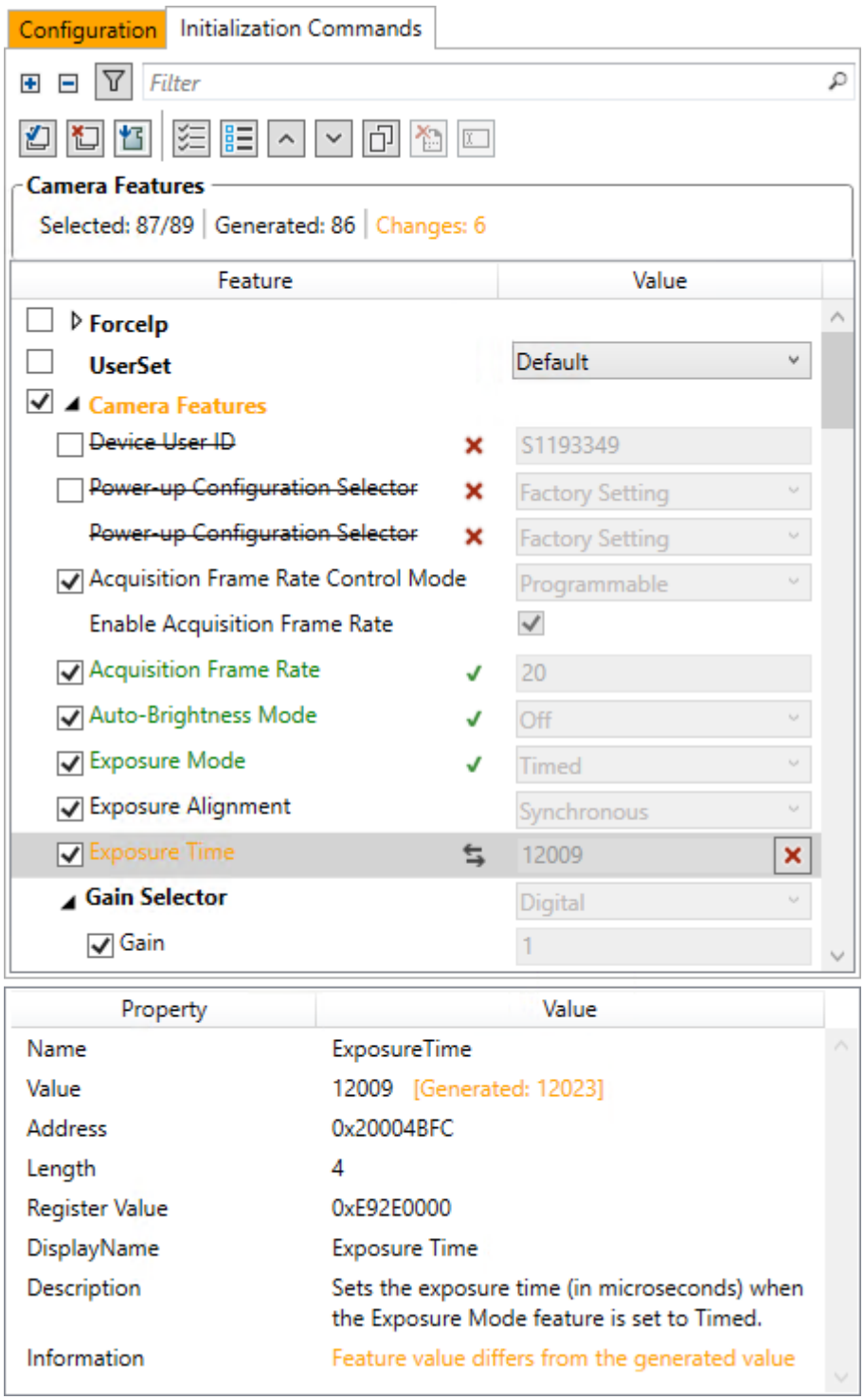
The read and write operations are triggered as follows:

Action	Trigger
Reading parameters from the camera	• Initial reading after setting up the camera • Reading when restarting the TcCOM objects (e.g. when clicking **Load Devices**) • Click on **Read Camera Parameters**
Writing individual parameters to the camera	• Individual changing and writing of the parameters by clicking **Write Value** in the configuration tree [▶ 78]. • Click on **Write Local Changes** • When connecting a camera for the first time after loading a TwinCAT project, if **Overwrite Camera Parameters when loading Project** is activated in the Settings [▶ 84]
Creating Initialization Commands in the camera object from current parameter values	• Click in the Initialization Commands tab on **Create and Test the selected Initialization Commands**
Writing parameters from the initialization commands to the camera	• If the Image Acquisition TcCOM object is switched to the OP state (start of TwinCAT), dependent on the parameter `InitializationAutoMode` in the Image Acquisition [▶ 99] object. • Call via a method of function block FB_VN_GevCameraControl [▶ 1621]

5.4.2.4.1 Configuration

The Initialization Commands tab consists of 3 areas, which are described individually below:

- Filter and controls
- Configuration list
- Properties window



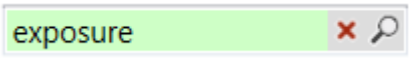
Filter elements



With the help of filter elements, features can be found quickly and easily in the configuration tree and in the Initialization Commands list:

- Click on the  symbol to expand all tree elements. All parameters are visible.
- Click on the  -symbol to collapse all tree elements. Only the main groups are visible.

- Clicking the  icon will filter the tree items and display only the lines with a change. The button is only enabled when changes are displayed.

- The search line 

filters the parameters by specific character strings or (hexadecimal) addresses. When entering a string, the background of the text box is white and an additional button for deleting or resetting the filter appears. If an entry has been confirmed by pressing Enter or clicking the filter button, the text box is also highlighted in green. This indicates whether the current string corresponds to the active filter and therefore the view of the tree elements.

Controls

The left group of buttons  provides the basic functionality. The functions from left to right are:

Button	Description
Create and test the selected Initialization Commands	The Initialization Commands are created with the current values of the configuration list. **The parameters are also written to the camera and read back for testing.**
Delete changes and Initialization Commands	The Initialization Commands created and stored in the TcCOM object are deleted. This also applies to all other changes that have been made to the list. **This resets the entire list to its initial state.**
Apply generated Initialization Commands	The Initialization Commands created and stored in the TcCOM object are written to the camera.

The right group of buttons  provides special functionality that is only active for the entries under the Camera Register node. The functions from left to right are:

Button	Description
Select all features of the CameraRegister category	All entries below the Camera Register category are selected.
Deselect all features of the CameraRegister category	All entries below the Camera Register category are deselected.
Move element up	The selected element is moved up one list position.
Move element down	The selected element is moved down one list position.
Copy element	A copy of the selected element is created. The set value is also copied. **The value of the copied entry cannot be changed later.**
Remove copied element	Removes a previously copied element.
Rename copied element	Opens a window to rename the copied item.

Configuration list

The configuration list contains three main groups:

- **ForceIP**: If this option is active, a ForceIp is performed when initializing the camera in the runtime. The CameraIPAddress from the Image Acquisition Module [▶ 99] is used. Since the command is sent to the respective Mac address of the camera, the Initialization Commands must be regenerated with the changed address when the camera is replaced. The values used are only displayed here. Default: disabled.
- **UserSet**: Some cameras have a UserSet function that can be used to load an image of parameters stored in the camera in order to initialize the camera. If the camera provides this function, the desired UserSet can be selected here. If the option is active, the commands for selecting and loading the corresponding UserSet are written to the Initialization Commands. Default: disabled.

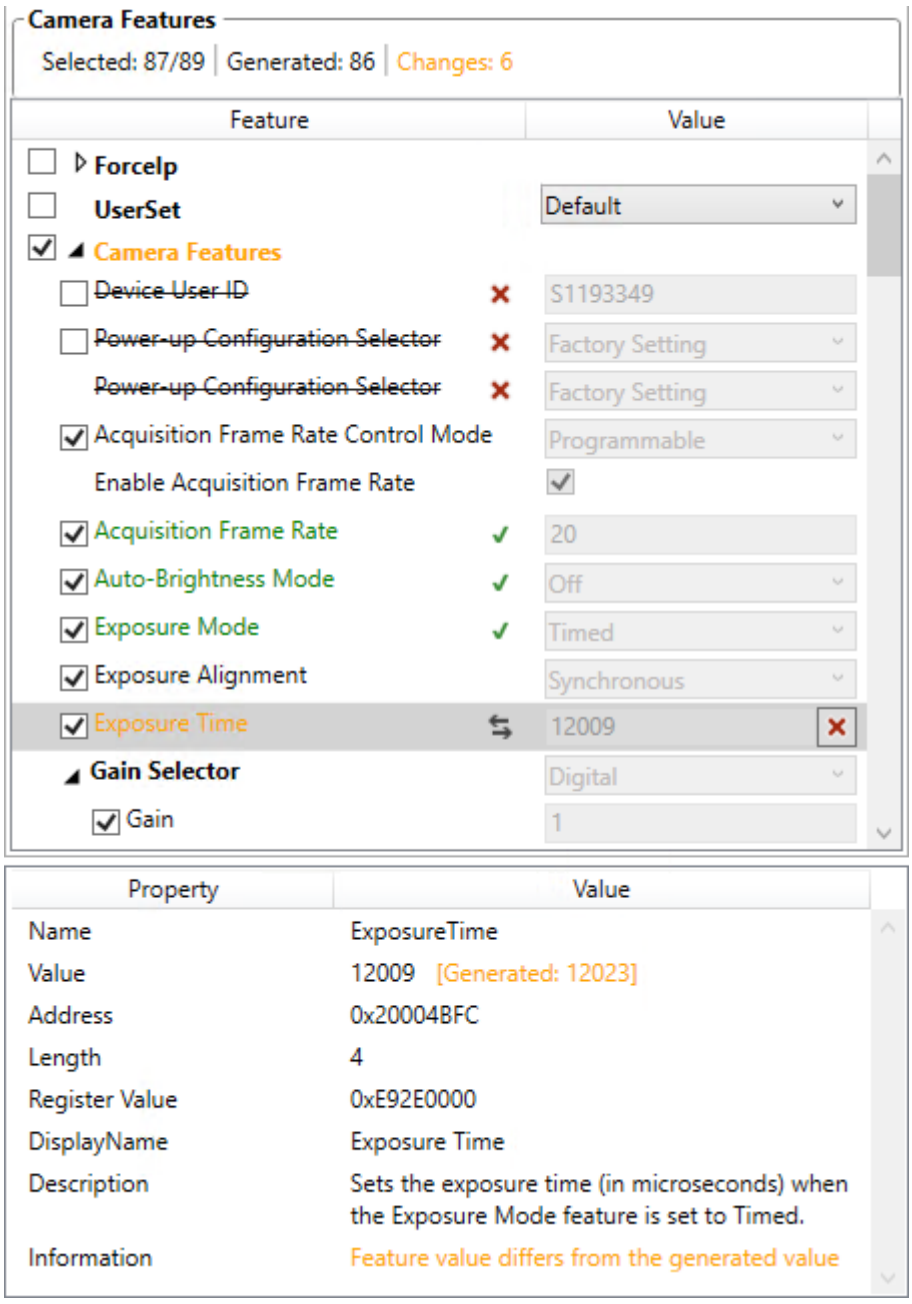
- **Camera Features:** This group lists all registers that are declared as writable in the read GenApi file and are available in the current camera configuration. This applies independently of the stored visibility, therefore also invisible registers are listed, which are displayed in the configuration tree only with the visibility level All. The initial order corresponds to the order in the GenApi file. If the Camera Features option is active, all registers selected below it are used for the creation of the Initialization Commands. It is possible to select and deselect the individual registers and changes are done with the corresponding operating elements. Registers that have the same address and differ only by masking are displayed only with a checkbox, because they cannot be written independently. Changes to the created Initialization Commands are identified in color and graphically. Default: activated. The **Reset Value** button can be used to reset individual value changes to the generated values. The **Apply generated Initialization Commands** button can be used to reset all values to the generated status.

When selecting or deselecting one of the groups or changing a setting below the groups, the Initialization Commands must be recreated. When using UserSets, the Initialization Commands contain the necessary UserSet commands instead of the parameters. The UserSet and CameraRegister options are mutually exclusive.

Camera Features Information

Three pieces of information about the Camera Features are displayed above the configuration list:

- Selected: Number of all currently selected Camera Features from the maximum number of available features. This is generally dependent on the current camera parameterization and therefore varies.
- Generated: Number of the most recently created Initialization Commands stored in the project.
- Changes: Total number of changes within the Camera Features list. This number is only displayed if there are changes, regardless of the active filter settings.

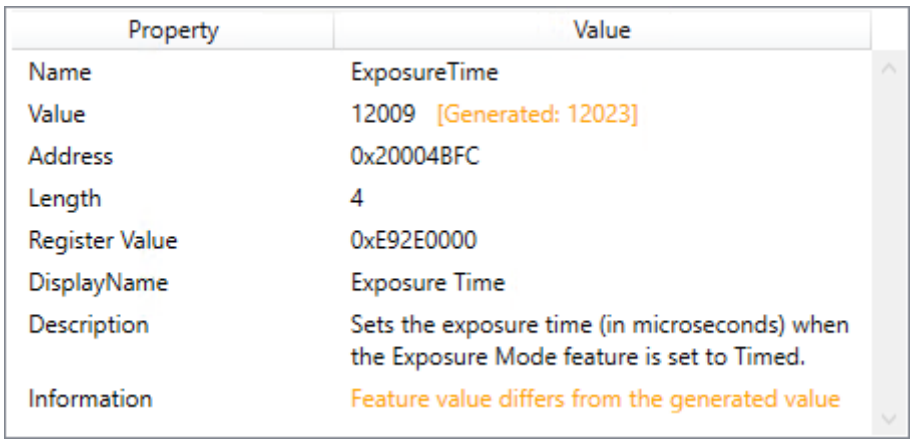


Selector Feature

If **Iterate Selectors automatically** is active in the settings [▶ 84], there is a separate group for each value of a Selector feature. Within this group, all registers that are dependent on the selector value and are writable are displayed. Therefore, the group and register names can occur more than once, in which case the values or register addresses differ. If the setting is not active, there is only one entry per selector.

Properties window

The Properties window displays the following information about the selected feature:



Property	Description
Name	Name of the parameter.
Value	Local value of the parameter. If the local value differs from the Initialization Commands value created, the feature is displayed in orange.
Address	Hexadecimal address of the parameter on the camera.
Length	Size of the parameter in bytes.
RegisterValue	Register value of the parameter.
DisplayName	Feature display name.
Description	Meaning of the parameter.
Information	Information whether the value is different (orange), the feature is added (green) or removed (black, strikethrough).

5.4.2.4.2 Creation

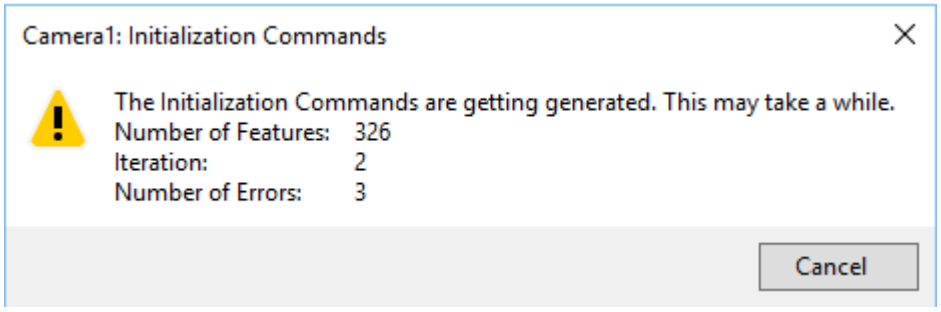
The creation of the Initialization Commands can only be initiated via the button **Create and test the selected Initialization Commands** on the Initialization Commands tab. Before doing so, the settings or changes should be checked to ensure that all required registers are selected or those not required are deselected and that the register values correspond to the desired settings for operation.

Creation steps

The creation of the Initialization Commands comprises several actions that are performed in the background:

- Test writing of the Initialization Commands to the connected camera.
- Test reading of the previously written Initialization Commands from the connected camera.
- If value differences or errors are detected, a list with detailed information appears. If no critical errors have occurred, the user can still accept the error-free Initialization Commands.
- Write the Initialization Commands array to the ImageAcquisition TcCOM object of the camera.

Test writing and subsequent reading enables you to validate whether the camera accepts the initialization commands. One of the following windows is displayed during the generation:



The following status information is displayed:

Number of Features	Number of parameters with which the Initialization Commands were created.
Iteration	Number of previous attempts to write the Initialization Commands to the camera. In case of success, this is "1". In case of failure this value increases, as an attempt is made again to write the Initialization Commands successfully to the camera without the erroneous parameters. Normally no more than two iterations should occur; if they do, however, the concatenation in the TcCOM parameters of the ImageAcquisition must be deactivated.
Number of Errors	Number of errors that have occurred. These are described in more detail in the following window.

If the writing of the Initialization Commands to the camera works very quickly and successfully, the window disappears again immediately and is not visible.

Messages

After the successful creation of the Initialization Commands, the camera - due to the test writing - has exactly the configuration that will be written to the camera during operation later on. If errors occur during the test writing of the Initialization Commands to the camera, these are displayed in the following window:

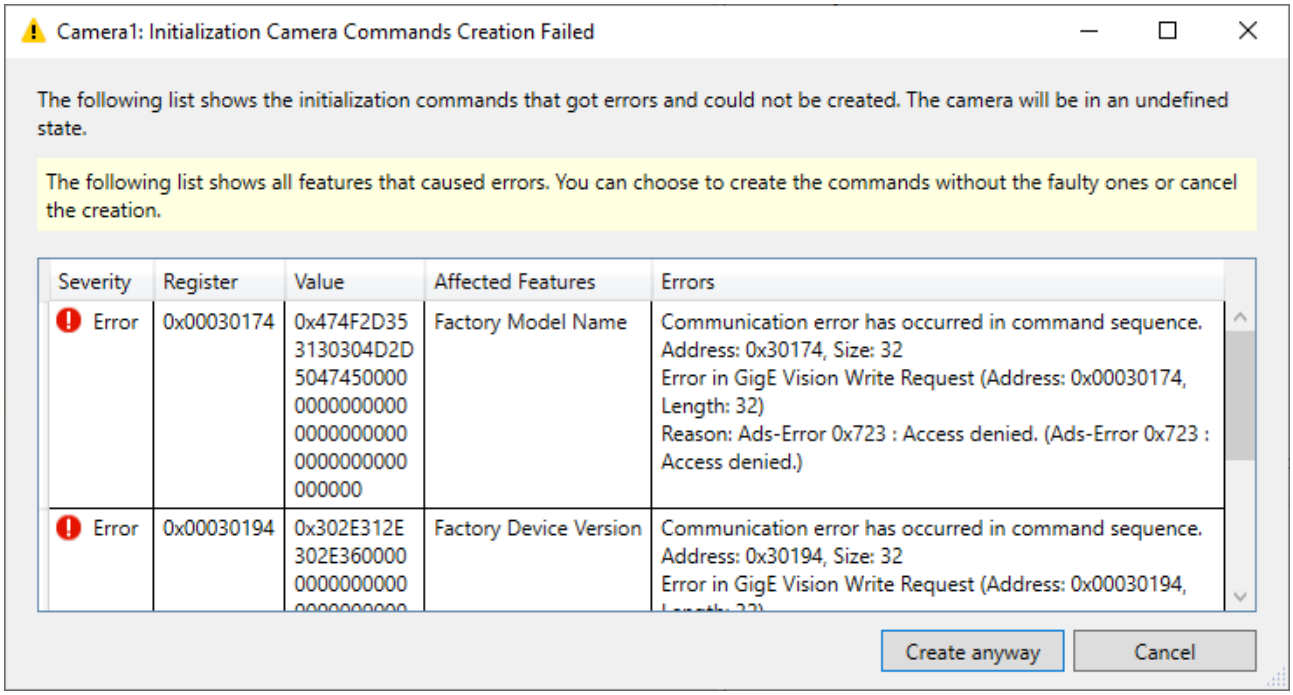


Fig. 1: Uncritical errors when creating Initialization Commands

As long as no FATAL errors have occurred, the Initialization Commands can be created and saved anyway by clicking **Create anyway**. In this case the erroneous parameters will be excluded from the Initialization Commands and accordingly not written to the camera later on.

In the case of a FATAL error the Initialization Commands cannot be saved. Therefore, you cannot click **Create anyway**. First of all the cause of the error must be rectified.

These error cases and their rectification are described below Troubleshooting [▶ 95].

Warnings when creating the Initialization Commands

Warnings can also occur in addition to errors. These are mostly just an indication that the camera has not accepted and corrected a value.

5.4.2.4.3 Troubleshooting

For the basic camera communication and parameterization as well as the creation of the Initialization Commands, three components are mainly relevant:

- the content of the GenApi device description file
- the camera behavior
- and the GevImageAcquisition [▶ 123] object behavior.

GenApi device description file

The GigE Vision [▶ 122] specification includes the GenApi module of the GenICam specification, which describes the functions supported by a camera. This description takes the form of an XML device description file, which must be created in conformity with the specification. The GenApi description provided by a camera is checked upon receipt. If non-compliant digits are included, they are reported as warnings or errors.

- A warning is issued, for example, if undefined registers are located in the reserved bootstrap area. These registers are subsequently ignored, but this does not usually lead to restrictions in operation. In individual cases it has to be checked whether the reported registers are needed.
- An error is reported, for example, if the described access rights do not match the camera behavior. That is, if the camera returns an error when reading registers marked readable or writing registers marked writable.
- If the GenApi description contains completely non-compliant passages, in rare cases the camera may not be able to operate at all.

Camera behavior

The camera behavior can have an impact if, as described earlier, the behavior differs from the GenApi description and if registers contain interdependencies or influence each other.

- Registers can, for example, be locked by other registers, so that these must first be activated or deactivated.
- Register values can be influenced by other registers or formulas which, for example, restrict the maximum value.
- The camera can change register values when other registers are written. For example, the camera may automatically reset the ROI when the pixel format is written.

Refer to the manufacturer's documentation

In case of problems with the camera behavior, compare precisely with the manufacturer's documentation. It is possible, for example, that the writing of individual ReadWrite registers is wrongly rejected by the camera.

GevImageAcquisition object behavior

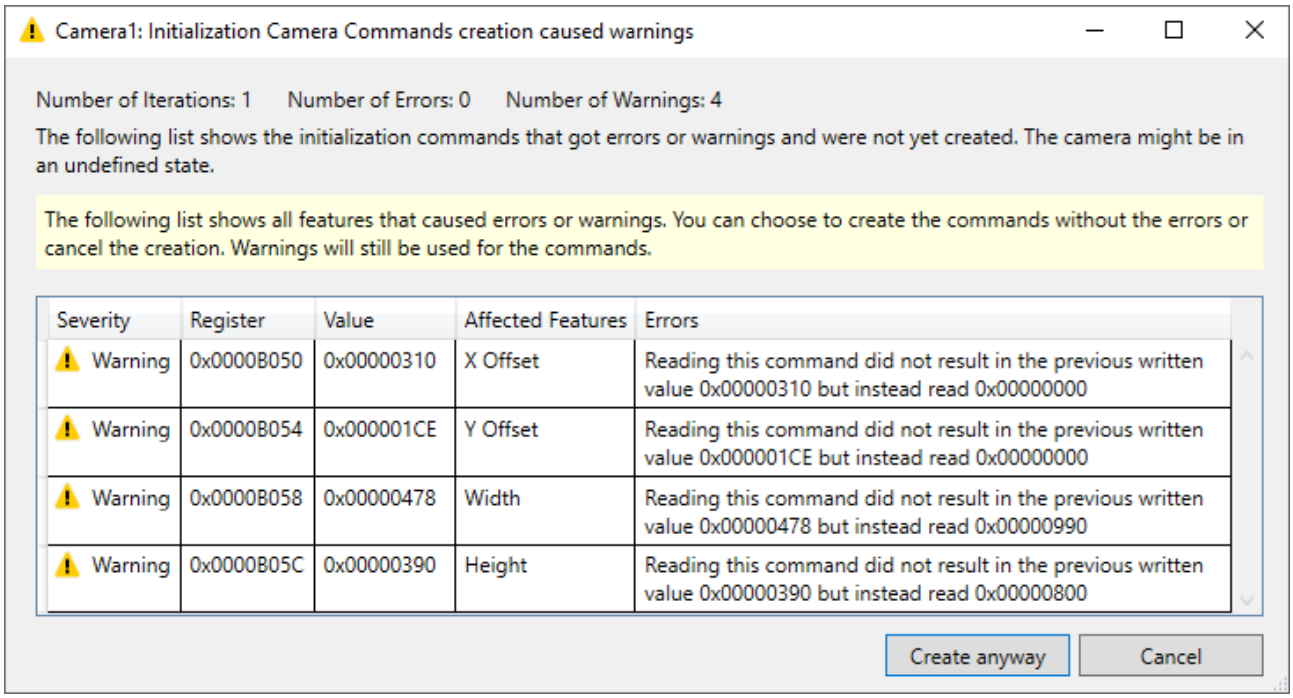
The GevImageAcquisition object is GigE Vision certified as part of the TC3 GigE Vision Connector product and therefore behaves in a standard-compliant manner. Concatenating commands, if supported by the camera, and iterating selectors is active by default in the settings. This means that all possible register combinations are read or written when communicating with the camera. This can occasionally lead to problems due to the amount of commands or the timing. To remedy this, you can disable **EnableConcatenatedCommands** in the Image Acquisition Module or the **Iterate Selectors automatically** in the settings.

Messages

The components described above and their behavior have different effects on the creation of the Initialization Commands or on the possible messages.

Warnings

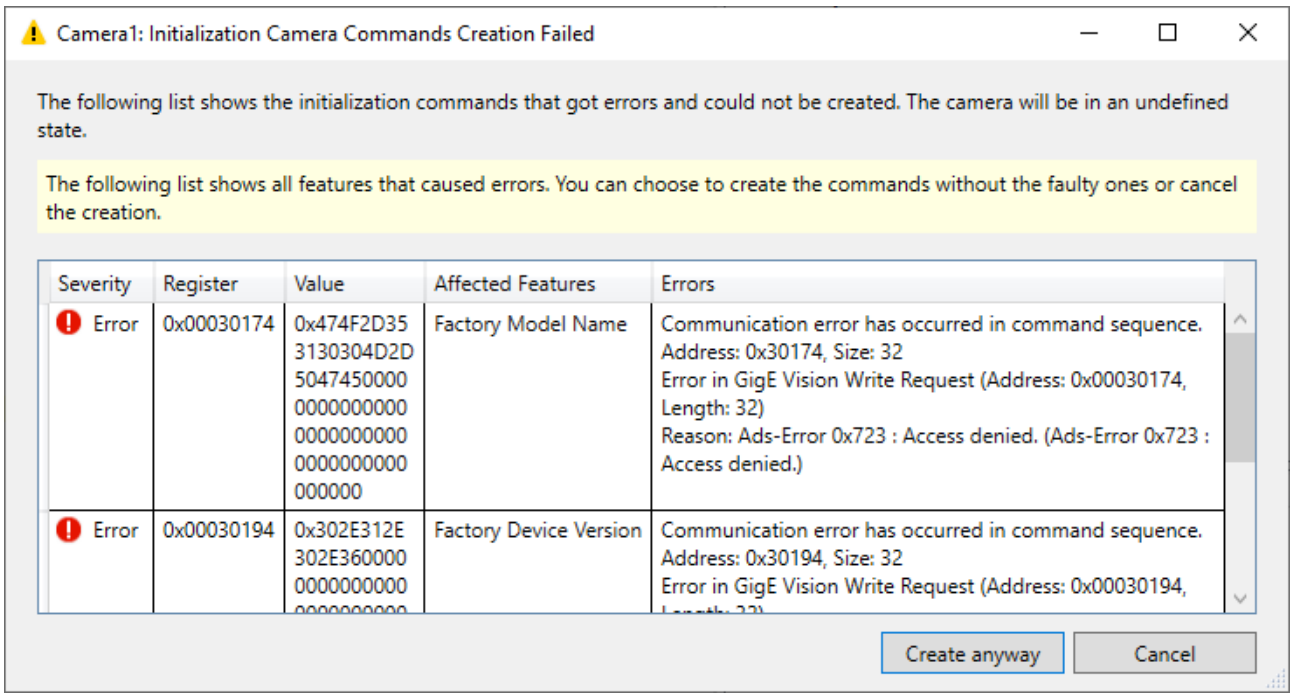
Warnings are displayed if the values read back differ from the values previously written. The differences can be caused by different camera behavior or dependencies between registers. This does not mean that the camera does not work with the Initialization Commands. Therefore warnings have no influence on the creation with **Create anyway**. Afterwards you should test if the reported registers have the correct value and the camera behaves as desired.



An example of when warning messages may occur is when an ROI is set. Depending on the current values in the camera, the warning occurs because the offsets are written first according to the GenApi order. If an offset greater than 0 is to be written, the width or height must be correspondingly smaller than the maximum value. In this case, the camera then resets all ROI values, causing the difference. In this case, the order of the ROI registers in the configuration list can be changed so that the width and height are written before the two offsets. If necessary in special cases, a register can also be copied, shifted in the order and thus written several times.

Error

Errors are reported if individual register values could not be written. As described above, this can have various causes, such as incorrect or changed limit values, addresses, write permissions or formula calculations at the time of writing. The affected registers are omitted when creating with **Create anyway** and checked off in the configuration list. Subsequently, it should be tested whether the omitted registers have an influence on the camera functionality.



If, for example, as can be seen in the screenshot, it is only the model name, or the register has not been changed and has a suitable default value, the camera can be used without restrictions despite this error message.

Fatal errors

Fatal errors only occur if the camera does not respond within a certain time. Then the process is aborted and one of the following messages appears. In these cases, the corresponding timeout must be adjusted.

Timeout

Force IP or GVCP timeout

In the course of writing the Initialization Commands, the Force IP command is also sent if it is configured. If the respective camera takes a long time to set the IP with this command, the following timeout error can occur:



Fig. 2: Error during Force IP

Persistent IP addresses

If no Force IP is used, it is recommended to set a persistent IP address.

In this case the **Force IP** can be deactivated in the configuration list [▶ 88]. Alternatively, or in the event of a GVCP timeout, the `TransmissionTimeout` and/or the `MaxTimeouts` can be increased in the TcCOM parameters from GVCP Module [▶ 100]. Note that the `ADS Communication Timeout` in the Settings window [▶ 84] should always equal at least the product of the two preceding parameters.

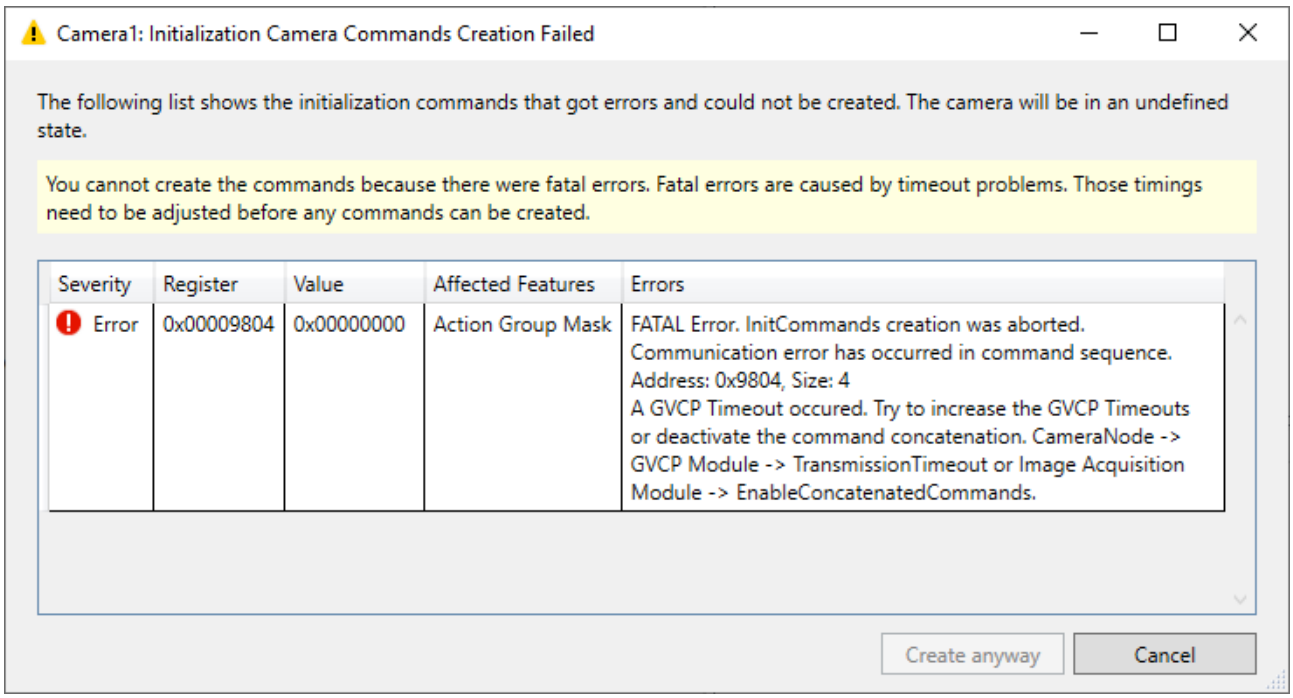


Fig. 3: Critical GVCP timeout error when creating Initialization Commands

ADS Timeout

In case of an ADS Timeout, increase the parameter `ADS Communication Timeout` in the settings window [▶ 84].

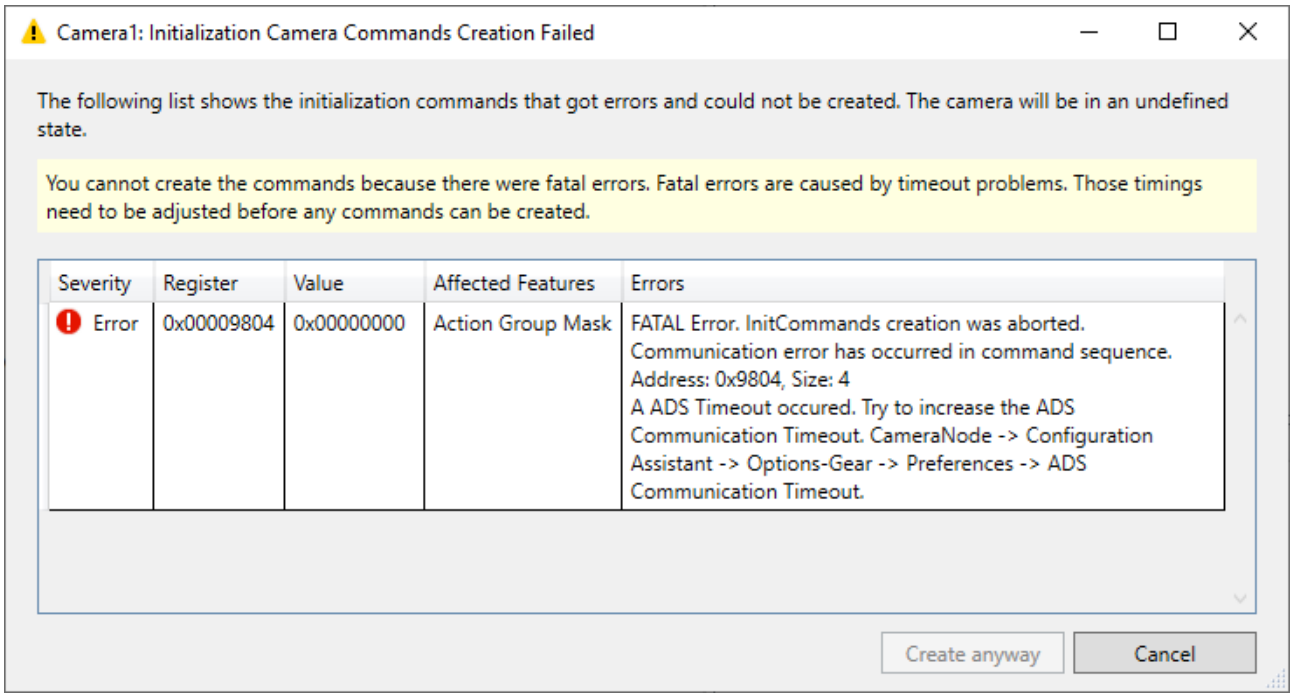


Fig. 4: Critical ADS timeout error when creating Initialization Commands

5.4.2.5 TcCOM parameters

The parameters of the three TcCOM modules Image Acquisition, GVCP and GVSP are described below. Adjusting these parameters can help certain cameras to establish a stable connection. The configuration tree [▶ 78] should be used to change the parameters.

Initially, only the parameter `InitializationAutoMode` is relevant here. The setting of the remaining parameters can, however, be necessary in the case of connection or initialization problems.

Reloading in the case of parameter changes

For parameter changes to become effective, the corresponding objects must be reloaded.

Image Acquisition module

Feature	Description
CameraIPAddress	IP address of the camera. This IP address is used with the Force IP command. If this IP address is changed, the Initialization Commands [▶ 87] must be re-created in order for the change to become effective. In addition, the corresponding option in the Configuration [▶ 88] must be activated for the Force IP command.
InitializationTimeout	Maximum time for transferring the initialization parameters to the camera. Depending on the camera used, the transmission may take longer, in which case the parameter has to be adjusted accordingly. Unit: milliseconds (ms) Default: 2,000
InitializationAutoMode	Selection of the way in which the initialization of the camera is to be executed automatically by the ImageAcquisition TcCOM object. Communication with the camera is impossible if it has not been initialized. **AUTOINIT_SO**: the camera is initialized during the start-up phase of the TcCOM object. In this phase only a limited amount of time is available. If initialization parameters are used, the initialization normally cannot be carried out within this start-up phase on account of the potential timeout. In this case the behavior of AUTOINIT_AFTER_SO is applied, even if AUTOINIT_SO is selected. **AUTOINIT_AFTER_SO**: the TcCOM object enters the OP state without initializing the camera. Only when the connection to the camera is opened from the PLC (e.g. with `FB_VN_SimpleCameraControl.StartAcquisition()` or with `FB_VN_GevCameraControl.OpenCamera()`), the camera is initialized automatically. **NO_AUTOINIT**: the camera is only initialized when `GevCameraController.InitializeCamera()` is called. This option is conceived for the case that full control over the initialization is required in order to execute it at a certain time. Standard: AUTOINIT_AFTER_SO.
EnableConcatenatedCommands	Option to select whether the associated commands should be linked or sent individually when writing several commands to the camera, for example the initialization parameters. The linking of the commands makes the transmission quicker, but may lead to problems with some cameras. Default: activated.

GVCP module

Feature	Description
LocalPort	The local UDP port through which the GVCP module communicates. If the value is zero, an automatically generated port is used. Default: 0
MaxTimeouts	The maximum number of transmission timeouts that may occur before a GVCP packet is discarded. After each timeout the GVCP packet is resent. If the maximum number of timeouts is reached, an error is output. Default: 3 (adapted to the camera)
TransmissionTimeout	After sending a GVCP packet, a response is awaited. If the response takes longer than the TransmissionTimeout time, it is assumed that the packet has not arrived, in which case the packet is retransmitted. Unit: milliseconds (ms) Default: 600 (adapted to the camera) **The TransmissionTimeout should be less than the HeartbeatRate.**
HeartbeatRate	The control channel of the application sends a heartbeat signal to the camera to check if the connection is still valid. The HeartbeatRate describes the time after which the next signal is sent. Unit: milliseconds (ms) Default: 1,000 **The HeartbeatRate should not be set arbitrarily high, since many cameras trigger a timeout after a few seconds without heartbeat.**
DiscoveryTimeout	The DiscoveryTimeout describes the time taken to wait for a Discovery command to respond. The Discovery command is sent when searching for cameras [▶ 69] in the network. Unit: milliseconds (ms) Default: 1,000

GVSP module

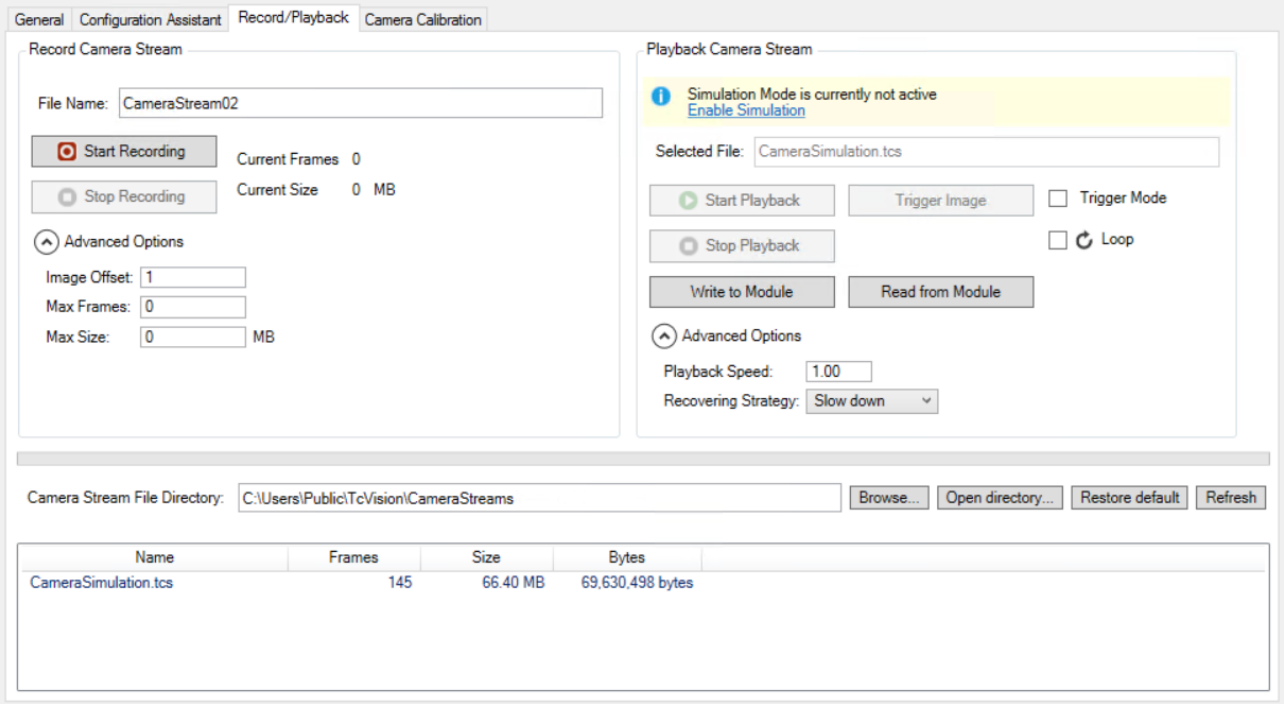
Feature	Description
LocalPort	The local UDP port through which the GVCP module communicates. If the value is zero, an automatically generated port is used. Default: 0
MaxPacketSize	The **MaxPacketSize** describes the maximum packet size of a GVSP packet. The header of the Ethernet frame is not included in this size. Unit: byte Default: 1,500 **Depending on the installed TwinCAT real-time Ethernet driver and the Jumbo packet size set in the properties, up to 9000 bytes can be used.**
MinPacketDelay	If the **MinPacketDelay** is increased, the GVSP packets are artificially delayed. This allows the bandwidth to be reduced. Unit: timestamp counter unit(s) Default: 0
MaxResendRequests	The maximum number of resend requests for a single GVSP packet that was faulty or took too long. An image is transmitted in a GVSP block consisting of several packets. The image is invalid if a packet of a block could not be successfully transmitted. Default: 1
MaxBlockNumber	Maximum number of GVSP blocks that can be processed simultaneously. If the value is zero, the number is unlimited. Default: 5
MaxPacketNumber	The maximum number of GVSP packets in a block. Blocks containing more packets cannot be interpreted as a valid image. Default: 25,000 **The number is calculated automatically if the option "Automatically calculate the MaxPacketNumber" is active in the Preferences settings.**
BlockTimeout	The **BlockTimeout** describes the time that may elapse between individual packets of a block. Unit: microseconds (µs) Default: 5,000

5.4.3 Record/Playback

Camera streams can be recorded and input to the PLC on the **Record/Playback** tab. This can be helpful for the development in order to record a camera stream on the desk and repeatedly play it back offline at a later time. The real behavior of the camera hardware is thus simulated with the original images.

Either the simulation mode is activated, in which case camera streams can be played back, or the simulation mode is deactivated, in which case camera streams can be recorded. It is not possible to do both simultaneously.

A camera stream consists of individual images that are saved together with the corresponding time markers in their own format (.tcs). Through the additional saving of the time markers, the time behavior of the camera on the machine can be simulated with application realism. The image data are saved in precisely the format in which they are sent by the camera; no further conversion takes place.



Camera requirement

The camera must support timestamps and enter valid values in the image header. The timestamps are needed per image to be able to simulate the temporal behavior.

System requirements

Use fast SSD hard disks and at least 8 GB RAM for the recording and playback of camera streams. If slower storage media are used, reduce the image rate if necessary via the parameter **Image Offset**.

(De)activating TcCOM objects

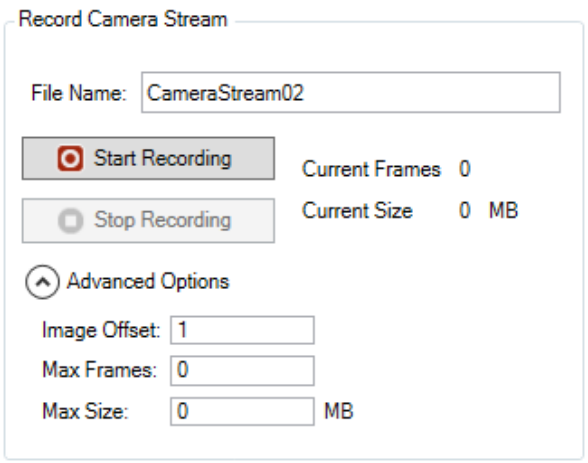
The activation/deactivation of the camera TcCOM objects should not be done manually, as further links have to be adapted when changing from the simulation mode and from the Ads Communication in the background.

The three areas of the Record/Playback assistant (Record, Playback, Stream files) are explained below.

Recording a camera stream:

The recording of a camera stream is only possible with an active camera connection. Accordingly, the simulation mode of the camera instance [▶ 69] must be disabled and the standard acquisition selected. The Ads Communicator TcCOM module is also required.

The settings for **Record Camera Stream** are located in the Record Camera Stream area:



File Name	Text box for the file name of the camera stream. The path used to save the file is defined in Selection of file path and streams [▶ 106].
Start Recording	Button for starting a recording. Only active if recording is possible.
Stop Recording	Button for stopping a recording. Only active during recording.
Current Frames	Number of images in the currently recorded camera stream.
Current Size	File size of the currently recorded camera stream.
Advanced Options	Option to show and hide advanced options.
Image Offset	Specifies the image recording sequence. Depending on the writing speed of the storage medium, the image size and the frames per second, it may be necessary to skip images. Default: 1 (i.e. an attempt is made to record every image).
Max Frames	Maximum permitted number of images in the camera stream to be recorded. The recording will be ended automatically as soon as this number of images is reached. Default: 0 (i.e. no limit).
Max Size	Maximum allowed file size of the camera stream to be recorded in MB. As soon as this file size is reached, the recording is automatically stopped. Default: 0 (i.e. no limit).

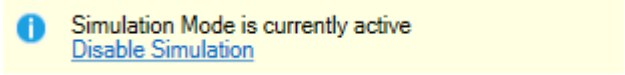
Enough free storage space?

Before recording a stream, make sure that there is sufficient storage space on your target system or limit the stream size if necessary with **Max Frames** and **Max Size**.

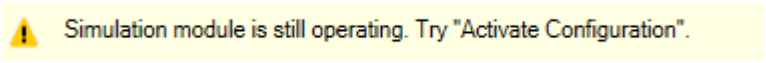
The following conditions must be met for a stream to be recorded:

- Simulation mode is disabled.
- An Ads Communication object exists and is activated.
- TwinCAT is in Run mode.
- The camera must be in the `ACQUIRING` status.

If these conditions are not met, the following notes and suggested solutions may appear:



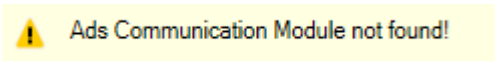
- Simulation mode means a state in which a stream can be played back but not recorded. The state can be changed via the **Disable Simulation** link in the note. This also requires that the configuration is reactivated .

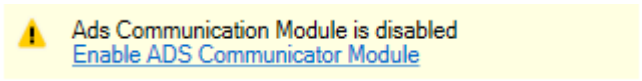


- Simulation mode has been switched off, but the change is only applied after the configuration has been activated.

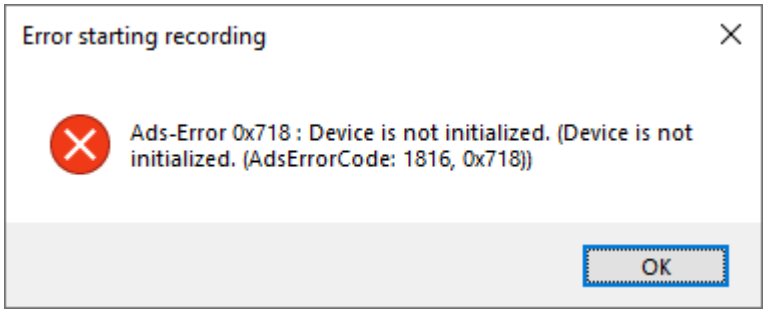


- The recording of a camera stream is only possible in TwinCAT Run mode. This warning is displayed in all other TwinCAT states.





- Recording a camera stream is only possible if there is an open connection to the camera, as data is read from the camera before recording. Therefore, the camera should already be in `ACQUIRING` status at this point.



- The Ads Communication module is required for recording camera streams. It can be activated via the link in the note. If no ADS Communicator object exists yet, it can be added via **Add New Item…** on the camera object and **Try to fix** on the General tab [▶ 71]. The change is accepted after activating the configuration  .

Activating an ADS Communicator module in advance

If you plan to record a camera stream at a later time without first stopping TwinCAT (e.g. during the runtime of a machine), you must activate the ADS Communicator module in advance.

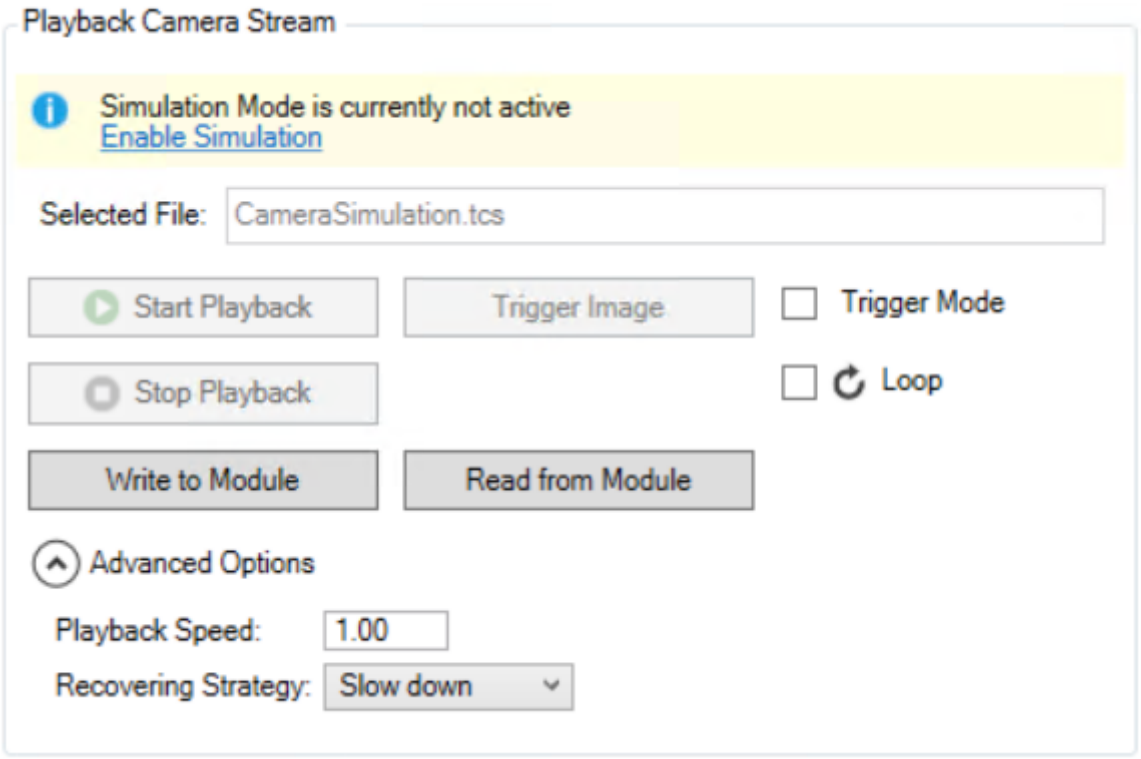
If the Ads Communicator object is activated, it is possible that images from the camera may still be referenced internally in the object for a short time and therefore cannot be directly transformed into displayable images. For further details see Displayable images [▶ 151].

Playback of a camera stream:

In order to play back a camera stream, the simulation mode of the camera must be activated; this is possible either via the General [▶ 71] tab of the camera object or in the message mentioned below.

If you write the configuration with Write to Module into the TcCom object and save it in the project, these values are already available after the activation of the project and the simulation starts directly with the Start Acquisition command from the PLC. The advantage of this is that you do not have to manually reset the settings and start the stream every time.

The settings for playing back a camera stream are located in the **Playback Camera Stream** area:



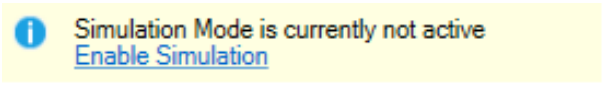
Selected File	Text box for displaying the selected stream file. The selection is made from the list of available streams.
Start Playback	Button for starting the playback. Only enabled if playback is possible. The playback always begins with image 1, irrespective of where the stream may have been stopped previously.
Stop Playback	Button for stopping the playback. Only enabled while a stream is played back.
Trigger Mode	Selection of the Trigger mode, in which the frames in the stream can be triggered individually.
Trigger Image	Button for triggering an image. Only active when Trigger mode is selected and while a stream is played back.
Loop	Option to select whether the stream is to be played once or in an infinite loop.
Write to Module	Writes the current settings to the TcCom module. Stored are: Playback Mode, Playback Speed, Playback Strategy, Loop and the path of the selected stream file.
Read from Module	Reads the stored values from the module and sets these settings.
Advanced Options	Show / hide advanced settings.
Playback Speed	Acceleration/deceleration factor for playback speed. (>1 acceleration; <1: deceleration) Default: 1.00 = speed matches the recording
Recovering Strategy	Strategy in the event that playback at the specified speed is not possible: • Slow down: Playback speed delayed (default setting) • Skip images: Skipping of images • Abort on lag: Abortion of the playback (not recommended)

The following conditions must be met for a stream to be played back:

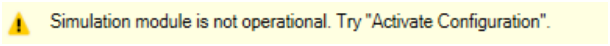
- Simulation mode is enabled.

- TwinCAT is in Run mode.

The following notes and warnings may appear above the text field with the selected file name:



- Simulation mode is the state under which a stream can be played back. This state can be reached via the **Enable Simulation** link and subsequent activation of the configuration  .



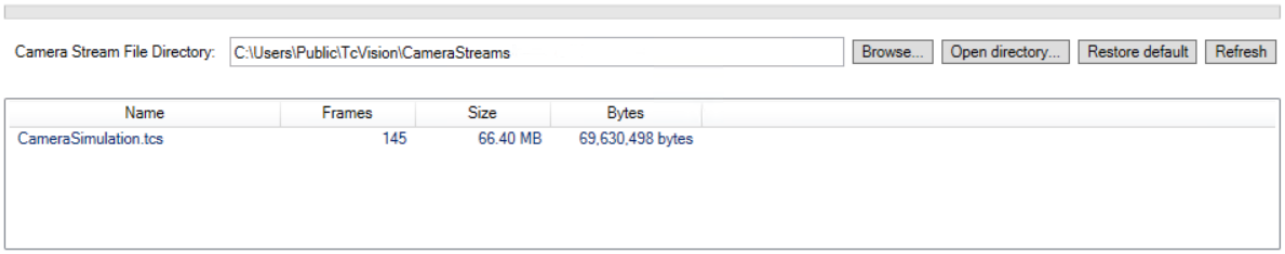
- Simulation Mode has been enabled, but the change only becomes active after the configuration has been enabled.

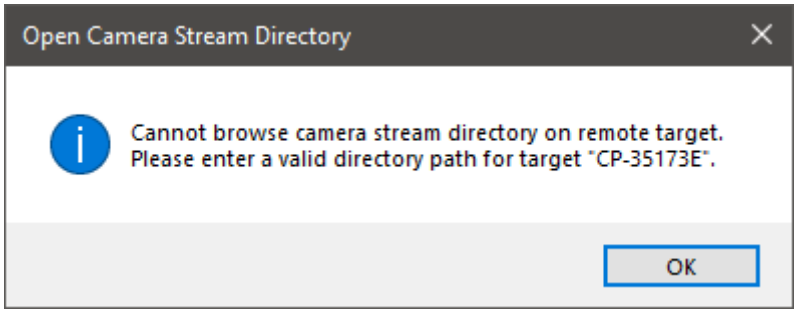
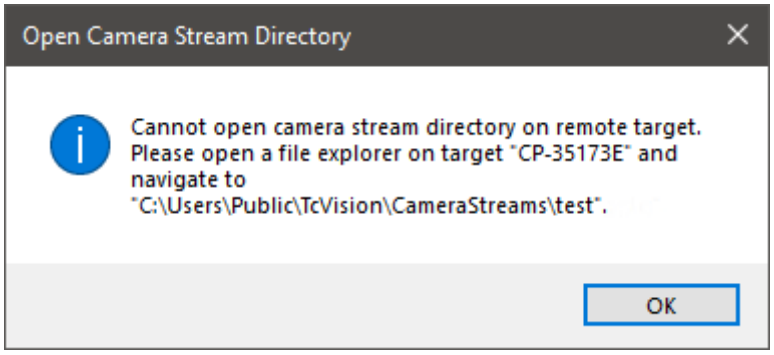


- Playback of a camera stream is only possible in TwinCAT Run mode. This warning is displayed in all other TwinCAT states.

Selection of file path and streams

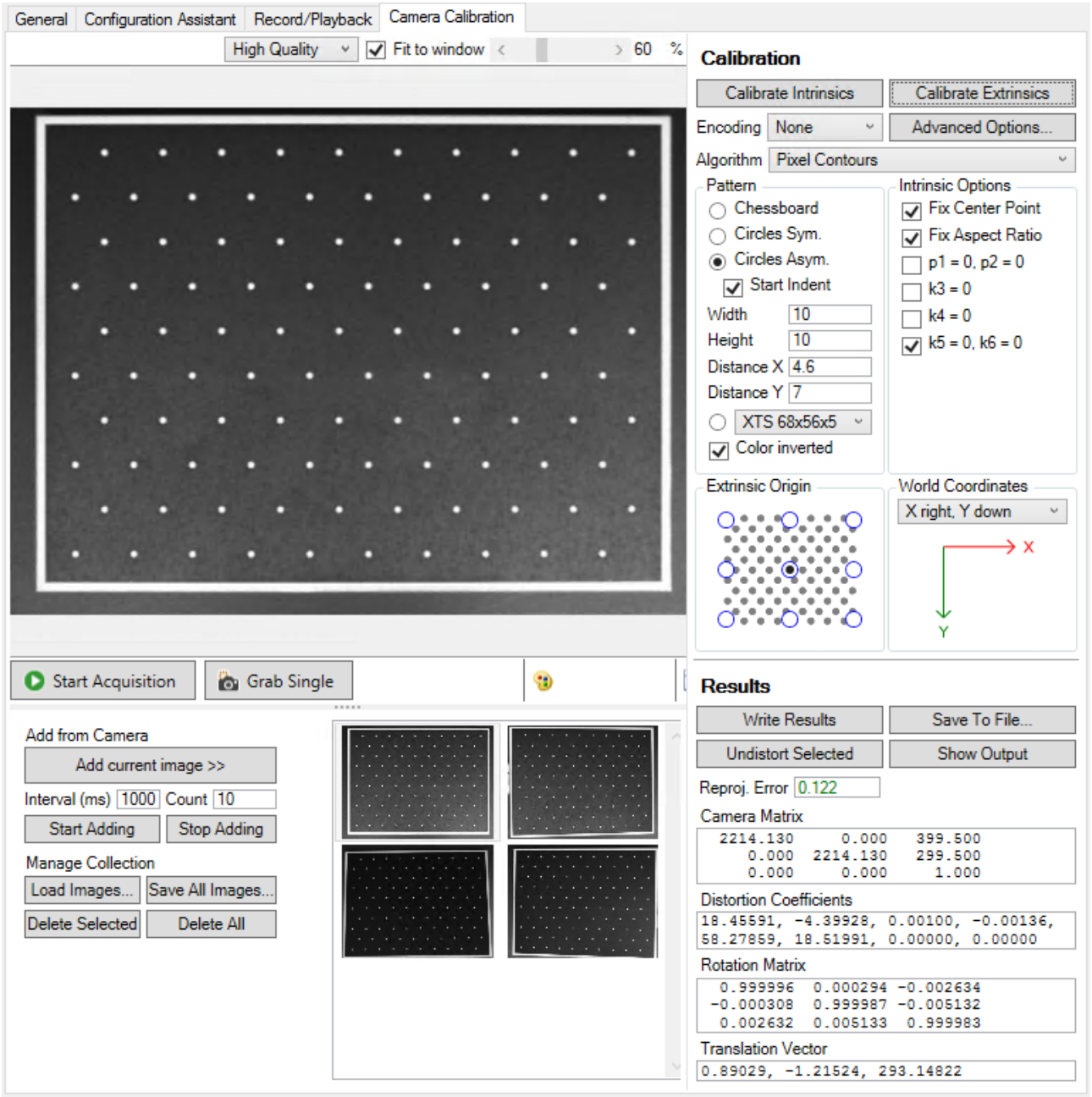
The file path and stream selection is shown below:



	Status bar for recording or playback
Camera Stream File Directory	Currently set file path. This path is used both for reading already recorded streams and for writing new streams. Subfolders are also searched for stream files. **The file path always refers to the target system, not the development system.** **Make sure that this file path is not affected by write filters or similar.** See Important paths [▶ 2860].
Browse…	This button opens a selection window for browsing the destination folder. **This is only possible when the development system is also the target system.** Otherwise the following message appears: 
Open directory…	Button for opening the specified file path in the local File Explorer. This can also be done by double-clicking in the free area in the file list. **This is only possible when the development system is also the target system.** Otherwise the following message appears: 
Restore default	Button for setting the default file path.
Refresh	Button to update the stream list.
Name	List of all camera streams in the selected file path, which are also used for stream selection for playback.
Frames	Number of images of which the respective camera stream consists.
Size	File size of the respective camera stream in MB.
Bytes	The size specification in bytes.

5.4.4 Camera Calibration

Area scan cameras can be geometrically calibrated with the help of a calibration pattern on the Camera Calibration tab.



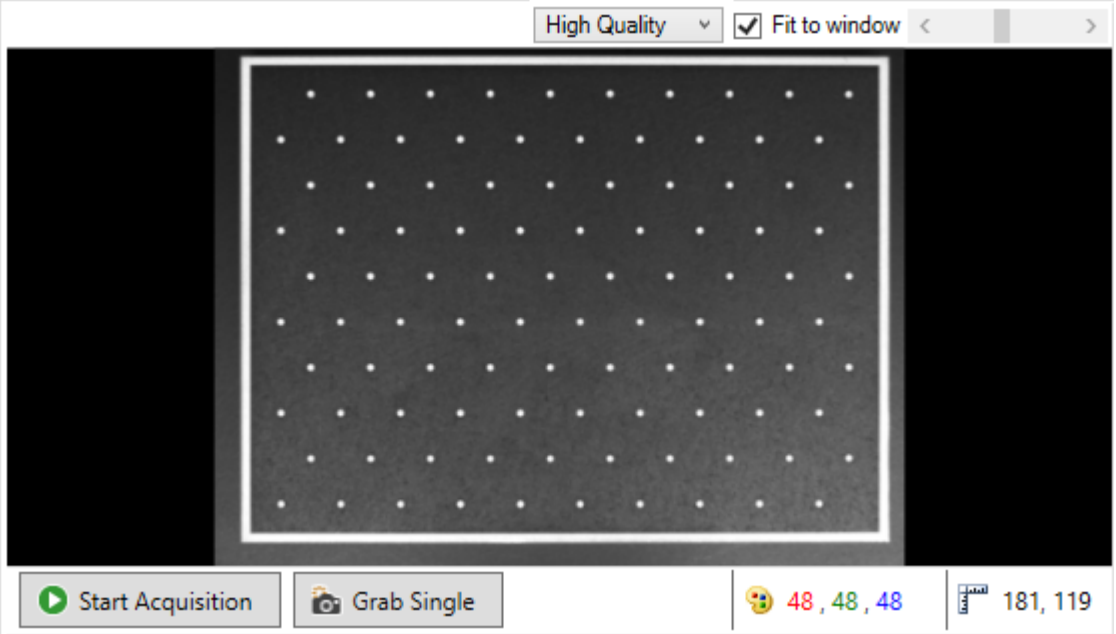
During geometric camera calibration of area scan cameras, the imaging process is reconstructed from the image. For this purpose, points in the image are required whose positions in the real world are known. Calibration patterns with defined points are available from which images are taken.

Supported calibration patterns:

- Chessboard pattern [▶ 116]
- Symmetrical circles pattern [▶ 117]
- Asymmetrical circles pattern [▶ 118]
- Individual circles pattern [▶ 119]

Image acquisition / selection

As in the Configuration Assistant [▶ 75] there is a preview image for the image acquisition above which there are display options and below which there are controls for the image acquisition:



Displaying the preview image

The image is always displayed just as it is received from the camera. An image in the Bayer format will therefore be displayed unencoded as a grayscale image. At this point the set encoding is not yet applied to the image. The encoding is relevant only for the calibration process, but not for the preview image.

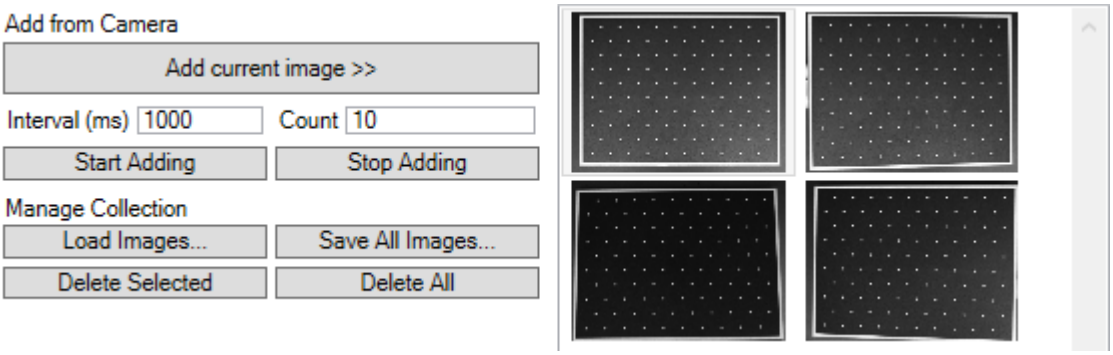
Above the preview window:

- **Interpolation of the preview image:**
  Combo box for selecting whether a high-quality or nearest-neighbor interpolation of the preview image should be performed. The selection only refers to the preview. The calibration is based on the original image data.
- **Size of the preview image**
  If **Fit to window** is selected, the image is scaled to the size of the preview window. Alternatively, a display size can be specified via the slider.

Below the preview window:

- **Start Acquisition / Stop Acquisition**
  Button for starting or stopping image acquisition. Streaming mode of the camera must be active.
- **Grab Single**
  Button for triggering a single image.
- **Pixel Value**
  Shows the color value of the pixel at the mouse cursor position on the image.
- **Cursor Position**
  Shows the mouse cursor position on the image. Please note that the zero point for images is always at the top left.

If the calibration pattern is within the camera's field of view, the corresponding image can be selected for calibration. The Controls are located below the preview image on the left, the collection of images for calibration on the right.

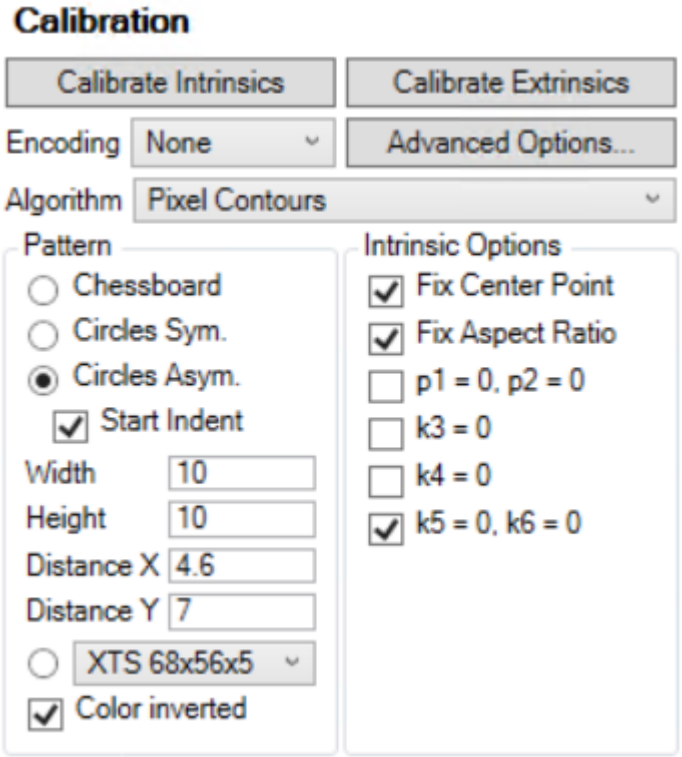


- **Add current image >>**
  Add current image adds the currently displayed camera image to the collection for calibration.
- **Start / Stop Adding**
  Instead of a single image, a series of images can be added to the collection. The image series is started via **Start Adding**. Subsequently, images are added at intervals of the time set for **Interval** until the number entered for **Count** is reached or the addition of images is terminated via **Stop Adding**.
- **Load Images...**
  As an alternative to image acquisition, saved images can be loaded via **Load Images**. Note here that these were recorded with the same camera system, the same camera settings, the same lens settings and the same calibration pattern.
- **Save All Images...**
  Save All Images can be used to save the complete image collection.
- **Delete Selected**
  Delete Selected deletes the selected images from the collection.
- **Delete All**
  Delete All deletes all images from the collection.

Image format

At this point only Mono8, RGB24 and Bayer8 images may be used.

Settings and calculations



Encoding

For measurement tasks we recommend using a monochrome camera. However, if it should be necessary to calibrate a color camera with Bayer pattern, the corresponding Bayer pattern must be specified by **Encoding**. This is automatically selected by default.

Algorithm

For circle pattern detection, you can choose between different edge localization functions. The subpixel function can be used to improve the result depending on the image quality. However, this also means that the entire process takes more time. See the chapter Measurement [▶ 1481] for more information on the subpixel functionality.

- **Pixel Contours**
  Contour-based
- **Subpixel Interpolation**
  With Subpixel by interpolation
- **Subpixel Erf Approximation**
  With Subpixel after Erf Approximation

Definition of the calibration pattern

The calibration pattern used is selected and defined under **Pattern**:

- **Chessboard**
  A chessboard pattern is used
- **Circles Sym.**
  A symmetrical circles pattern is used
- **Circles Asym.**
  An asymmetrical circles pattern is used
  - **Start Ident**
    Checkbox to specify indentation of the first row of the circle pattern. In the standard system without a check mark, a row that is not indented is expected.
- **Width**
  Number of feature points in horizontal direction
- **Height**
  Number of feature points in vertical direction
- **Distance X**
  Distance in the world coordinate system of the feature points in horizontal direction
- **Distance Y**
  Distance in the world coordinate system of the feature points in vertical direction
- **Drop-down menu for individual circle patterns**
  Selection of an individual circle pattern (entries are only displayed if corresponding files are found in the specified folder)
- **Color inverted**
  Checkbox for selecting color inversion of the image. By default (no check mark), black objects are expected on a white background.

Symmetrical pattern

In the case of wholly symmetric patterns, the assignment of the points found may deviate from those defined or the coordinate origin may lie in a different quadrant. This can also occur if the pattern from the selected image is in a different orientation to that defined.

Therefore, after completion of the calibration procedure, you should check whether the coordinate system corresponds to your specification.

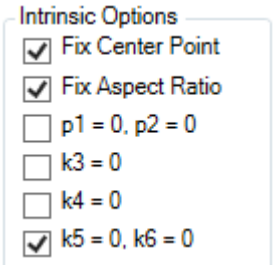
Check for filled circles

In the case of circle patterns, an additional check is made to see whether the circles are filled in the corresponding color. If deviations occur at this point, the corresponding point is not recognized as part of the pattern.

For information on patterns and parameter settings, please refer to the supported calibration pattern types:

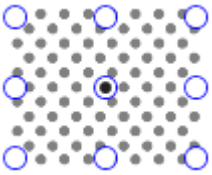
- Chessboard pattern [▶ 116]
- Symmetrical circles pattern [▶ 117]
- Asymmetrical circles pattern [▶ 118]
- Individual circles pattern [▶ 119]

Intrinsic Options



- **Fix Center Point:** the principle point corresponds to the center of the image.
- **Fix Aspect Ratio:** the camera sensor is assumed to have square pixels, so that the aspect ratio of the pixels is the same in the x and y directions.
- **p1=0, p2=0:** it is assumed that no tangential distortions exist, and the corresponding coefficients are set to 0.
- **k3=0:** the radial distortion coefficient k_3 is assumed to be 0
- **k4=0:** the radial distortion coefficient k_4 is assumed to be 0
- **k5=0, k6=0:** the radial distortion coefficients k_5 and k_6 are assumed to be 0

Extrinsic Origin



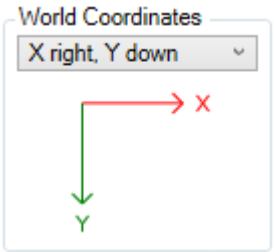
Extrinsic Origin specifies the origin point of the world coordinate system in the calibration pattern.

Depending on the selected calibration pattern, a corresponding background image is displayed.

Origin

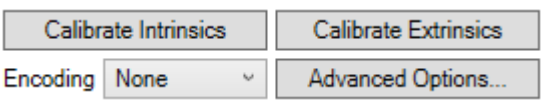
The selected origin refers to the corresponding point of the calibration pattern, not to the complete image.

World Coordinates



World Coordinates sets the orientation of the world coordinate system. This can be used to set a different orientation compared to the image coordinate system. With the default setting shown, the two coordinate systems match.

Actions



Calibrate Intrinsics

Calibrate Intrinsics calculates the camera matrix and the distortion coefficients. All images of the selection are used for this purpose. These must all contain the defined calibration pattern in its entirety. If this is not the case, a negative **Reproj. Error** is displayed.

Calibrate Extrinsics

Calibrate Extrinsics calculates the rotation matrix and the translation vector. A prerequisite for this is the intrinsic calibration (camera matrix and distortion coefficients) and selection of the image where the points of the 2D calibration pattern are located in the subsequent measuring plane.

Duration of a calibration

The duration of a calibration increases with the number of reference points in the calibration pattern and the number of calibration images. Therefore, when choosing a calibration pattern, make sure that the duration remains within acceptable limits for you.

Deviation of the results

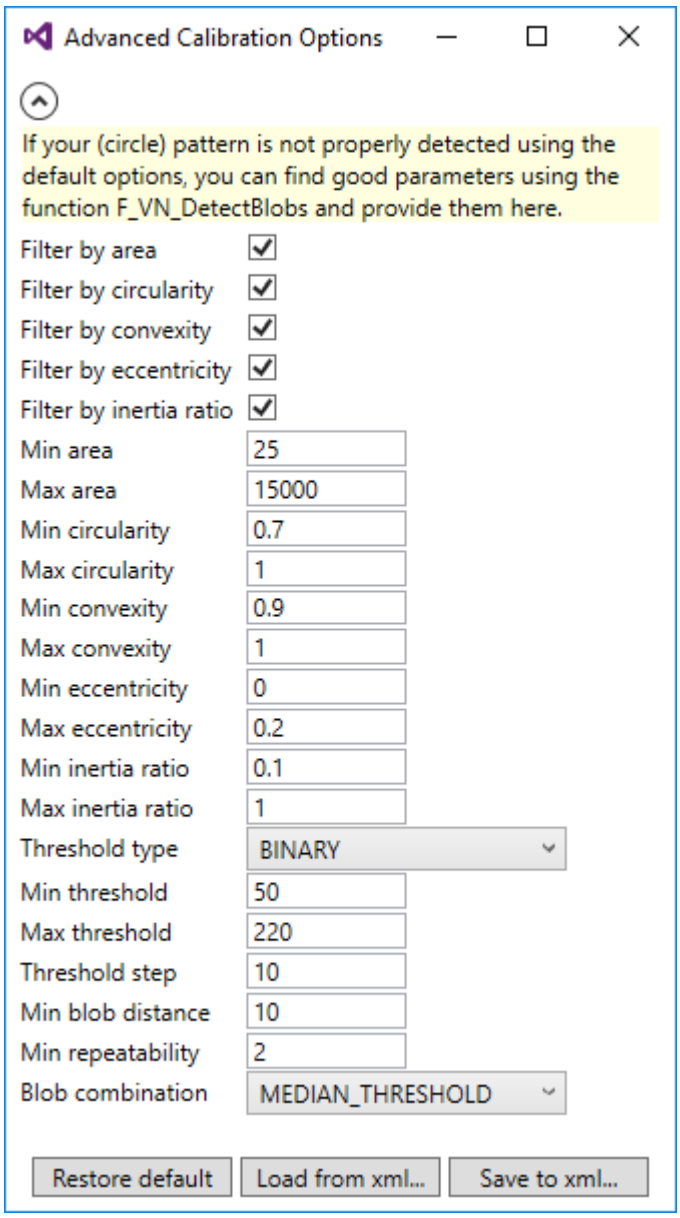
On account of the algorithm used, the results of the calibration may differ slightly from one another in the decimal place range from call to call. This concerns both the results of the calibration in the calibration assistant and those in the PLC as well as between both calibration variants.

Transfer of the settings into the TwinCAT project

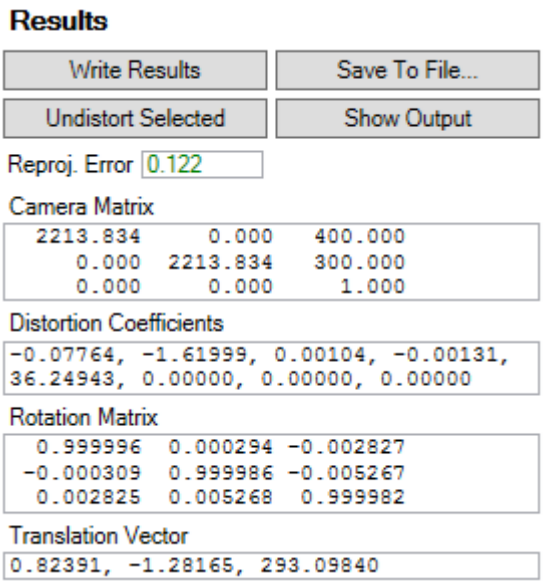
The set values and options are only applied to the TwinCAT project after executing the **Calibrate Intrinsic**s or **Calibrate Extrinsics** function. If the project is saved afterwards, these values will be reloaded when it is opened again. Otherwise, the default settings are always used.

Advanced Options

Circle pattern recognition can be reparameterized via the **Advanced Calibration Options**. This should only be done if there are difficulties with the default settings. For a better understanding of the setting parameters, see the parameterization of the function F_VN_DetectBlobs [▶ 1178].


Advanced Calibration Options
If your (circle) pattern is not properly detected using the default options, you can find good parameters using the function F_VN_DetectBlobs and provide them here.
Filter by area
Filter by circularity
Filter by convexity
Filter by eccentricity
Filter by inertia ratio
Min area 25
Max area 15000
Min circularity 0.7
Max circularity 1
Min convexity 0.9
Max convexity 1
Min eccentricity 0
Max eccentricity 0.2
Min inertia ratio 0.1
Max inertia ratio 1
Threshold type BINARY
Min threshold 50
Max threshold 220
Threshold step 10
Min blob distance 10
Min repeatability 2
Blob combination MEDIAN_THRESHOLD
Restore default
Load from xml...
Save to xml...

Results


Results
Write Results
Save To File...
Undistort Selected
Show Output
Reproj. Error 0.122
Camera Matrix
2213.834 0.000 400.000
0.000 2213.834 300.000
0.000 0.000 1.000
Distortion Coefficients
-0.07764, -1.61999, 0.00104, -0.00131, 36.24943, 0.00000, 0.00000, 0.00000
Rotation Matrix
0.999996 0.000294 -0.002827
-0.000309 0.999986 -0.005267
0.002825 0.005268 0.999982
Translation Vector
0.82391, -1.28165, 293.09840

Reproj. Error
The Reprojection Error indicates the result of the intrinsic calibration:

- **>0**: valid result
  - **<1**: good result
  - **>2**: inaccurate result
- **<0**: no result and specific error number:
  - **-100**: no image found
    The collection of images below the preview window is used for the intrinsic calibration. The live image of a camera is not taken into account.
  - **-200**: the pattern could not be found in at least one image.
    All images in the collection below the preview window must show the complete pattern, including all feature points. For a better analysis, result images are created under *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput*, which show the feature points that were found.
  - **-300**: none of the existing images contains the selected pattern.
    See description for -200, in addition, we recommend checking the pattern configuration.
  - **-400**: the user-defined description file for an individual calibration pattern is invalid.
  - **-500**: an error has occurred.

Camera Matrix

The camera matrix describes the imaging process from 3D to 2D.

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

- The parameters $\mathbf{f}_x$ and $\mathbf{f}_y$ indicate the focal length in the x and y directions.
- The parameters $\mathbf{c}_x$ and $\mathbf{c}_y$ indicate the intersection point of the optical axis with the image, which is referred to as the principal point. Ideally, this should be in the center of the image. If **Fix Center Point** is selected under **Intrinisic Options**, the parameters $\mathbf{c}_x$ and $\mathbf{c}_y$ are automatically set to the center of the image.

Distortion Coefficients

Aberrations of the optical system can lead to distortions in the image. These can be partly compensated by the **Distortion Coefficients** during the coordinate transformation.

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

- The coefficients $\mathbf{k}_1$ **to** $\mathbf{k}_6$ describe the radial distortions in the image.
- The coefficients $\mathbf{p}_1$ and $\mathbf{p}_2$ describe the tangential distortions in the image.

Rotation Matrix

The rotation matrix describes the orientation of the camera coordinate system relative to the world coordinate system and vice versa.

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

Translation Vector

The translation vector describes the displacement of the coordinate origin of the camera coordinate system relative to the world coordinate system and vice versa.

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

Write Results

The results are written to the Image Provider TcCOM object of the camera. This means that they can be called from the PLC via the methods of the function block F_VN_GevCameraControl [▶ 1621].

Even if a calibration has not yet been carried out, the description of the set calibration pattern will be written. This can be retrieved in the PLC with the method GetCalibPatternRef [▶ 1629]. Alternatively, the description of the calibration pattern can also be generated directly in the PLC via the F_VN_GenerateCalibrationPatternReferencePoints [▶ 1629].

Save To File...

The results can be saved in a json file. Alternatively, the results can be read into the PLC via the FB_VN_ReadCalibrationResult [▶ 1595].

Undistort Selected

The **Distortion Coefficients** are applied to the selected image, and the result is displayed in a separate window.

Show Output

Opens the directory *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput*. Result images are created in order to be able to analyze which feature points were found and how they were assigned.

Alternatives in the PLC

If a camera has to be calibrated while the machine is running, the function F_VN_CalibrateCamera [▶ 1073] is available for this in the PLC.

Calibration in the PLC

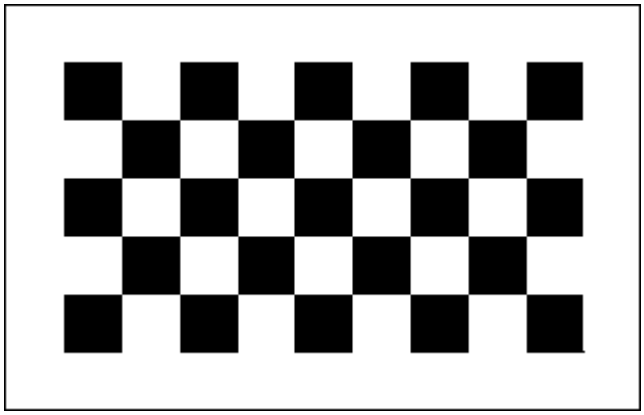
The duration of a calibration can be much longer than a PLC cycle. Therefore, make sure that no problems are caused in your machine sequence when you carry out a calibration in the PLC.

Calibration Assistant sample

Calibration Assistant [▶ 2838]

5.4.4.1 Chessboard pattern

A chessboard pattern consists of alternating light and dark squares, which together form a rectangular pattern. See chessboard example below with 4 x 8 intersections:



Requirements for the pattern

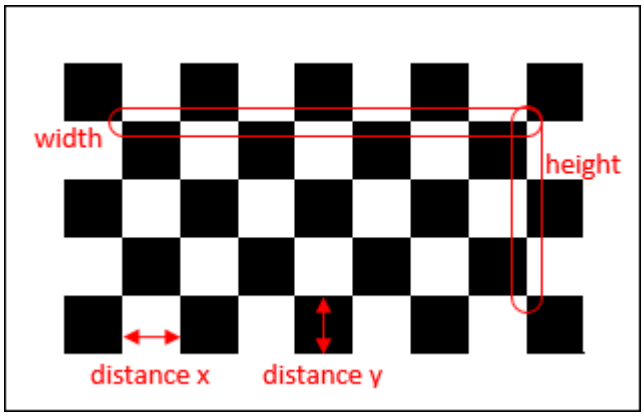
- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The light and dark squares must
  - be arranged in an alternating pattern,
  - have the same size
  - and be square.
- A pattern must contain at least 16 fully visible squares, which together form at least 3 x 3 intersections.
- The background color around the pattern must form a 'quiet zone'.

Requirements for the image acquisition

- Exposure
  - Homogeneous illumination of the pattern
  - Good contrast between the light and dark squares in the image
  - No opening at intersections due to overexposure
- Figure
  - A square must be represented by at least 15x15 pixels.

Configuration in the Camera Calibration Assistant [▶ 107]

- **Width**
  Corresponds to the number of intersections in horizontal direction.
- **Height**
  Corresponds to the number of intersection points in vertical direction.
- **Distance x**
  Corresponds to the distance between two intersection points in horizontal direction.
- **Distance y**
  Corresponds to the distance between two intersection points in vertical direction.



5.4.4.2 Symmetrical circles pattern

A symmetrical circle pattern is composed of circles that are arranged evenly in rows and columns.

Other supported patterns with circles are asymmetric circle pattern [▶ 118] and individual circle pattern [▶ 119].

Requirements for the pattern

- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The circles should be as equal in size as possible.
- A sample must consist of at least 9 points.

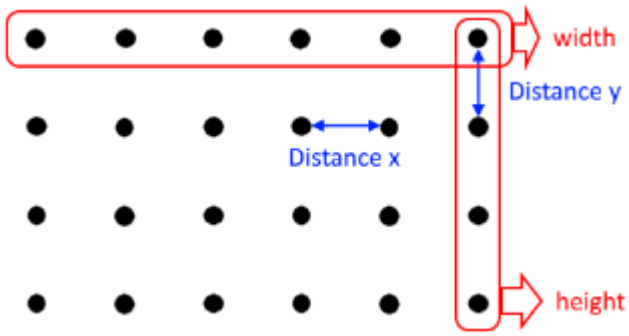
- The circle centers must be applied at the same distance horizontally and vertically. The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

Requirements for the image acquisition

- Exposure
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.
- Figure
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the **Advanced Calibration Options** must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

Configuration in the Camera Calibration Assistant [▶ 107]

- **Width**
  Corresponds to the number of circles per row.
- **Height**
  Corresponds to the number of circles per column.
- **Distance x**
  Corresponds to the distance between two circle centers in horizontal direction.
- **Distance y**
  Corresponds to the distance between two circle centers in vertical direction.



- **Color inverted**
  By default, dark circles on a light background are assumed. If the pattern instead consists of light circles on a dark background, the colors can be inverted with **Color inverted** option.

5.4.4.3 Asymmetrical circles pattern

An asymmetric circle pattern is similar to a symmetric circle pattern [▶ 117], except that every second row is offset by half a column. This allows a smaller distance between two rows, allowing more lines in the same area.

Pattern requirements

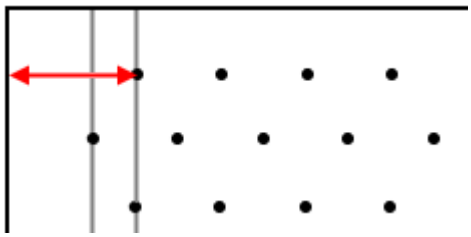
- The pattern should be applied to a surface that is as flat as possible (planar), e.g. a glass pane.
- The circles should be as equal in size as possible.
- A sample must consist of at least 9 points.
- The circle centers must be applied at the same distance horizontally and vertically. The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

Requirements for the image acquisition

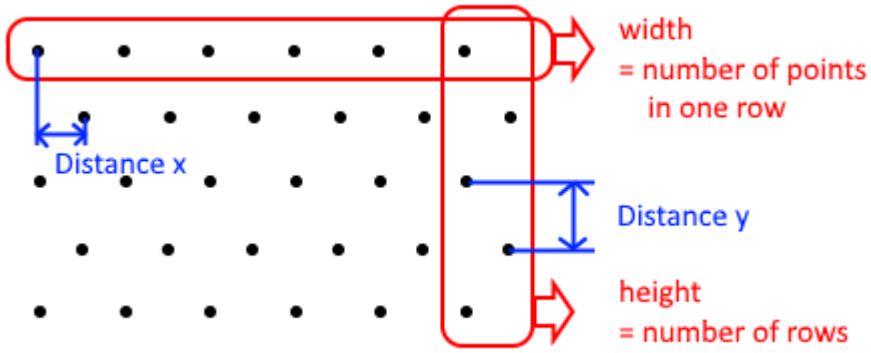
- Exposure
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.
- Figure
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the **Advanced Calibration Options** must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

Configuration in the Camera Calibration Assistant [▶ 107]

- **Start indent**
  The first row is indented relative to the second.



- **Width**
  Corresponds to the number of circles per row.
- **Height**
  Corresponds to the number of circles per column.
- **Distance x**
  Corresponds to the distance to the nearest circle center in horizontal direction (across rows).
- **Distance y**
  Corresponds to the distance to the nearest circle center in vertical direction (within a column).



- **Color inverted**
  When a light pattern is used on a dark background.

5.4.4.4 Individual circles pattern

An individual circle pattern consists of circles, with the option of user-defined position and thus arrangement of the individual circle components. This also enables the use of complex custom 3D patterns for calibration.

Requirements for the pattern

- The pattern should be applied to surfaces that are as flat (planar) as possible
- The circles should be as equal in size as possible.
- A pattern must consist of at least 9 points.

- The distance must be large enough to ensure that the circles do not touch each other and are recognized as separate objects.
- The background color around the pattern must form a 'quiet zone'.

Requirements for the image acquisition

- Exposure
  - Homogeneous illumination of the pattern.
  - Good contrast between circles and background.
- Figure
  - Perspective distortions should be kept to a minimum.
  - The standard specifications of the **Advanced Calibration Options** must be adhered to or adapted. By default, a circle must be represented by an area of at least 25 pixels and a maximum of 15000 pixels. The area should not be too small, otherwise interference from the background will be picked up.

Requirements for the description file

- The pattern must be described in the form of an XML file and stored on the development computer in the following folder: *C:\Users\Public\TcVision\CalibrationPattern*
- The XML file must be validated against the TcVnCalibrationPattern.xsd schema in the *%TwinCAT3DIR%\Components\Vision\TcVision* folder.
- The description file must contain at least the DisplayName, the Origin and the 2D or 3D coordinates of the circle points.

Center point

As the center point is automatically given by the description of the points, the specification of the origin serves only as further information and is not evaluated in the calculation.

2D pattern

When specifying 2D patterns, a Z-coordinate of 0 is used internally for the calculation.

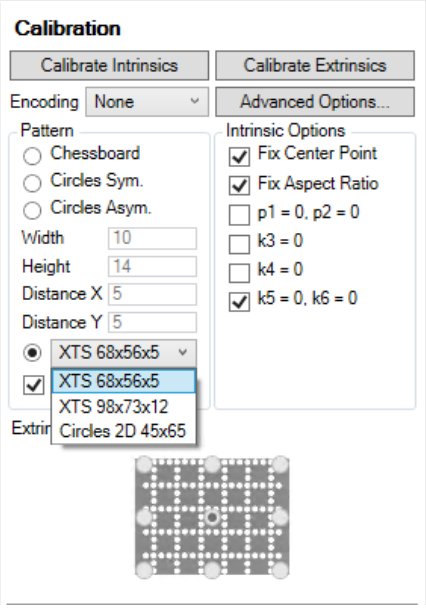
Constant offset

A constant offset can be implemented as a 3D pattern with constant Z-value.

Configuration in the Camera Calibration Assistant [▶ 107]

If valid XML description files were found in the specified folder, they are displayed with the DisplayName that was read from the file.

- **Drop-down menu for individual circle patterns**
  Selection of an individual circle pattern.
- **Color inverted**
  Checkbox for selecting color inversion of the image. By default (no check mark), black objects are expected on a white background.
- **Extrinsic Origin**
  This is automatically set to the value stored in the file for the selected pattern and cannot be changed via the user interface.
  - The optional background image for the schematic representation of the calibration pattern is displayed accordingly, if the image name including file extension was specified in the description file and the image file is located in the same folder.



Sample template of a description file for individual circle patterns

The following code block can be used as a template for creating your own description files and contains samples of completed metadata and 2D point descriptions. The 3D point descriptions, which are also included, are commented out. If 3D is required, the Pattern2D block would have to be completely replaced by the Pattern3D block, i.e. mixing is not allowed.

Assignment of the points

It is helpful if the points are sorted by rows or columns and directly match the positions of the points in the calibration image. The points can be assigned more quickly and more reliably as a result.

```
<?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 or the product TC3 GigE Vision Connector is certified according to the GigE Vision standard (version 2.0). TwinCAT Vision can thus be used with all GigE Vision-compatible cameras. As the GigE Vision standard is subdivided with each version into features that are obligatory and optional, a certification does not mean that all features of the version are supported. As cameras for their part may also have additional features that do not meet the GigE Vision standard, e.g. data compression, no statement can be made about full support. It is therefore recommended to compare and test the required features individually.



The GigE Vision standard

GigE Vision is an interface standard for industrial image processing. It facilitates configuration and operation of industrial cameras. GigE stands for Gigabit Ethernet. By using the Gigabit Ethernet communication protocol, GigE Vision benefits from the following features:

- The default transmission rate is 1 Gigabit/s (extendable with 2.5 GigE, 5 GigE and 10 GigE).
- Cable lengths up to 100 meters without the need for amplification.
- Use of standard hardware and software interfaces for Gigabit Ethernet.

The GigE Vision standard defines communication with compatible devices via UDP/IP and consists of the following four elements:

- The GigE Vision Control Protocol (GVCP) defines how GigE Vision devices are to be addressed. It specifies the data channels and mechanisms for the transfer of images and configuration data between PC and camera.
- The GigE Vision Stream Protocol (GVSP) specifies the different data types and transmission methods used to transfer images from a camera to a PC. A Packet Resend option is also available to fix transmission errors.
- The GigE Device Discovery Mechanism makes it possible to find cameras in the network using search queries.
- An XML file contains the GenAPI description that defines all public functions of the camera. This description is based on the GenICam standard.

The GenICam standard

Information on the GenICam standard and on the GenAPI description of cameras can be found on the EVMA website.

Compatibility

How to find out whether a camera is compatible with TwinCAT Vision: look to see whether the name "GigE Vision" is explicitly named in the product description or the GigE Vision logo is shown:



"GigE" versus "GigE Vision"

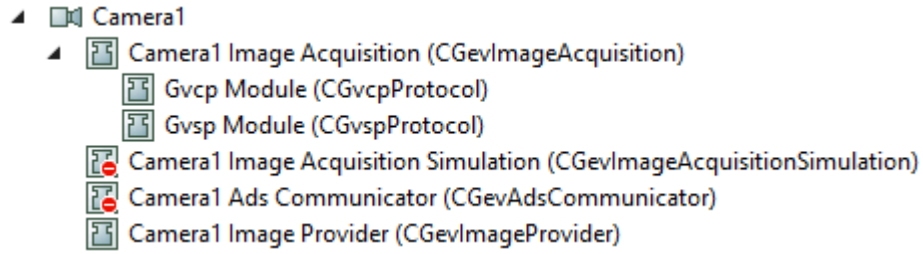
Note the difference between "GigE" and "GigE Vision"! Only GigE Vision compliant cameras are supported.

Connection to other devices is not supported

Moreover, the GigE Vision standard allows connection to other devices (such as strobe controllers). This function is not currently supported by TwinCAT Vision.

5.4.6 TcCOM objects

Each GigE Vision camera instance contains a number of TcCOM objects in the TwinCAT project:



TcCOM objects automatically managed by TwinCAT Vision

TcCOM objects are always managed automatically by TwinCAT Vision and do not have to be adapted or configured manually. Therefore the following information essentially serves to provide a more comprehensive technical understanding.

Cyclic tasks

The TcCOM objects `GevImageAcquisition`, `GevImageAcquisitionSimulation` and `GevAdsCommunicator` must be linked to a cyclic task if they are activated. The corresponding configuration can be found on the **Context** tab of the objects. Normally, there is no need to manually link tasks, as tasks required when creating a Vision device are automatically created and linked. Details on creating and linking tasks can be found in the chapter on CPU cores and tasks [▶ 61].

GigE Vision Image Acquisition

The GevImageAcquisition object coordinates the communication with a GigE Vision camera. It consolidates and manages two sub-objects that implement the two protocols of the GigE Vision [▶ 122] standard:

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

Diagnostic information on the camera communication can be found on the **Parameter (Online)** tab of the two sub-objects.

It also uses a UDP/IP stack on an appropriate network adapter to communicate with a GigE Vision camera. Diagnostic information can also be retrieved in these objects. Details on both diagnostic options can be found in the Appendix/Troubleshooting at Camera communication [▶ 2877].

The RT-Ethernet adapters and IP stacks can be found in the TwinCAT project under the I/O node:

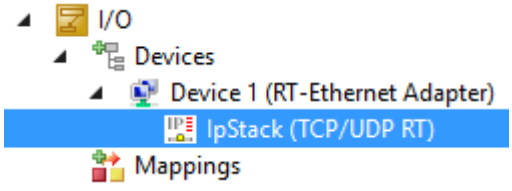


Image acquisition and the two protocols (GVCP and GVSP) are parameterized in the Configuration Assistant [▶ 75] in the configuration tree [▶ 78]. A description of the parameters can be found in Chapter TcCOM parameters [▶ 99]. This object is deactivated when camera simulation is active.

GigE Vision Image Acquisition simulation

The GevImageAcquisitionSimulation object simulates the streaming behavior of a GigE Vision camera and is deactivated by default. If the simulation mode is active, the GevImageAcquisition object is deactivated instead.

ActivateBlockIDCorrection	This setting can be used to specify that artificially generated continuous block IDs should be used instead of original IDs. Default: disabled

GigE Vision Ads Communicator

The GevAdsCommunicator object is required when recording a camera stream [▶ 101]. It sends corresponding images from the router memory to the TwinCAT Vision service via ADS. In contrast to saving images via the function block FB_VN_WriteImage, a separate TcCOM object is required for the record functionality, since the recording runs independently of the PLC processing.

In Config mode, a further instance of the GevAdsCommunicator module is used so that all Camera Assistants can communicate with the camera.

PublishImages	With this setting, the Ads Communicator sends every nth image to the TwinCAT Vision service. This setting is automatically set when activating a stream recording and depends on the **Image Offset** parameter on the Record/Playback tab [▶ 101]. It should not be set manually. Default: 0

GigE Vision Image Provider

The GevImageProvider object represents the interface between a camera and corresponding function blocks in the PLC. These include `FB_VN_GevCameraControl`, `FB_VN_SimpleCameraControl`, `FB_VN_ReadRegister` and `FB_VN_WriteRegister`, and `FB_VN_ReadMemory` and `FB_VN_WriteMemory`. To this end, the object in the Symbol Initialization tab of the project instance must be linked with the desired PLC symbols of the function blocks. If several camera instances exist in the project, ensure that the correct GevImageProvider module is selected based on the name.

ImageQueueSize	Number of acquired images that are to be buffered if they are not accepted directly by the function block in the PLC. Default: 1

Linking the TcCOM objects

The following overview graphic shows a simplified relationship of the different objects that communicate with one another or retrieve data.

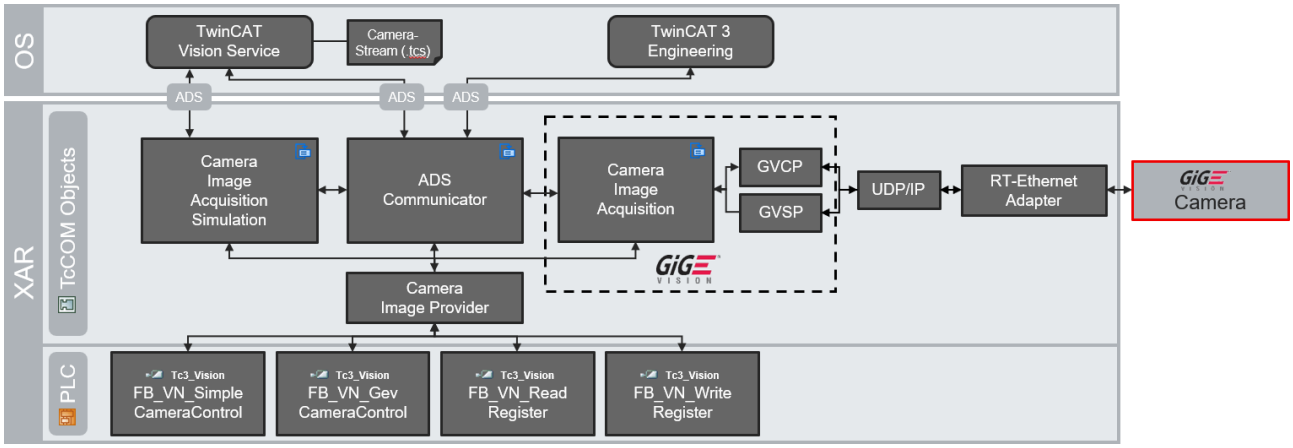


Fig. 5: Architecture of the GigE Vision Camera functionality

The linking of the TcCOM objects differs depending on the setting of the simulation mode and the Ads Communicator object (see General tab [▶ 71]). The corresponding link setting can be found in the **Interface Pointer** tab of the individual objects. By default, the link is automatically created based on the settings in the Camera Assistant and should not be changed manually. If the links were changed by mistake or the TcCOM objects were added to the camera afterwards, the links can be repaired by clicking **Try to fix** in the **General** tab.



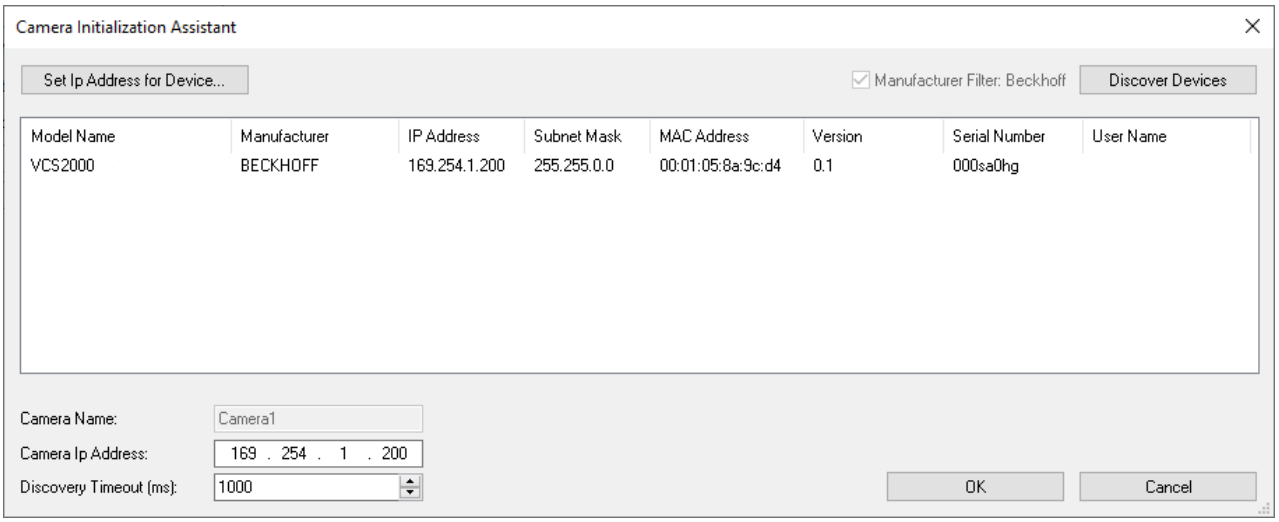
Fig. 6: Note on the invalid linking of TcCOM objects

5.5 Beckhoff Camera objects

Each Beckhoff Camera object represents a camera within a project and is identified by a different name and icon. A project can contain several camera objects. As the configuration and operation is identical to the GigE Vision Camera object [▶ 69], the detailed information can be found on the corresponding pages of this chapter.



Only cameras from Beckhoff can be linked and operated with a Beckhoff camera object. The manufacturer filter is therefore always active in the Camera Initialization Assistant and cannot be deselected by the user. When establishing a camera connection, this is also checked so that use in the Configuration Assistant and in Run Mode is only possible with cameras from Beckhoff.



5.6 File Source objects

The File Source represents an alternative image source to the camera. It enables stored images to be loaded from the file system to the TwinCAT real-time instead of live images. Note that compared to the Record/Playback [▶ 101] function of a GigE Vision Camera object, a File Source is not intended for the loading of continuous camera streams, but for the loading of individual images.

Images can be used as long as they are saved in a standard image format, e.g. bmp, jpg, png.

Compressed image formats

When using compressed image formats, it depends on whether the applied method is lossy, as the image data would then no longer correspond to the original image data of the camera. Therefore, we recommend using BMP as the storage format.

Grayscale images can be color images

Images that look like grayscale images can be saved with several channels. Pay attention to this when calling conversion functions.

The function blocks FB_VN_FileSourceControl [▶ 1613] and FB_VN_SimpleCameraControl [▶ 1646] exist so that images can be received in the PLC. Communication to the File Source Control takes place via these function blocks in a similar way to communication to a camera. An alternative to loading individual images into the TwinCAT real-time is in addition the function block FB_VN_ReadImage [▶ 1598].

Image delivery modes

The File Source Control has two modes for providing images:

1. Streaming mode: Like a streaming camera, new images are provided in a specified cycle.
2. Trigger mode: A new image is provided with a trigger signal. This can be triggered either by a software trigger or by manually actuating a button.

The settings and the image selection are done under File Source Control [▶ 126].

Creating a File Source node

Creating a File Source node is described in First Steps [▶ 27] under Creating a File Source Control [▶ 35].

Export/Import of a File Source node

If you wish to save the configuration of a File Source or import it into a project, follow the instructions in the chapter Exporting/importing configurations [▶ 135].

5.6.1 File Source Control

The configuration of the File Source Control takes place on the tab of the same name:
VISION Node > File Source Node > File Source Control tab.

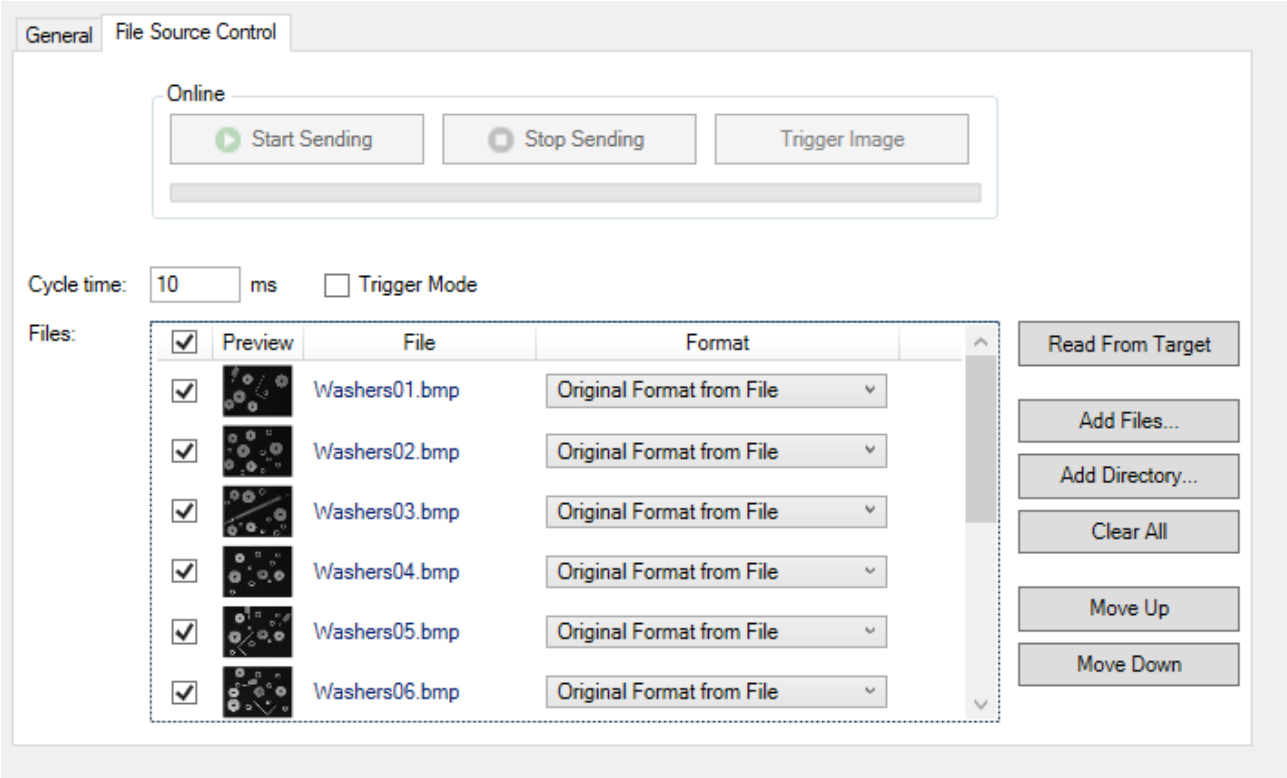


Fig. 7: File Source Control

Manage image selection

The images to be used by the File Source Control are managed via a file list and the associated controls:

Read from Target

After creating a File Source, **Read from Target** must be pressed once and the configuration activated so that images can be added.

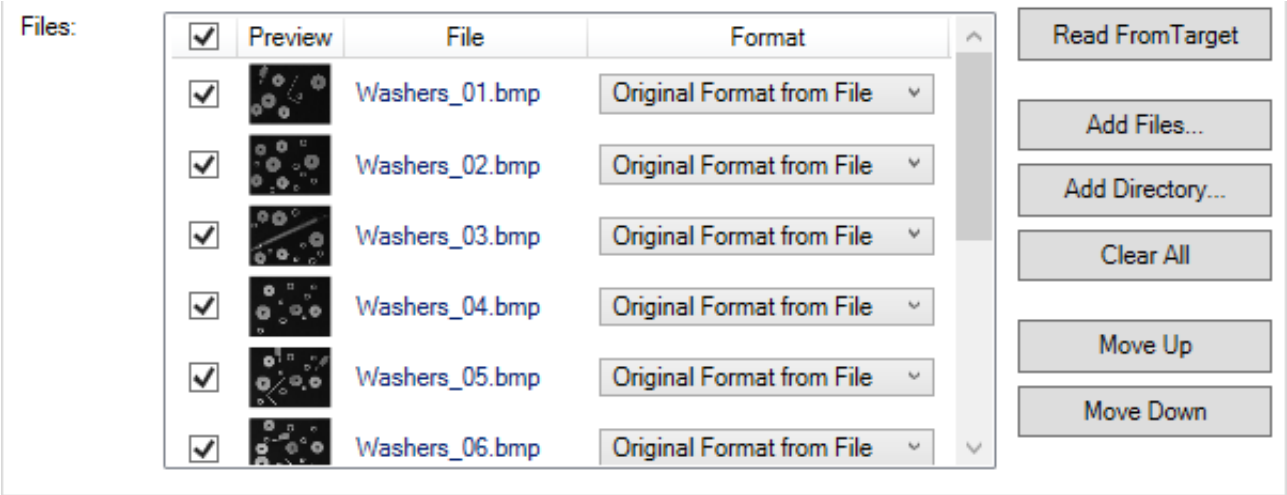


Fig. 8: Image selection in the File Source Control

Checkbox	The checkbox in the first column indicates whether an image is used or not.
Preview	The size of the preview image depends on the column width.
File	File name of the image, including data type.
Format	By default, images are sent in the saved format. Alternatively, it can be specified to convert this into a 1-channel grayscale image or 3-channel RGB image in advance and then send it. **We recommend that you explicitly specify the format for each image.**
Read From Target	The information which images are stored on the target system for the associated File Source Control is read out.
Add Files	Images from the development system can be selected via a file explorer. These are then copied to the target system in the background, so that they are available there independently of the development system. Please note that only files with unique names can be used. In addition, only image formats, i.e. not video formats, can be added here. If you want to load a camera stream (*.tcs) into the PLC, please use the Record/Playback [▶ 101] function. **Unless configured otherwise, the image files are stored on the target system in the path "C:\Users\Public\TcVision\FileSources". Make sure that the corresponding path is not affected by other software such as WriteFilter.**
Add Directory	A directory of the development system from which all images are to be used can be selected via a file explorer.
Clear All	All images will be removed from the file list.
Move Up / Move Down	A selected image is moved one position up/down in the file list.

Context menu

Use **CTRL+click** to select several images in the file list. By right-clicking on one of the selected images, you can then adjust the settings for all selected images:

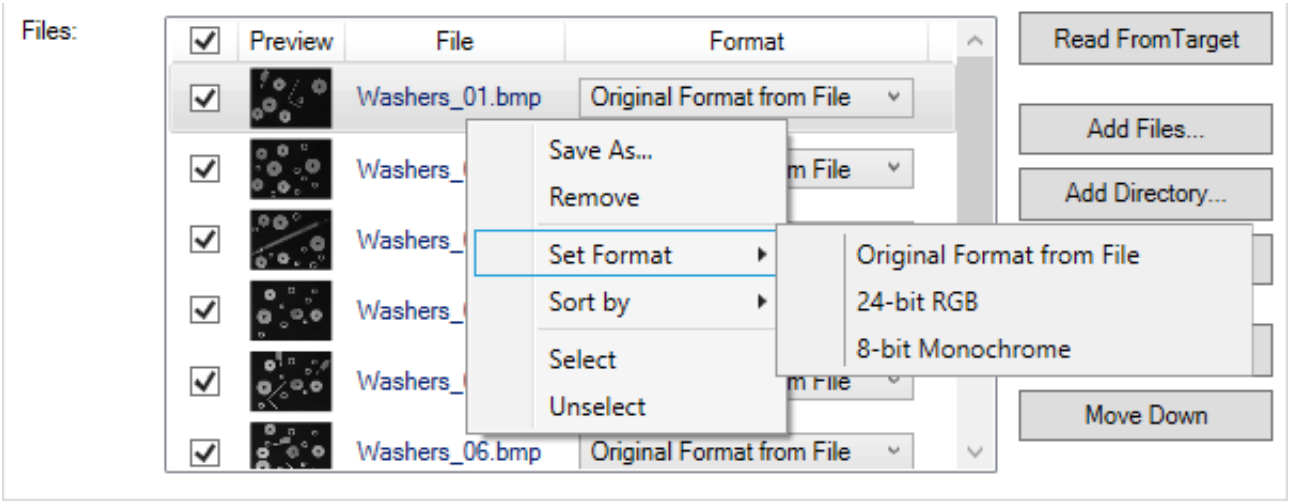


Fig. 9: Context menu for the image selection in the File Source Control

Save As...	Opens a file explorer dialog for saving the selected images.
Remove	Deletes the selected images.
Set Format	Sets the image format for all selected images: • **Original Format from File:** the format in which the image is stored is used. • **24-bit RGB:** the images are sent as 3-channel RGB images by the File Source. • **8-bit Monochrome:** the images are sent as 1-channel grayscale images by the File Source.
Sort by	Sorts all images by file list: • **Name (Ascending):** ascending by name • **Name (Descending):** descending by name
Select	Selects the highlighted images so that they can be sent via File Source Control.
Unselect	Deselects the highlighted images so that are not sent via File Source Control.

Image transfer settings

Like a streaming camera, the File Source Control can provide a new image in a specified time cycle or after calling a software trigger.

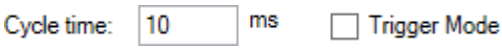


Fig. 10: Setting options in the File Source Control

Cycle Time	If no Trigger mode is selected, the images are sent to the real-time context at the Cycle Time interval. The order of the images according to the file list is taken into account. This parameter is comparable to the frame rate of a camera. **The maximum speed for the loading of images depends on the system and the images. The Cycle Time is merely the desired value.**
Trigger Mode	If Trigger Mode is selected, an image is sent after a corresponding software trigger is called. With the standard software trigger, the order of the images according to the file list is observed. Alternatively, a specific image can be triggered based on the file name.

Online

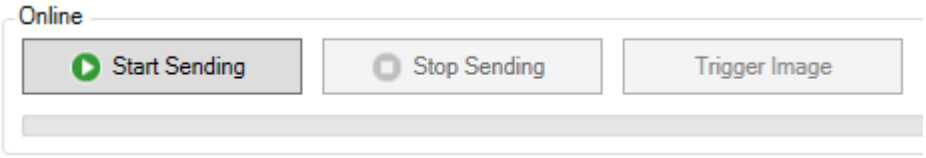
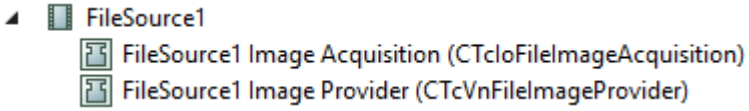


Fig. 11: Operation in the File Source Control

Start Sending	**Start Sending** in the PLC corresponds to the StartAcquisition [▶ 1617] method of the function block FB_VN_FileSourceControl [▶ 1613]. If no trigger is selected, the images are then sent via File Source Control at the **Cycle Time** interval. If the trigger is selected, File Source Control responds to a software trigger and sends the images according to a trigger signal.
Stop Sending	**Stop Sending** in the PLC corresponds to the StopAcquisition [▶ 1618] method of the function block FB_VN_FileSourceControl [▶ 1613] and terminates the sending of images via File Source Control.
Trigger Image	**Trigger Image** in the PLC corresponds to the TriggerImage [▶ 1620] method of the function block FB_VN_FileSourceControl [▶ 1613]. This can be used to trigger images manually. This button is only available if Trigger Mode is enabled and image transfer via File Source is active.

5.6.2 TcCOM objects

Each FileSource instance contains two TcCOM objects in the TwinCAT project:



TcCOM objects are automatically managed by TwinCAT Vision

TcCOM objects are always managed automatically by TwinCAT Vision and do not have to be adapted or configured manually. Therefore the following information essentially serves to provide a more comprehensive technical understanding.

File Image Acquisition

The TcIoFileImageAcquisition object coordinates the loading of images from the hard drive for FileSource instances. This object contains the following parameters (in addition to `CycleTime` and `TriggerMode`, which can be set in the File Source Control [▶ 126]):

BaseDir	File path on the target system under which images added to the selection list are stored. Standard: *C:\Users\Public\TcVision\FileSources\<UniqueId>*
ServiceTimeout	Timeout for communication with the TwinCAT Vision Service [▶ 66] in milliseconds. Default: 5000

Cyclic tasks

The File Image Acquisition object must be linked with a cyclic task. The corresponding configuration can be found in the **Context** tab of the objects. Normally, there is no need to manually link tasks, as tasks required when creating a Vision device are automatically created and linked. Details on creating and linking tasks can be found in the chapter on CPU cores and tasks [▶ 61].

File Image Provider

This object represents the interface between the FileSource object and the FileSource function block in the PLC. The ObjectID variable of the function block FB_VN_FileSourceControl [▶ 1613] or FB_VN_SimpleCameraControl [▶ 1646] must be linked with the corresponding File Image Provider TcCOM object on the **Symbol Initialization** tab of the project instance after the compilation of the PLC code. If several FileSource instances exist in the project, care must be taken on the basis of the name that the correct object is selected. This object contains the following parameters:

ImageQueueSize	Number of loaded images that are to be buffered if they are not accepted directly by the function block in the PLC. Default: 1

Linking the TcCOM objects

For a File Source device only the Image Provider has to be linked to Image Acquisition. By default, this link is created automatically and should not be changed manually. The following overview graphic shows a simplified relationship of the different objects that communicate with one another or retrieve data.

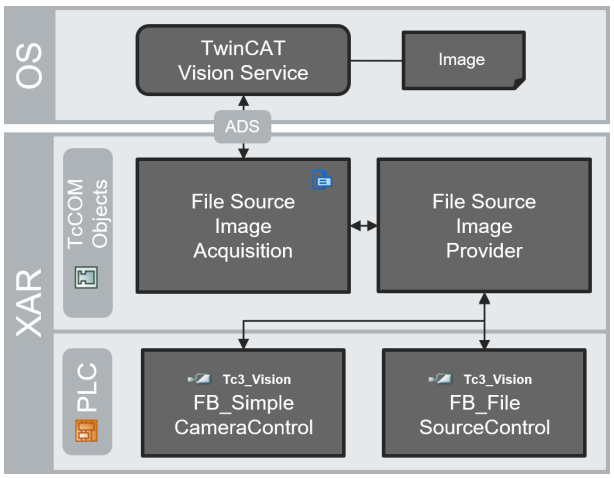


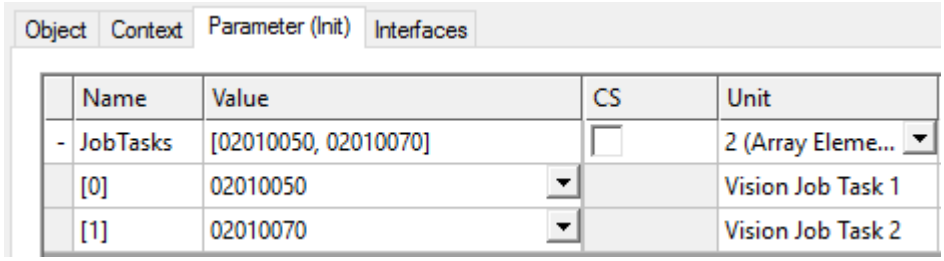
Fig. 12: Architecture of the File Source functionality

5.7 Vision Job Pool

The Vision Job Pool is used to manage Job Tasks [▶ 64] for image processing tasks. This enables the parallelized execution of selected TwinCAT Vision algorithms, which are marked in the API reference [▶ 137] by the following supplement: "Can use available TwinCAT Job Tasks for executing parallel code regions".



Clicking on the **Vision Job Pool** opens a window. The assignment of the Job Tasks takes place in this window on the **Parameter (Init)** tab. In the first row, the **Unit** column indicates how many Job Tasks are to be assigned. The array elements under **Job Tasks** can then be assigned a Job Task using the dropdown menu in the **Value** column. Each Job Task may only be assigned to one element.



NOTICE

Be sure to refer to the restrictions and recommendations [▶ 65] when creating Job Tasks.

5.8 ADS Image Watch

With the ADS Image Watch, images of the type ITcVnDisplayableImage [▶ 411] can be retrieved from the PLC and displayed in the development environment. Open the ADS Image Watch via the Visual Studio menu **TwinCAT > Windows > ADS Image Watch**.

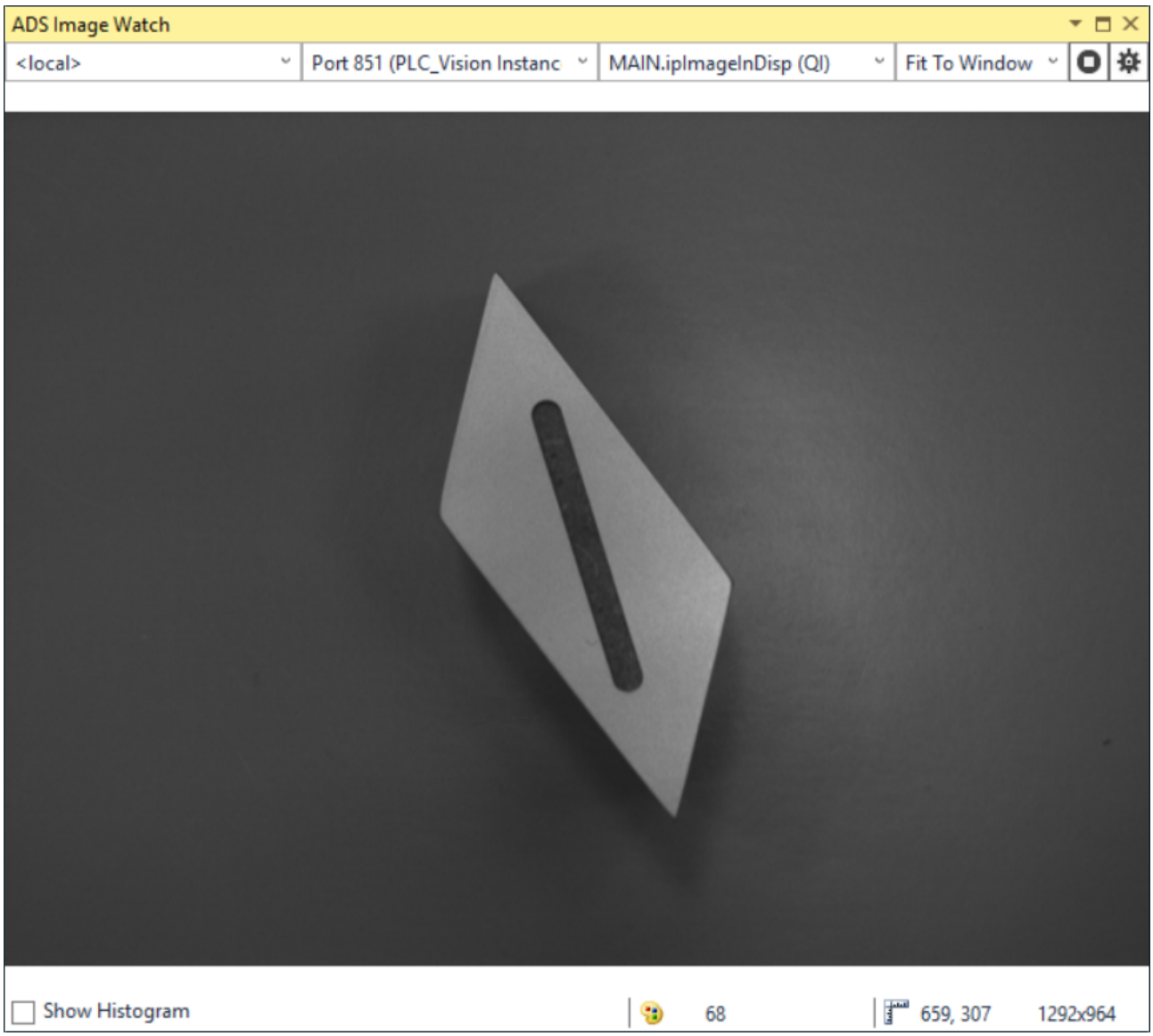
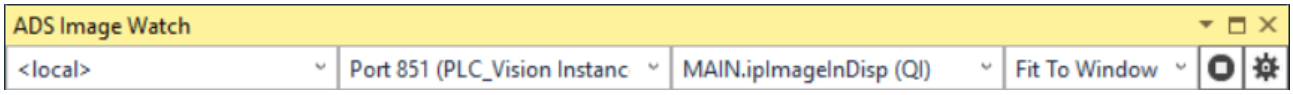


Image selection



The image selection is made using the selection fields from left to right.

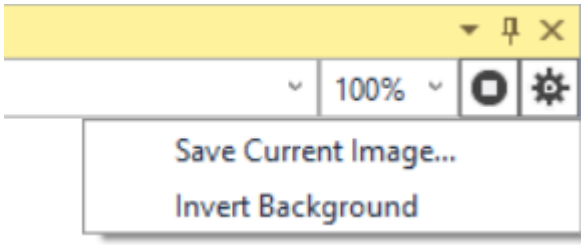
Target system	ADS devices are addressed via the ADS-AmsNetId and the ADS port. The ADS-AmsNetId unambiguously identifies the target system that must be selected first. The local system is selected by default. However, you can select any system for which a route has been created; see Choose Target System [▶ 29].
Port	The individual devices on the system are identified by ADS port numbers. A fixed port number range is defined for the PLC runtimes, so that the selection list is pre-filtered based on these. If only one PLC port exists on the target system when this is selected in the ADS Image Watch, the PLC port is selected automatically.
Image variable	All variables of type ITcVnDisplayableImage [▶ 411] are displayed here and can be selected, regardless of whether the variable points to an image with a supported image format [▶ 134] at the time.
Size	Without additional settings, the image is scaled to the size of the current window (**Fit To Window**). However, it can optionally be set to a size relative to the original image. Please note that the image is scaled down by the *PlcAux task* in the real-time environment before transmission via ADS if the selected size is smaller than the original image. If the image is then saved, it does not have the original size.
Stop / Play	The alternating Stop / Play button can be used to stop or restart the updating of the image display. If the pause icon is visible, the update is active and images are continuously received. If the play icon is visible, the update has stopped and the display of the last image is maintained. No more images are received. This makes it possible to analyze the image and every detail, e.g. by zooming in.

Image scaling

If the selected size is not 100%, the displayed image will be interpolated. For **Fit To Window** or a reduction, linear interpolation is used, and for enlargements, nearest-neighbor interpolation is used.

Depending on the image content or pixel values, deviations may occur. If individual pixels are to be analyzed in detail, the size should therefore be set to 100 %.

Options



The currently displayed image can be saved using the **Action** icon at the top right next to the image size, then **Save Current Image**.

Furthermore, the background around the display image can be changed between light and dark with **Invert Background**. This can be helpful with very bright or dark images, so that the image content or image edge can be clearly seen and distinguished by a high contrast to the background.

Image size

The image is saved in the size that the ADS Image Watch queries from the PLC. In order to save the image in the original size, the image must be set to at least 100% or the window pulled to at least the original image size. If the image size is set to larger than 100%, the image will also be saved in the original size. With a smaller setting or with **Fit to Window** and too small a window, the image will be stored in the correspondingly reduced size.

Image formats supported

The following image formats can be displayed in the following color spaces in the ADS Image Watch:

- **1-channel images** are interpreted as **grayscale images**. All images with pixel types of enum ETcVnElementType [▶ 199] are supported.
- **3-channel images** are interpreted as **RGB color images**. Images with 8 bits or 16 bits per channel are supported; this corresponds to the pixel types [▶ 199] USINT, SINT, UINT and INT.
- **4-channel images** are interpreted as **RGBA color images**. Images with 8 bits per pixel per channel are supported here; this corresponds to the pixel types [▶ 199] USINT and SINT.

Other image formats

Other image formats and color spaces can be processed in the PLC as ITcVnImage. To display them, however, they must first be converted to one of the displayable formats by means of F_VN_ConvertColorSpace before they are converted into a ITcVnDisplayableImage. If this is not done, the images will either be displayed in the wrong color representation or not at all.

Displaying 16-bit images

16-bit images are transferred from the PLC to the ADS Image Watch as 8-bit images. Therefore the pixel values are only displayed in the status bar in increments of 256 (0, 256, 512, etc.), but nevertheless relate to 16 bits: (0: Minimum, 65.280: Maximum).

Displaying (L)REAL images

(L)REAL images are transferred from the PLC to the ADS Image Watch as 8-bit images. For this purpose the value range [-1, +1] is scaled to [-127, +127] and transmitted as an integer. Floating point values outside these limits will be interpreted as -1 or +1. In the ADS Image Watch the pixel values to be transmitted are divided by 127 again so that the floating point values are displayed in the range [-1, +1].

Status Bar

The status bar below the image shows the following information (from left to right):



Fig. 13: Status bar in the ADS Image Watch

- Color values/intensity values at the mouse pointer position on the image
  or number of pixels in the corresponding container at the mouse pointer position in the histogram
- Image coordinates at the mouse pointer position on the image
  or container at the mouse pointer position in the histogram
- Image resolution

Histogram

When the **Show Histogram** checkbox is checked, the histogram of the image is displayed at the bottom left of the image. This can have several channels that are displayed superimposed, depending on the image type.

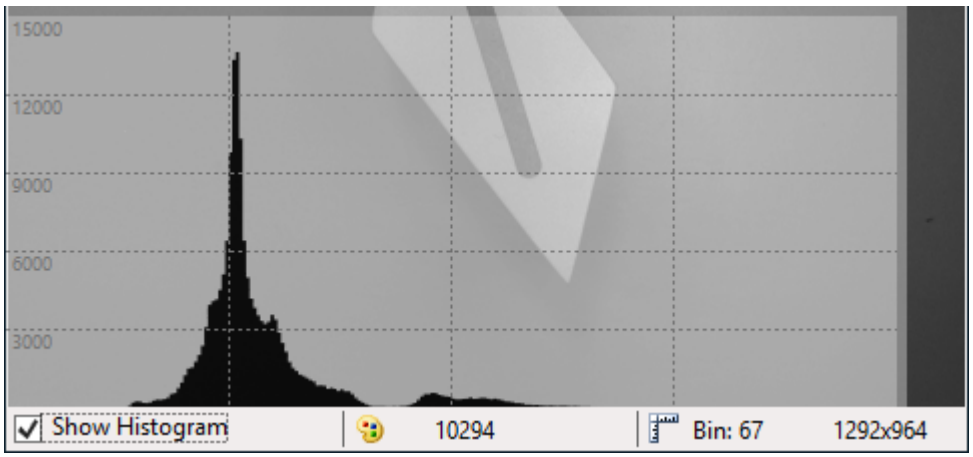


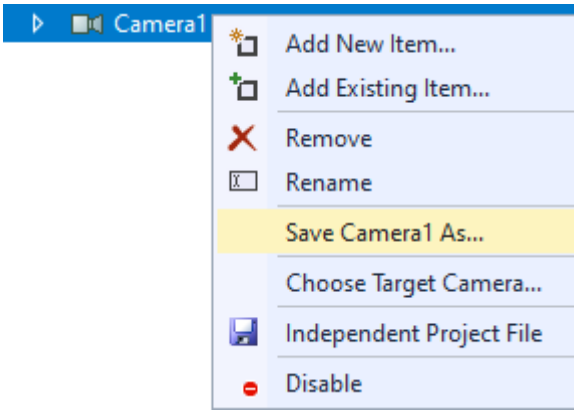
Fig. 14: Histogram in the ADS Image Watch

5.9 Exporting/Importing nodes

TwinCAT offers the option of saving individual configuration nodes such as I/O or MOTION elements in a separate XTI file or importing them from this file. This is also possible with application nodes as well as with GigE Vision Camera and File Source objects. In this way, configurations for Vision devices can be saved independently of the project and re-imported elsewhere.

Export

1. Right-click on the node to be exported and select **Save <Name> As…**

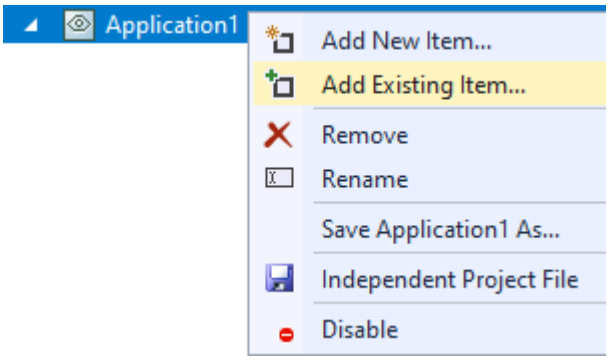


2. Select a location for the XTI file.

Import

Depending on whether you want to import an entire Application node or just a single Vision device, the following steps should be performed on the VISION node or on a corresponding Application node.

3. Right-click on the node into which you want to import the configuration and click **Add Existing Item…**



4. Select the XTI file you want to import.

5. If necessary, apply system-specific settings (e.g. network adapter) for the imported Vision devices.

Duplication of camera configurations

Please note that a successful transfer of the same configuration for several cameras can only be guaranteed if both the camera model and the firmware are identical. Otherwise problems may occur, e.g. if the register addresses of the camera are changed.

Independent project files

It is also possible to save the configuration nodes in a file of their own within the project, rather than exporting them completely. In this case the configuration for the corresponding nodes is not saved in the TwinCAT project file, but in a separate XTI file. This helps, for example, with version control systems, so that when a configuration is modified, the entire project file is not changed.

6 API reference

This chapter describes the TwinCAT Vision API, which provides options for programmatic use of the TwinCAT 3 Vision library. Since TwinCAT Vision can be used in the TwinCAT 3 PLC and in TwinCAT 3 C++, these two documentations form the basis. All TwinCAT Vision-specific descriptions are therefore divided into the following two sub-chapters.

6.1 PLC

This chapter contains the TwinCAT Vision API description for the programmatic use of the library in the PLC. As TwinCAT Vision brings along some innovations, the software concept is explicitly explained:

- Software concept [▶ 137]
- Data types [▶ 159]
  (aliases, constants, arrays, enums, structures)
- Interfaces [▶ 246]
  (images, containers, etc.)
- Functions [▶ 433]
  (image processing)
- Function blocks [▶ 1570]
  (communication with cameras and the file system, image processing)

6.1.1 Software concept

This chapter describes the innovations or special features that TwinCAT Vision brings along. The principles of the PLC programming in TwinCAT 3 as well as the reference programming can be found in the general TwinCAT 3 documentation. Within the description of the PLC you will find a chapter with TwinCAT 3 programming conventions. The recommendations and notes as well as the advantages and disadvantages that it contains enable a standardized program structure, a consistent naming of objects, variables, instances and an easily readable and more understandable code to be achieved and thus the simplification of the development, use and maintenance of the programs.

Unambiguous access to library modules

If the Tc3_Vision library is to be included (referenced) in another library and can thus occur several times in a project, an unambiguous access must be made via the namespace as prefix of the identifier. Default e.g. `hr := Tc3_Vision.F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);`

For more information and setting options, see the PLC project settings and the library properties.

⚠ WARNING

Access to the same data when using several tasks

If several tasks are to access the same image, for example, these accesses must be synchronized according to these guidelines. Otherwise there is the danger of an inconsistent data set, which could lead to unforeseeable consequences and program malfunctions.

6.1.1.1 Interface Pointers

Dynamic memory areas are required for TwinCAT Vision in order, for example, to save images and containers. These images and containers are represented by variables of type ITcVnImage [▶ 411] and ITcVnContainer [▶ 370]. Since they are interfaces pointing to a dynamic memory area, these variables are called interface pointers.

With the TwinCAT Vision API functions, the associated memory management is automatically controlled within the functions; nothing needs to be observed here.

For example, a new image is created (800 x 600 pixels of the type `TCVN_ET_USINT` with one channel):

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

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

this image is now to be converted into an RGB image with 3 channels. Therefore 3 times as much memory is required. The following function call is sufficient for this:

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

Internally it frees up the previous memory area and allocates new memory in a size to suit. Interface pointers are thus a simple way of managing objects with dynamically required memory in the PLC.

⚠ WARNING

Consequences of misuse

If you do not follow the following instructions for the use of interface pointers, this can lead to **complete system crashes** and to **memory leaks**!

Reference counter

As soon as the variables are no longer exclusively used as function parameters for the TwinCAT Vision API functions, however, it is necessary to familiarize yourself more precisely with the concept of interface pointers.

It is important to know that the interface pointer variable is only a pointer to an object in memory. If you assign one variable to another, only the pointer to this memory area is copied, not the data itself.

This means that if the data in one of the two variables is changed, it is also automatically changed in the other.
If the memory in one of the variables is released (as in the above example F_VN_ConvertColorSpace [▶ 1241]) and the second variable still has its pointer to this memory area, accessing this already released memory tends to result in a system crash.

For this reason, the objects in the memory all have a reference counter. As soon as a variable is assigned to another, this reference counter must be increased. If a new pointer is assigned to a variable, the reference counter must be decreased again.

When the reference counter reaches 0, the memory is released. The corresponding data are thus completely deleted.

In order to indicate whether a pointer variable points to a valid memory area, this pointer must always be set to 0 if it is no longer in use (and the reference counter was thus reduced). By default, all created variables (and thus the pointers too) in the PLC are initially set to 0 when the program starts.

Management of interface pointers

In the TwinCAT Vision API functions (marked by F_VN_...), the reference counter is automatically managed by interface pointers. Thus, nothing special needs to be observed when using them. However, if interface pointers are manually copied or used, for example, as function parameters, manual management is necessary (see below).

Checking the validity of interface pointers

Since the value 0 for an interface pointer basically corresponds to just one invalid memory area – and access to it also leads to a critical system error – it is important to check before each use whether the value is non-zero.

In the TwinCAT Vision API functions (`F_VN_...`), this check and the corresponding handling take place internally and automatically. However, this is not possible with interface methods, such as `TcRelease()` in the following sample, since their call already leads to an error if the variable is 0. The variable must therefore be checked explicitly beforehand.

⚠ WARNING

Method access to invalid memory area leads to system crash

An internal check is not possible for interface methods, since the corresponding objects must exist to call the method.

Even outside the method call, make sure that the corresponding object exists (the interface pointer should be non-zero).

In addition, a system crash does not necessarily always occur. If the invalid memory area coincidentally remains unused, a system crash may not occur.

Release of an interface pointer

The release of an interface pointer variable consists of checking whether this is already 0 and otherwise reducing the reference counter and setting the variable to 0. As this is a frequent use case, a function `FW_SafeRelease` is provided that carries out all necessary steps with a single call:

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

The internal mode of operation of the `FW_SafeRelease` function is illustrated by the following code snippet:

```
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
```

Copying of an interface pointer

When copying an interface pointer, its reference counter must be manually incremented. A direct assignment by means of the `:=` operator, as well as function/method/function block calls in which an interface pointer is transferred as an input, output or return value, count as a copying procedure. A transfer by means of I/O image also counts. The following method of incrementing the reference counter is available:

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

In addition, it must be ensured that the interface pointer to be written is enabled before copying. Otherwise memory leaks will occur. A manual procedure for copying an interface pointer could look like the following, for example:

```
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
```

Please note that only the interface pointer, not the underlying image is copied.

Rules of thumb

Stick to the following rules of thumb when handling interface pointers.

It is irrelevant whether the interface pointers are (displayable) images, containers, iterators, etc. – they are mutually interchangeable in the samples.

If you call a method on an interface pointer

It must be ensured that the interface pointer is valid
(check for `<> 0`):

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

In general it is recommended to use the alternative `F_VN_` function (if it exists) instead of an interface method, because the check will then be automatically performed internally:

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

If you carry out the check and the call of the interface method in the same expression, you must use `AND_THEN` instead of `AND`. Only in that case will the rear part be executed if the front part returns `TRUE`:

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

Manually copy an interface pointer

Make sure beforehand that the interface pointer to be written has been enabled. Increment the reference counter of the interface pointer after copying:

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

Declare interface pointer in VAR_INPUT

Nothing needs to be observed for functions and methods. For programs and function blocks, the pointer in the variables is retained after the call and should therefore be set to 0 at the end of the POU:

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

This ensures that the memory area can no longer be accessed the next time it is called, since it could have already been released outside. Increasing the reference counter within the POU is not a solution, since a new assignment overwrites the previous pointer and therefore cannot be released, resulting in a memory leak. If the interface pointer is to be kept, it should be copied to a local variable.

Declare interface pointer as reference in VAR

Release the interface pointer at the end of functions and methods. The interface pointer no longer exists after ending the POU; therefore the reference counter must be decremented beforehand. For programs and function blocks, the interface pointer should be released as soon as it is no longer required:

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

Not recommended - declare interface pointer in VAR_OUTPUT

The declaration of an interface pointer in VAR_OUTPUT is not recommended, since the use of outputs is optional and thus memory leaks can occur.
In programs and function blocks, the pointer in the variables is retained even after the call, so that in the event of changes outside, a released memory area can be accessed in the function block.
To prevent this, you must always ensure that the outputs are used and, in the case of programs and function blocks, that the reference counter is incremented or the output variable is set to 0 after the output variable has been copied.
This is very laborious and error-prone, since you have to take care yourself to implement correctly at every point in the code. Therefore, transferring it as a reference in VAR_INPUT is recommended instead.

NOTICE

Memory leaks

The use of an interface pointer in VAR_OUTPUT is not recommended. Failure to observe the above points may result in memory leaks and system crashes.

Declare interface pointer as reference in VAR_INPUT

The transfer as a reference should be used if a change to the interface pointer (e.g. assigning a new image/container) is to be passed to the outside within the POU for further use there.

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

The sample Self-written functions [▶ 2830] explains the correct use of interface pointers in self-written functions and methods.

Consequences of misuse

If an interface pointer variable is copied, but the reference counter is not adjusted at the same time with `TcAddRef()`, there are two pointers; however, only one of them is known to the program. Therefore, the data to which both interface pointer variables point will then be deleted as soon as one of the two variables is released. The release can take place both by a manually executed `FW_SafeRelease` and by a TwinCAT Vision API function that rewrites the variable. If the remaining interface pointer variable is then accessed by a method call, the system **crashes** as described above.

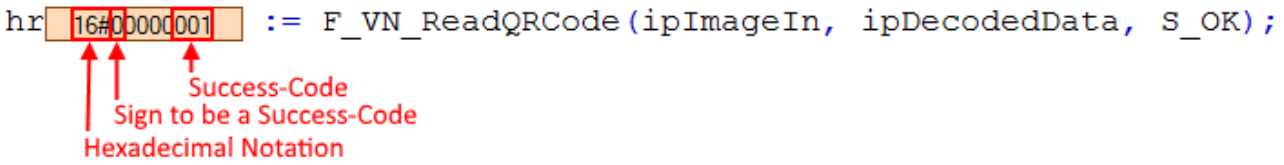
If, contrary to this, interface pointer variables are rewritten without prior release (e.g. through a `:=` assignment), this leads to **memory leaks**. This can occur, for example, if interface pointers are used as method variables and not released before ending the method. An indication for memory leaks can be found in the information window of the AMS router (see Router memory [▶ 54]). If the memory marked there as available decreases over time, this indicates a memory leak.

6.1.1.2 HRESULT

All TwinCAT Vision functions return an HRESULT after their execution. Its value indicates whether the execution was successful or not.

SUCCESS codes

A successful execution is indicated by a POSITVE code. In hexadecimal notation the first digit is between 0 and 7.



Frequent SUCCESS codes are:

Code	Name	Description
16#000	S_OK	Function was executed successfully
16#001	S_FALSE	Function was executed successfully but did not reach a complete result. (Occurs, for example, with the Code Reading functions if no code was found in the image.)
16#203	S_PENDING	Asynchronous method was started but there is no result yet (occurs at the first call of fbCameraControl.StartAcquisition(), for example; at the second call S_OK is returned, for example)
16#256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog.

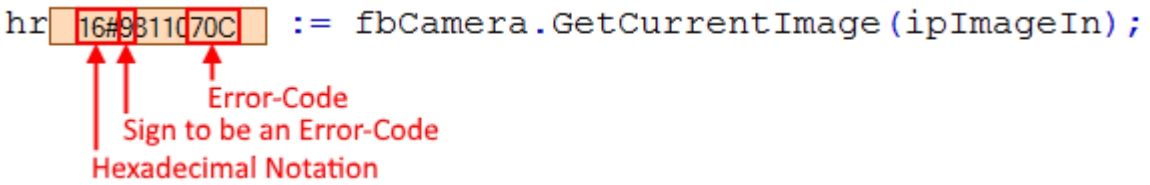
ERROR codes

A failed execution is indicated by a NEGATIVE code, or in hexadecimal notation the first digit is >= 8. The final digits of the code correspond to the ADS Return Codes [▶ 2879].

NOTICE

Error code

If an error code is returned, it means that all returns of the function are invalid. and must therefore not be used.



Common ERROR codes with the TwinCAT Vision API elements are:

Hex	Dec	Name	Description
16#70A	1802	NOMEMORY	Insufficient memory
16#70B	1803	INVALIDPARM	Invalid parameter values
16#70C	1804	NOTFOUND	Not found (file, image, ...)
16#70E	1806	INCOMPATIBLE	Objects do not match
16#712	1810	INVALIDSTATE	The method of an FB was called in an impermissible state.
16#719	1817	TIMEOUT	timeout
16#71A	1818	NOINTERFACE	Interface query failed
16#71B	1819	INVALIDINTERFACE	Wrong interface requested.
16#71D	1821	INVALIDOBJID	Object ID is invalid.
16#734	1844	OUTOFRANGE	Outside the valid range.

Please refer to the ADS Return Codes [▶ 2879] for the complete list.

Application

In the event of an error, the `HRESULT` is negative (error code), otherwise it is positive (success code). Use the `FAILED ()` and `SUCCEEDED ()` functions to check this.

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

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

Restricted query

Note that the above query only checks for successful execution of the function, not for a correct or desired result.

For example, Code Reading functions [▶ 837] return the success code `S_FALSE` (16#001) if no code is found or read in the image. To detect this case, `hr` must be directly compared with `S_OK` or `S_FALSE`. In contrast, measurement functions [▶ 1481] return `NOTFOUND` (16#70C), if the object to be measured is missing from the image.

Most TwinCAT Vision functions expect a `HRESULT` as the last input parameter and are only executed if this is a SUCCESS code and it is therefore confirmed that no error has occurred so far. Otherwise, the respective function is not executed and instead returns the previous `HRESULT`. This ensures that, when the `HRESULT` is passed from function to function, the first error code that occurred is output instead of a subsequent error.

SUCCESS codes

SUCCESS codes (e.g. `S_FALSE`) are not forwarded and overwritten by the next function. If necessary, the HRESULT value of the respective functions must be checked explicitly.

Accessing interface methods

If a function is to create an interface pointer, the HRESULT of the function indicates whether the interface pointer now exists. In addition, the interface pointer must be checked to ascertain whether it is zero.

⚠ WARNING

Method access to invalid memory area leads to system crash

An internal check is not possible for interface methods, since the corresponding objects must exist to call the method.

Even outside the method call, make sure that the corresponding object exists (interface pointer not 0) and that the program has not yet returned an error code.

This is a secure query to use a newly created interface pointer:

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

Integration into user-defined function blocks

An HRESULT can be integrated into user-defined function blocks as follows:

```
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
```

This is explained in more detail in the sample Self-written functions [▶ 2830].

Extraction of the ADS Return Code

If, for example, the pure ADS Return Code is required for display in an HMI, this can be extracted from the HRESULT in numeric and textual form as follows:

```
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);
```

The library Tc2_Utilities is required for this purpose.

6.1.1.3 Machine Learning

In Machine Learning, a model is optimized to solve a task. This process is known as training. Sophisticated data analyses can be carried out automatically with the help of trained models. This means that complex, manually created program constructs can be replaced. The Machine Learning models are created, trained and executed in the TwinCAT 3 real-time environment using the Machine Learning functions [▶ 1384] provided. The tasks that can be solved with it are diverse and can be roughly divided into five task areas (classification, anomaly detection, regression, cluster analysis and feature transformation). Depending on the task, different model types are available, each of which has general or task-specific advantages and disadvantages.

In Machine Learning, a distinction is made between supervised and unsupervised learning. Supervised learning requires additional information such as annotated (e.g. labeled) data sets so that the model can learn a mapping to known categories (classes) or to desired target variables. The model can then make a prediction of the categories or target variables for the unknown data. In contrast, unsupervised learning without additional information is based solely on the features themselves.

All classification and regression models can generally be assigned to supervised learning. In contrast, all models for anomaly detection and clustering are assigned to unsupervised learning. Both options are available for feature transformation, with the Linear Discriminant Analysis method based on supervised learning and Principal Component Analysis based on unsupervised learning.

The following assignment of model types to task areas helps with the choice of suitable models. It should be noted that the handling of machine learning in TwinCAT Vision is shown here specifically and not in general.

Model types	Classification	Anomaly detection	Regression	Clustering	Feature 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				

Only models that were created using one of the `F_VN_Create` functions from the Machine Learning [▶ 1384] group can be used. The models created and subsequently trained in this way can be saved or loaded to the hard disk via the function blocks FB_VN_WriteMlModel [▶ 1607] and FB_VN_ReadMlModel [▶ 1600]. The import of externally trained models, e.g. as ONNX files, is not intended.

Basic procedure for machine learning:

- Analyze the problem and define the desired result:
  - Among other things, this determines the area of application and the model types that can be used.
- Collecting data:
  - Training, validation and test data
  - The data must contain information that is suitable for solving the problem.
  - The data must cover all variants and characteristics that may occur later.
  - With supervised learning, the data must also be annotated.
- Selection and extraction of suitable features:
  - Which features are unique, special or provide relevant information for differentiation?
  - Redundant features should be avoided.
  - A feature transformation can be helpful here.
  - Finally, a feature normalization should be carried out so that the value ranges of the different features are comparable and the requirements of the model are met.
- Determination of a suitable model and the hyperparameters (validation and optimization). (Training of the selected model with the determined hyperparameters on the training and validation data).
- Final test with previously unused, independent data.

The data must be available in the form of samples. A sample is a vector consisting of features. The features can be both tangible variables of the associated image (e.g. color values) or contour values (e.g. in the form of the center of gravity) etc., as well as abstract variables (e.g. the result of a PCA), which describe the

image statistically in a meaningful way, but which cannot be interpreted directly by the user. For the training of supervised models, additional information such as a class assignment is required for each sample. A collection of several samples is referred to as a batch.

Depending on the model type, functions for sample and / or batch training and prediction are available. Some models can be retrained in order to further optimize the model during runtime or to adapt it to new conditions.

Feature extraction

Feature extraction involves the extraction of numerical features from image data and is an essential component of a Machine Learning solution. A distinction must be made between whether a feature can be determined directly by a function, whether a combination of functions is required, or whether logical features of an object, such as the number of holes, can be used for differentiation. You will find an exemplary list in the chapter Feature extraction functions [▶ 145].

Feature normalization

The scaling of features for normalization is primarily performed to ensure that all features contribute equally to the learning process of the model. Features with larger values can dominate distance calculations, making distance-based models more sensitive to these features. To avoid this and to meet the requirements of the model, we generally recommend performing a feature normalization.

The function F_VN_GetFeatureScales [▶ 1443] provides three scaling options (ETcVnFeatureScalingType [▶ 202]). You can also create and use your own scaling. The scaling is applied using the function F_VN_FeatureScaling [▶ 1431].

Same feature extraction and normalization

It should be noted that all processing steps must be carried out both during training and when using a model. The functions must therefore be executed in both places and, if necessary, parameters must be saved for later use.

6.1.1.3.1 Feature extraction functions

This list contains functions that are suitable for feature extraction. This list is only an example and does not claim to be exhaustive.

Image features

- F_VN_ImageAverage [▶ 1218]
- F_VN_ImageAverageStdDev [▶ 1219]
- F_VN_ImageMoments [▶ 1226]
  - F_VN_HuMomentInvariants [▶ 1538]
- F_VN_CountNonZeroPixels [▶ 1212]
- F_VN_HistogramExp [▶ 1247]
- F_VN_ImageCenterOfMass [▶ 1221]
- F_VN_MaxPixelValue [▶ 1227]
- F_VN_MinPixelValue [▶ 1229]
- F_VN_RegionOrientation [▶ 1231]

Contour features

- F_VN_ContourArea [▶ 958]
- F_VN_ContourCircularity [▶ 960]
- F_VN_ContourRoundness [▶ 970]
- F_VN_ContourMoments [▶ 966]
  - F_VN_HuMomentInvariants [▶ 1538]

- F_VN_ContourConvexity [▶ 961]
- F_VN_ContourPerimeter [▶ 969]
- F_VN_ContourOrientation [▶ 967]
- F_VN_ContourCenterOfMass [▶ 959]
- F_VN_ContourEccentricity [▶ 962]
- F_VN_ContourElongation [▶ 963]
- F_VN_ContourExtremePoint [▶ 964]
- F_VN_ContourInertiaRatio [▶ 965]
- F_VN_ConvexHullPoints [▶ 971]
- F_VN_ConvexityDefects [▶ 973]
- F_VN_FourierDescriptors [▶ 982]
- F_VN_EnclosingCircle [▶ 976]
- F_VN_FitEllipse [▶ 979]
- F_VN_EnclosingRectangle [▶ 977]

Image texture

- F_VN_Granulometry [▶ 1554]
- F_VN_HaralickFeatures [▶ 1556]

Key Points

- F_VN_KeyPointsAGAST [▶ 1357]
- F_VN_KeyPointsAndDescriptorsAKAZE [▶ 1359]
- F_VN_KeyPointsAndDescriptorsORB [▶ 1364]
- F_VN_KeyPointsFAST [▶ 1366]
- F_VN_KeyPointsGFTT [▶ 1368]
- F_VN_KeyPointsSB [▶ 1372]

6.1.1.4 Watchdogs

Since the runtime of many image processing algorithms depends on the image content, function calls can take much longer than usual under unfavorable circumstances.

This can be caused, for example, by changing lighting conditions, unexpected objects in the image and other factors that can drastically influence the calculation time of functions such as F_VN_FindContours() [▶ 1187]. If, for example, only 10 contours can be found under normal conditions, with other lighting conditions there may be 100 or more.

This can lead to cycle overruns, which must be avoided at all costs, as they lead to undefined behavior.

Watchdogs are helpful for this purpose. Watchdogs can abort individual Vision functions or entire code sections with several functions after a certain time. Some TwinCAT Vision functions can be aborted during execution and return the partial results calculated up to that point. If several functions are monitored by a watchdog, all remaining functions within the watchdog range are skipped when the specified time has elapsed. However, the subsequent functions (after the watchdog area) are executed normally again.

There are two functions available to define the start of a watchdog-monitored area, which differ in terms of when the specified time starts to count:

- F_VN_StartAbsWatchdog [▶ 990] defines an absolute abort time relative to the start of the task cycle
- F_VN_StartRelWatchdog [▶ 991] defines a relative abort time relative to the start of the watchdog area

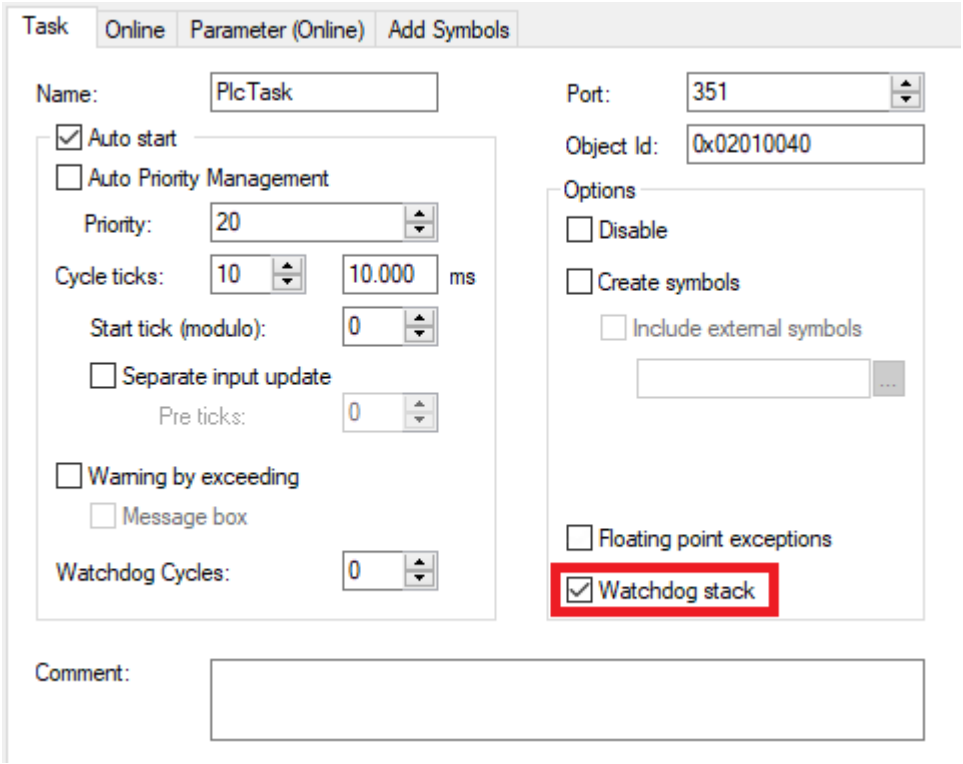
In both cases, the end of the area is defined by the function F_VN_StopWatchdog [▶ 993], which optionally returns the handling component and the required time.

NOTICE

Watchdog with relative abort time

When using `F_VN_StartRelWatchdog`, care must be taken that the abort time is not set too long, as otherwise cycle overshoots can continue to occur. Therefore, observe the cycle time of the corresponding task, the time already elapsed up to the call of the function and the time still required until the end of the task after `F_VN_StopWatchdog`.

In order for functions to be monitored by a watchdog, the option "Watchdog stack" must be enabled on the respective executing task.



Sample

In the sample below, a watchdog is started with a stop time of 10 ms relative to the call of the watchdog function. If, for example, the cycle time is 20 ms, 4 ms have already passed in the current task cycle when F_VN_StartRelWatchdog [▶ 991] is called and the watchdog is started with a stop time tStop of 10 ms relative to the current time, then the watchdog stops the monitored functions (i.e. any function called between the start and stop of the watchdog) after 14 ms of the current cycle have elapsed.

```
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);
```

Now two situations are possible:

- Both functions are terminated in time – this is the normal case.

Suppose F_VN_Threshold [▶ 1334] takes 1 ms and F_VN_FindContours [▶ 1187] takes 5 ms. When F_VN_StopWatchdog [▶ 993] is called, 4 ms remain. The watchdog runtime information is:

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

- The watchdog must intervene.

Scenario: The function F_VN_Threshold [▶ 1334] is not dependent on the image content (only on the number of pixels) and therefore only takes 1 ms, but the lighting conditions have changed unfavorably, so that F_VN_FindContours would take longer than 9 ms. Therefore the watchdog stops F_VN_FindContours [▶ 1187], which nevertheless returns the contours found so far. The watchdog runtime information could then be as follows:

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

In this case, an estimated handling percentage of 70% was calculated when the function was aborted. The remaining time is negative, i.e. the planned stop time was exceeded by 50 us, so that after the function F_VN_StopWatchdog [▶ 993] 14050 us have passed since the start of the task cycle, instead of the planned 14000 us. This overrun is due to the fact that the partial results of the function that have already been calculated are to continue to be used. Therefore, on the one hand the algorithm can only be aborted at specific points, on the other hand the previous results must be organized and returned. The maximum overrun generally depends on the specific function and the image content. In a program, the termination time should therefore always be selected such that a safety buffer remains at the end of the task cycle.

Monitored functions

The following functions provide partial results in the event of a watchdog termination:

- F_VN_AdaptiveThreshold(Exp) [▶ 1327]
- F_VN_ConvertColorSpace [▶ 1241]
- F_VN_DetectBlobs(Exp) [▶ 1178]
- F_VN_FindContourHierarchyExp [▶ 1183]
- F_VN_FindContours(Exp) [▶ 1187]
- F_VN_HoughCircle(Exp) [▶ 1192]
- F_VN_LocateCircularArc(Exp) [▶ 1489]
- F_VN_LocateEdge(Exp) [▶ 1495]
- F_VN_LocateEdges(Exp) [▶ 1501]
- F_VN_LocateEllipse(Exp) [▶ 1507]
- F_VN_MatchTemplate(Exp) [▶ 1199]
- F_VN_MatchTemplateAndEvaluate(Exp) [▶ 1200]
- F_VN_MeasureAngleBetweenEdges(Exp) [▶ 1513]
- F_VN_MeasureEdgeDistance(Exp) [▶ 1518]
- F_VN_MeasureMinEdgeDistance(Exp) [▶ 1523]
- F_VN_NormalizeImageForDisplay [▶ 1253]
- F_VN_ReadDataMatrixCode(Exp) [▶ 845]
- F_VN_ReferenceColorSimilarity(Exp)_ITcVnColorModel (_ITcVnMlModel) [▶ 1254]
- F_VN_ReferenceColorSimilarity(Exp)_TcVnVector3_LREAL [▶ 1256]

- F_VN_ResizeImage [▶ 1159]
- F_VN_Threshold [▶ 1334]
- F_VN_TrainImageColor(Exp(2)) [▶ 1261]
- F_VN_WarpAffine(Exp) [▶ 1165]
- F_VN_WarpPerspective(Exp) [▶ 1170]

Other functions cannot be aborted during the execution, but will be skipped if the watchdog time has expired.

Examples

- Blob Detection with watchdog monitoring [▶ 2780]

6.1.1.5 Image processing

In the following chapters, some concepts of the image processing are explained both generally and specifically with reference to TwinCAT Vision. They will help you to understand the mode of operation of many API functions [▶ 433].

6.1.1.5.1 Images

Images are handled in TwinCAT Vision by image objects, which could have implemented various interfaces such as ITcVnImage [▶ 411]. Access to the actual image data of the image object is gained via an interface pointer, which is then used in the PLC for processing. The images can consist of various pixel types and a number of channels, and may have a virtually arbitrary size. Due to processing from the real-time and the dynamic size, the image data are located in the router memory. Camera images are stored there directly after receipt, whereas images from the hard disk must first be loaded via a File Source or PLC function block and end up in the router memory in this way. The assistants in the camera object do not work with images from the router memory.

Querying an image interface of a Vision device

In many cases images are created with a camera or by loading an image file. The following function blocks are available for this purpose:

- FB_VN_SimpleCameraControl [▶ 1646]
  Can be connected to a GigE Vision camera instance or a File Source instance.
- FB_VN_GevCameraControl [▶ 1624]
  Is connected to a GigE Vision camera instance.
- FB_VN_FileSourceControl [▶ 1613]
  Is connected to a File Source instance.

Depending on the setting of the Vision device, a startup process or an image trigger may be necessary. Only the querying of the interface pointer in the PLC, via which access is gained to the image in the memory, is displayed here:

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

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

Image formats

The two most commonly used image formats are mono and RGB. In the camera, the transferred image format is set under the `PixelFormat` tab. In order to reduce the amount of data and thus be able to realize higher transfer rates, there are also packed or filtered image formats, such as the Bayer format. Since these transmitted image formats cannot be used or do not correspond to the original image due to the filter, the formats should be converted directly after receipt.

- In the case of packed images, the conversion happens internally when you call `GetCurrentImage()` and you get the unpacked image directly.
- In all other cases, like with the Bayer format, the image should be converted first with the F_VN_ConvertColorSpace [▶ 1241]. After that, the input image is no longer needed and can be discarded.

This point is especially important to note when using functions that interpolate image information, such as F_VN_CompensateLensDistortion [▶ 1092], F_VN_WarpAffine [▶ 1165], F_VN_RotateImageExp [▶ 1162] or such as the functions for calibration that are disturbed by the filter image. Therefore, it is generally recommended to first read the image into the PLC, convert it there, and then apply all subsequent functions to the original image.

Creating images

There are situations where an image is needed that has not been captured by a camera or loaded from the file system. For example, a color image might be created to display results. The function F_VN_CreateImage [▶ 784] can be used for this purpose by specifying the image size, pixel type and number of channels. The pixel type `TCVN_ET_USINT` selected here corresponds to a usual image with 8 bits per channel. Please note that the memory required for the image is only allocated, but not initialized. Therefore the image can be set to a uniform color with the function F_VN_SetPixels [▶ 806].

Maximum image size

The maximum image size that can be created, received, or processed is limited by the number of 2^20 pixels for rows or columns, or by the total size of 2^30 pixels for an image, whichever comes first.

```
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);
```

Alternatively, an image can be created from existing image data using the F_VN_CreateImageFromArray [▶ 786] function.

```
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
);
```

Accessing image data

The functions F_VN_GetPixel [▶ 799] and F_VN_SetPixel [▶ 805] can be used to access individual pixels of an image. This method is easy to use. However, it has a relatively long runtime when applied repeatedly to many pixels. In this case, direct access to the image data array is more appropriate. A pointer for a complete row of pixels is fetched and the array operator is used to access the individual pixels:

```
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
```

Displayable images

In order to be able to transfer and display an image by ADS (e.g. in the ADS Image Watch [▶ 131]), it must first be converted into a displayable image of the type ITcVnDisplayableImage [▶ 411].

The strict distinction between displaying and processing images is made for the following reason: Image transmission via ADS is not synchronous with the PLC cycle. Consequently, image processing functions could be performed at the same time as image transfer. In order to avoid memory conflicts during image transfer, it is therefore necessary to separate the memory areas from the images to be displayed and processed.

The two interfaces ITcVnImage [▶ 411] and ITcVnDisplayableImage [▶ 411] prevent image access for unwanted purposes (either processing or transferring). The following functions are available for conversion into a displayable image (ITcVnDisplayableImage [▶ 411]):

- F_VN_CopyIntoDisplayableImage [▶ 781]
- F_VN_TransformIntoDisplayableImage [▶ 812]
- F_VN_TransformIntoDisplayableImageExp [▶ 813]

The F_VN_CopyIntoDisplayableImage [▶ 781] function creates a deep copy of the image data and makes it available as a displayable image. It should always be used if the image to be displayed is to be used in the following program sequence:

```
// 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);
```

The function F_VN_TransformIntoDisplayableImage [▶ 812] uses the existing image data and releases the original interface pointer of the image. This means that the image data can no longer be processed and are fully available for image transmission. Since no copy of the image data is created, the execution time of this function is much shorter and no additional memory is used. However, this function only works if there is no other pointer to the image data. Otherwise the error code `E_INCOMPATIBLE (16#70E)` is returned. This

can occur among other things if the Ads Communicator object [▶ 124] is activated (e.g. stream recording [▶ 102]), as this continues to reference the image internally for a short time. This function is therefore mainly useful for the end of the processing chain in order, for example, to display an already generated result image.

```
// 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);
```

In addition, the function F_VN_TransformIntoDisplayableImageExp [▶ 813] offers the option to create a deep copy of the original image on demand. Thus, the resource-saving transform variant is used if possible, while the copy variant is used if necessary. A corresponding use case can occur if the AdsCommunitor object [▶ 123] of a GigE Vision Camera [▶ 69] instance is activated to save images as a stream to the file system. In addition to the pointer in the PLC, there is another image data pointer in the AdsCommunicator object. If this is not yet enabled in the PLC at the time the image is used, the image must be copied for display.

```
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 ...
```

Memory limitation for displayable images

Images that are displayed in the development environment (ADS Image Watch or camera assistants) are located in the same RAM area as the respective Visual Studio instance. If you now wish to view a very large number of images or images of a large size, bottlenecks may occur due to the maximum available memory per instance.

As the images are retrieved from the PLC for display in the ADS Image Watch (ADS communication) and this exchange takes place via the router memory, sufficient memory space must be available at this point.

Meta information

Meta information for an image can be obtained with the function F_VN_GetImageInfo [▶ 797]. As a result of the function, a structure of the type TcVnImageInfo [▶ 230] is returned. The information it contains can be used, for example, to ensure that an image has the expected format in order to be converted:

```
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

A typical task in image processing is to find objects in the image and to determine their shape, position and size. The objects found can be described, for example, by their contour points. Since the number of contour points depends on the specific object, it can vary greatly. For this reason, containers of dynamic size are used to store these contours.

However, the containers are not limited to storing contour points, they can also contain a variety of other data types. Different data types cannot be mixed within a container. Some basic concepts for handling containers are explained below.

Creating a container

A container can be created with a single function call. Only the type (see available types [▶ 160]) and the initial number of elements have to be defined:

```
PROGRAM MAIN
VAR
    hr           : HRESULT;
    ipContainer  : ITcVnContainer;
END_VAR

hr := F_VN_CreateContainer(ipContainer, ContainerType_Vector_REAL, 10, hr);
```

In this sample a new container with 10 elements of type *REAL* was created, which is internally organized as C++ vector. Alternatively, the function F_VN_CopyContainer [▶ 750] can be used to create a deep copy of an existing container. Not only the pointer to the data is copied - as with an assignment to another variable - but the data is copied to a newly allocated memory area.

All available types of containers are stored as global constants [▶ 160] and named according to the following pattern: All names begin with "`ContainerType_`". This is followed by the type, which is composed of the description of the actual container type (based on C++ containers) and the element types (e.g. `REAL`, `UDINT`, etc.). Containers can in turn also contain further containers as elements. Each level of the container structure is represented in the name by the respective container type, e.g. "`Vector_`" or "`String_`". Therefore the resulting type name of a container with `REAL` elements is `ContainerType_Vector_REAL`.

Adding elements

Since the containers are able to dynamically expand the allocated memory, additional elements can be added to an existing container. The functions F_VN_AppendToContainer [▶ 549] and F_VN_InsertIntoContainer [▶ 679] can be used for this.

However, as it may be internally necessary to allocate a completely new memory area and to copy the existing data, it is recommended to reserve the maximum required memory in advance. Consequently, new memory is allocated only once and the program has an overall higher performance. A function exists for this, F_VN_ReserveContainerMemory [▶ 770], which merely reserves memory, but does not change the number of elements in the container.

```
// 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);
```

Accessing elements

In the case that only a few individual elements of the container are to be accessed, it is recommended to use the F_VN_GetAt_... or F_VN_SetAt_... functions. Since no function name overload is possible in the PLC, there is a separate function name for each supported container type. The first element in the container has the 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
```

The advantage of using these ready-made functions is that only a single line of code is required and all checks (e.g. whether the container type is correct and the index is not too large) are performed internally. If you wish to access all elements in the container one after the other, it is recommended in particular with large containers to access the container elements via iterators and the associated access interface in order to achieve a better performance:

```
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));
```

If the elements in the container are in turn containers, use the following procedure instead.

Container of containers

Since, for example, the function F_VN_FindContours [▶ 1187] can find not only one but several contours, containers can contain further containers as elements (in which case all inner containers must contain elements of the same type). In principle, these can be used in exactly the same way as the simple containers described above. For this purpose there is, for example, the function F_VN_GetAt_ITcVnContainer [▶ 645] for obtaining a deep copy of any inner container that is usable from the PLC (it is technically impossible to obtain an interface pointer to the original data). Due to the general ITcVnContainer data type at this level, it is not necessary to distinguish which base element types are contained in the inner containers. This simplifies access via iterators, since the special Access Interface is no longer necessary:

```
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));
```

Displaying containers

At present the content of containers cannot be displayed in the Visual Studio Live Debugging without further action. Instead, the data can be exported to an array and viewed in this way:

```
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
```

Overwriting containers

Generally, containers with API functions can simply be overwritten. At present, however, overwriting is not possible if the existing container and the container to be written are of different types. This is signaled by the error code 70E (INCOMPATIBLE).

6.1.1.5.3 Contours

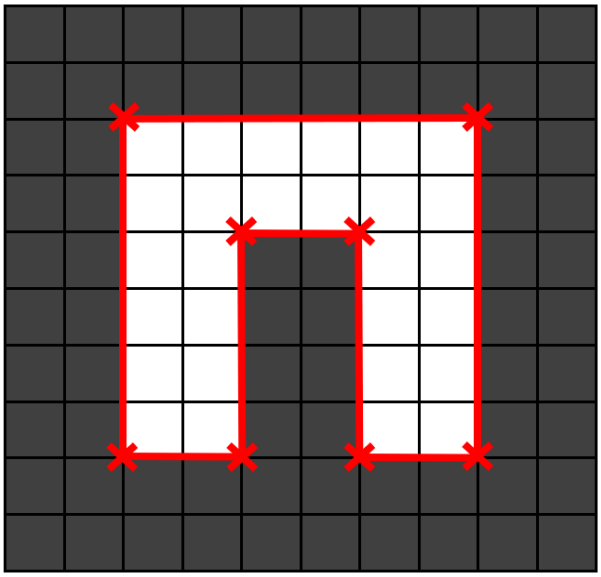
This chapter describes the use of contours in TwinCAT Vision.

What are contours?

In general, a contour (also called an outline) is the demarcation of an object from its surroundings; this is classically two-dimensional in image processing. A contour describes the shape, size and location of this object. Within the scope of image processing, this means an object in an image [▶ 149]; hence, a contour describes the outline of the object in the image.

How are contours displayed?

A contour is displayed as a collection of 2D points. The points mostly refer to the origin of the image in which the object is mapped, provided no ROI has been set. If you connect all the points in this field with straight lines, you get a drawing of the contour.



How are contours displayed in TwinCAT Vision?

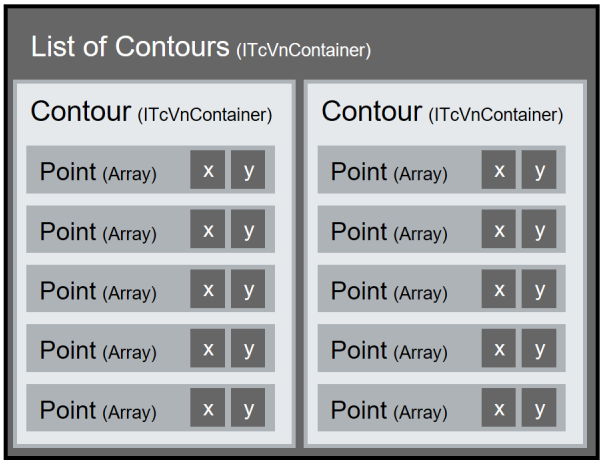
A dynamic array of 2D points is handled in TwinCAT Vision as a container [▶ 152] (`ITcVnContainer`) of points (`TcVnPoint2_DINT/REAL`). A corresponding container then has the type designation `ContainerType_Vector_TcVnPoint2_DINT/REAL`.

The points then frequently lie directly on the pixel grid (e.g. in case of a contour search). In this case, `DINT` elements are used to save the points. However, there are also cases in which the accuracy is insufficient, and `REAL`s therefore have to be used (e.g. in case of measurements). The points then do not lie exactly at the center point of a pixel, but somewhere else.

Drawing functions can only draw contours with integer elements in an image. In case of doubt, therefore, the element type of the container must be converted.

Fields of contours

A frequent occurrence is where several contours are to be handled at the same time (e.g. several contours are usually found with F_VN_FindContours [▶ 1187]). This is technically implemented by using a two-dimensional container. Therefore, there is an array of contours or an array of arrays of 2D points. The corresponding type designation is then `ContainerType_Vector_Vector_TcVnPoint2_DINT/REAL`.



What is contour approximation?

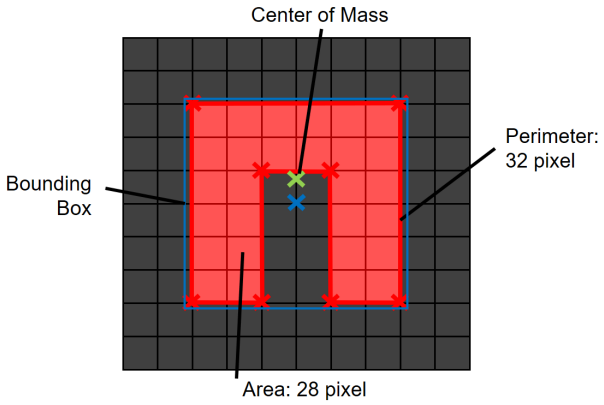
Generally, a very large number of points can always be used for displaying an object. However, this is frequently unnecessary, as (approximately) straight lines can be approximated through their start and end point. The points in between do not need to be saved in addition, because the start and end point suffice for the definition.

Depending on the method used and the geometry of the objects, the approximation can be more or less lossy. In any case, in the case of differing approximation, the geometrical features can also be calculated with slightly different values.

Geometrical features

On contours, i.e. from the arrays of points, a large number of individual features can be calculated that geometrically describe the respective contour. The contour and thus the respective object can be examined and, for example, classified via these features. These features include:

- Scope
- Surface
- Center points



Closed and open contours

A contour is normally closed. In this case the first and last element of the array are simply regarded as being connected. However, it is also possible to have an open contour. Open and closed contours thus do not differ in terms of their technical representation, but by the information regarding how the data are to be interpreted. Whether the respective contour is to be used open or closed will be queried accordingly at the relevant points of the API.

Samples

- Object Detection [▶ 2780]
- Contour analysis [▶ 2771]

6.1.1.5.4 Masks

This chapter describes the meaning of **masks** in TwinCAT Vision.

What is a mask?

In relation to image processing, a mask means a mask image that belongs to another image [▶ 149]. The mask image has the same size and consists of a grayscale channel. The intensities of the mask image define which pixels of the associated image are to be taken into account for certain work steps.

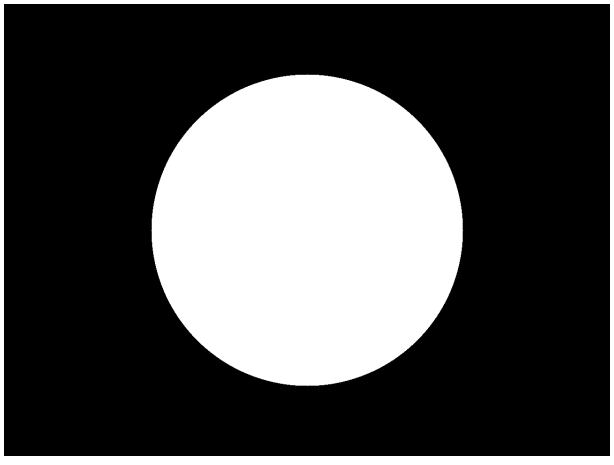
What is a mask needed for?

Without further action, images are limited to a rectangular shape. If shapes other than rectangles are to be defined during image operations, this can be implemented by means of a mask. The mask is displayed by a further image with the same shape and size on which the areas (pixels) to be taken into account are marked differently.

How is a mask technically displayed?

The intensities of the mask grayscale image describe which pixels are to be considered. The interpretation of the intensities can in principle deviate from function to function. Binarily, it is frequently the case that all pixels with a value of 0 are interpreted as not to be considered and all pixels with a different value (>=1) are interpreted as to be considered. However, cases are also conceivable in which the degree of consideration can be set with a finer granularity.

A mask that restricts the observation area to a circular area in the image can look like this, for example:



When applying this mask, the pixels in the corner of an associated image have no further influence on a subsequent operation.

Sample

By way of example, a mask is created for the image `ipImageWork` that says that only a circular area of the image is to be considered:

```
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);
```

The average intensity value of the image within the mask area is subsequently calculated:

```
hr := F_VN_ImageAverageExp(ipImageWork, aAverage, ipImageMask, hr);
fAverageIntensityInCircle := aAverage[0];
```

Without a mask it would not be possible without further action to restrict the average value to the circular area instead of the complete rectangular image.

The full sample can be found here: Average intensity in shapes of any kind [▶ 2795].

6.1.1.5.5 Region of Interest

This chapter describes the use of a **Region of Interest** in TwinCAT Vision.

What is a Region of Interest?

A Region of Interest (ROI) is a rectangular image region to which an image is temporarily restricted. The image itself is not changed, but the section to be processed merely reduced in size. An ROI can be reset to the entire image at any time.

What is a Region of Interest needed for?

An existing image is frequently larger than the area to be processed. In this case an ROI should be set in order to obtain the following advantages:

1. Better performance: The image processing time is shorter if the image region to be processed is smaller.
2. Less interference: Interfering image objects that are outside the ROI are excluded from processing and therefore do not affect the result.

It is equally possible through the use of ROIs to process several image areas separately by setting the ROI at various points in the image, one after the other.

NOTICE

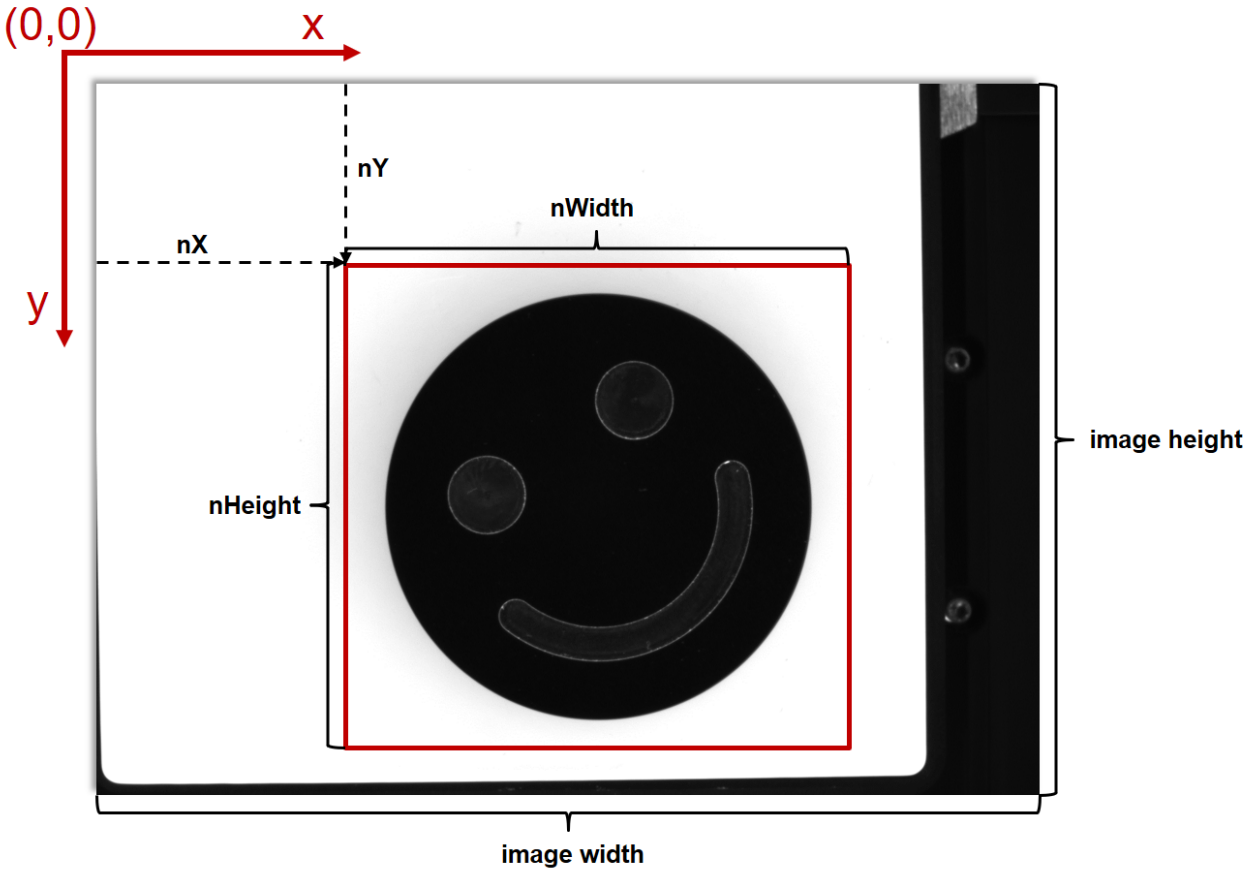
ROI configuration in the camera

If you use a camera as an image source, it is best to implement an ROI there in order to reduce the data quantity to be transmitted and to keep it as low as possible. Depending on the setting and sensor, the possible frame rate may also increase as a result of this. See camera configuration examples: Region of Interest (ROI) [▶ 2845]

How is a Region of Interest technically displayed?

The information about a set ROI is stored in TwinCAT Vision directly in the respective image object. The setting of an ROI merely consists of the setting of a rectangle description. The image data outside of this area still exist, but they are ignored when calling functions.

The rectangle description consists of the position (`nX` and `nY`) of the top left corner and the size (`nWidth` and `nHeight`) of the rectangle. Make sure that the rectangle does not extend beyond the edge of the picture (`nWidth ≤ ImageWidth - nX` and `nHeight ≤ ImageHeight - nY`).



To set an ROI, these parameters are transferred directly or as a structure of the type TcVnRectangle_UDINT [▶ 245], depending on the function:

- F_VN_SetRoi [▶ 807]
- F_VN_SetRoi_TcVnRectangle_UDINT [▶ 809]

The function F_VN_GetRoi [▶ 801] is available for checking whether and which ROI is set on an image. An ROI can also be set on an image that already has an ROI. The function F_VN_ResetRoi [▶ 803] can be used if the ROI is to be enlarged or reset to the entire image size.

6.1.2 Data Types

There are four data types, each with the prefix "TcVn":

- Alias names [▶ 160] for certain array types refer to image processing
- Arrays [▶ 162] with a specific combination of data types and a defined number that are frequently used by the image processing functions.
- Enums [▶ 162] help with the parameterization of functions and contain the additional prefix "E"
- Structures [▶ 225] consolidate several related pieces of information (e.g. for geometric descriptions)

In addition, constants [▶ 160] are available for the description of container types.

6.1.2.1 Aliases

Name	Type
GVCP_REGISTER_ADDRESS	UDINT
TcVnPoint	TcVnPoint2_DINT [▶ 160]
TcVnPoint2_DINT	TcVnVector2_DINT [▶ 162]
TcVnPoint2_LREAL	TcVnVector2_LREAL [▶ 162]
TcVnPoint2_REAL	TcVnVector2_REAL [▶ 162]
TcVnPoint3_LREAL	TcVnVector3_LREAL [▶ 162]
TcVnPoint3_REAL	TcVnVector3_REAL [▶ 162]
TcVnRectangle	TcVnRectangle_DINT [▶ 244]

6.1.2.2 Constants

The following constants represent types of containers [▶ 152] as `GUID`. These are required, for example, with functions such as F_VN_CreateContainer [▶ 753].

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_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 [▶ 411]
TcVnArray3_Point2_REAL	array [0..2] of TcVnPoint2_REAL [▶ 160]
TcVnArray33_UDINT	array [0..32] of UDINT
TcVnArray4_LREAL	array [0..3] of LREAL
TcVnArray4_Point2_REAL	array [0..3] of TcVnPoint2_REAL [▶ 160]
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.

Further information

This enum provides setting options for the search strategy of code reading functions. For more information see Code reading search strategies [▶ 838].

Related functions

- F_VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadQRCodeExp [▶ 858]

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.

Related functions

- F_VN_AdaptiveThresholdExp [▶ 1329]

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.

Related functions

- F_VN_ReadBarcodeExp [▶ 841]

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.

Related functions

- F_VN_ReadBarcode(Exp) [▶ 839]

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.

Further information

The enum `ETcVnBlobCombination` is used in the structure TcVnParamsBlobDetection [▶ 234].

Related functions

- F_VN_CalibrateCamera(Exp) [▶ 1073]

- F_VN_DetectBlobs(Exp) [▶ 1178]

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.

Further information

Border interpolation is used to simulate pixel values outside of the image. This is necessary, for example, when applying a filter or a morphological operator to an image. For example, if a filter iterates over the image, it must be specified how the filter behaves at the image edges where parts of the filter kernel lie outside of the image. For this purpose the non-existent pixels outside of the image are simulated so that the filter operation can still be performed. The method with which this simulation is performed is defined with the enum `ETcVnBorderInterpolationMethod`. The value `TCVN_BIM_REFLECT` is often a good choice; it reflects the outer image pixels with the image border as a mirror plane.

Related functions

- F_VN_BilateralFilterExp [▶ 1280]
- F_VN_CustomFilterExp [▶ 1290]
- F_VN_GaussianFilterExp [▶ 1298]
- F_VN_LaplacianFilterExp [▶ 1301]
- F_VN_PadImageBorderExp [▶ 1070]
- F_VN_ScharrFilterExp [▶ 1315]
- F_VN_SeparableCustomFilterExp [▶ 1319]
- F_VN_SobelFilterExp [▶ 1323]
- F_VN_WarpAffineExp [▶ 1169]
- F_VN_WarpPerspectiveExp [▶ 1174]

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.

Further information

The state machines used for these camera states are described in the respective function blocks for image acquisition [▶ 1611]:

- FB_VN_GevCameraControl [▶ 1621]
- FB_VN_FileSourceControl [▶ 1613]
- FB_VN_SimpleCameraControl [▶ 1646]

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:

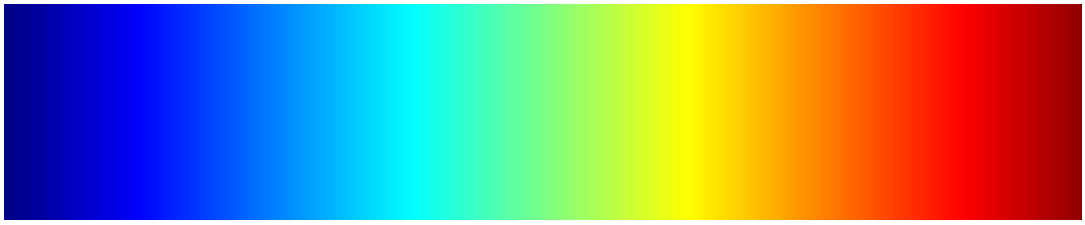
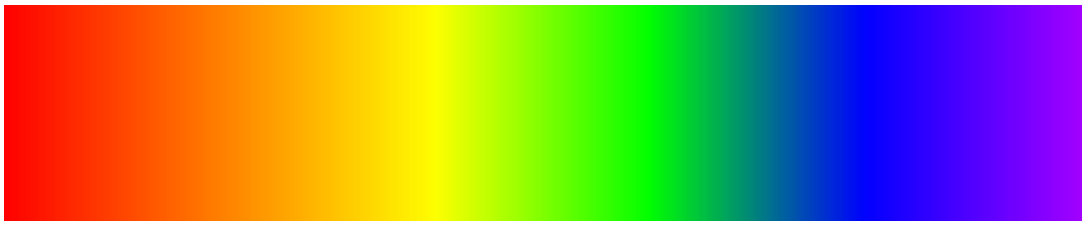
```
TYPE ETcVnColorMap :
(
    TCVN_CM_AUTUMN           := 0,
    TCVN_CM_BONE             := 1,
    TCVN_CM_JET              := 2,
    TCVN_CM_WINTER           := 3,
    TCVN_CM_RAINBOW          := 4,
    TCVN_CM_OCEAN            := 5,
    TCVN_CM_SUMMER           := 6,
    TCVN_CM_SPRING           := 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
```

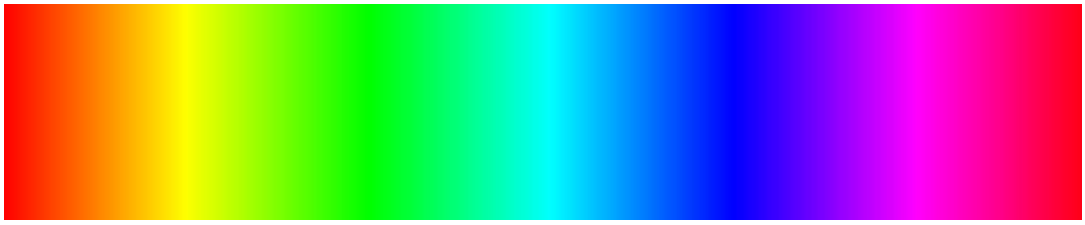
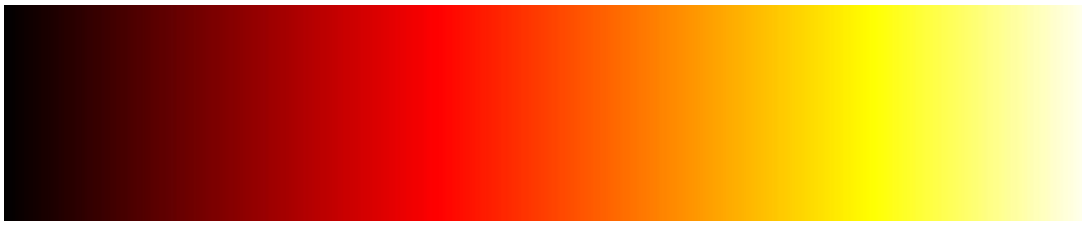
Values

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

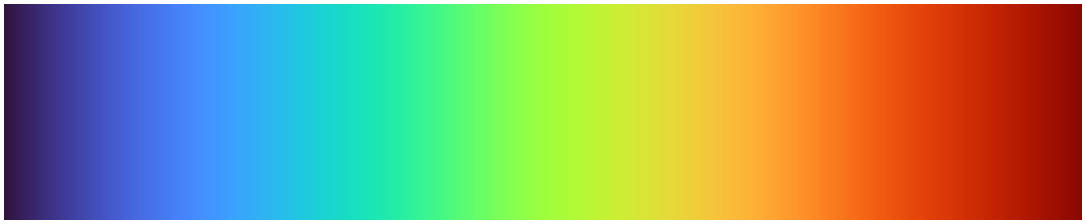
Color gradients

The enum `ETcVnColorMap` defines color gradients for the creation of color tables with the help of the function F_VN_GenerateColorMap [▶ 1241]. The following color gradients are predefined:

NONE	
AUTUMN	
BONE	
JET	
WINTER	
RAINBOW	
OCEAN	

SUMMER	
SPRING	
COOL	
HSV	
Pink	
HOT	
PARULA	

MAGMA	
INFERNO	
PLASMA	
VIRIDIS	
CIVIDIS	
TWILIGHT	
TWILIGHT_SHIFTED	

TURBO	

Related functions

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]

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)

Related functions

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]

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.

Related functions

- F_VN_ConvertColorSpace [▶ 1241]

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.

Related functions

- F_VN_TrainImageColorExp [▶ 1264]

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)

Related elements

- FB_VN_WriteContainer [▶ 1603]

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
```

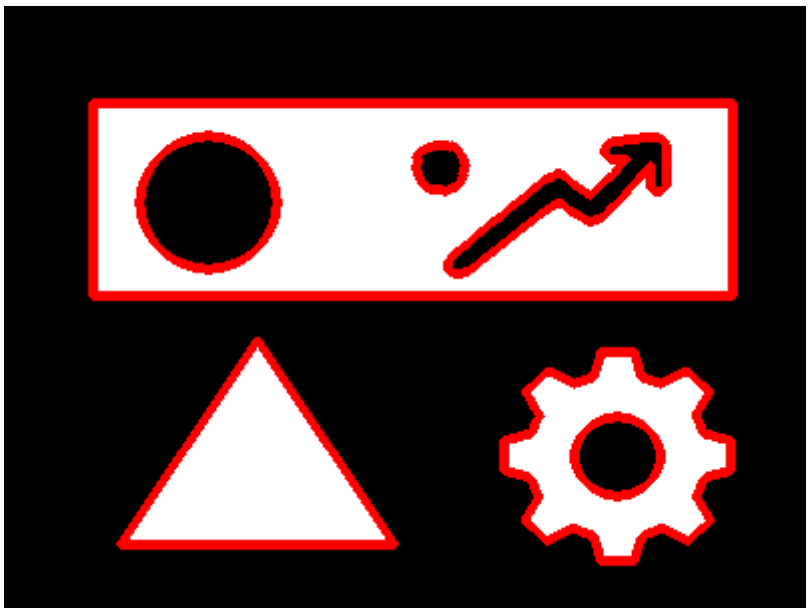
Values

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).

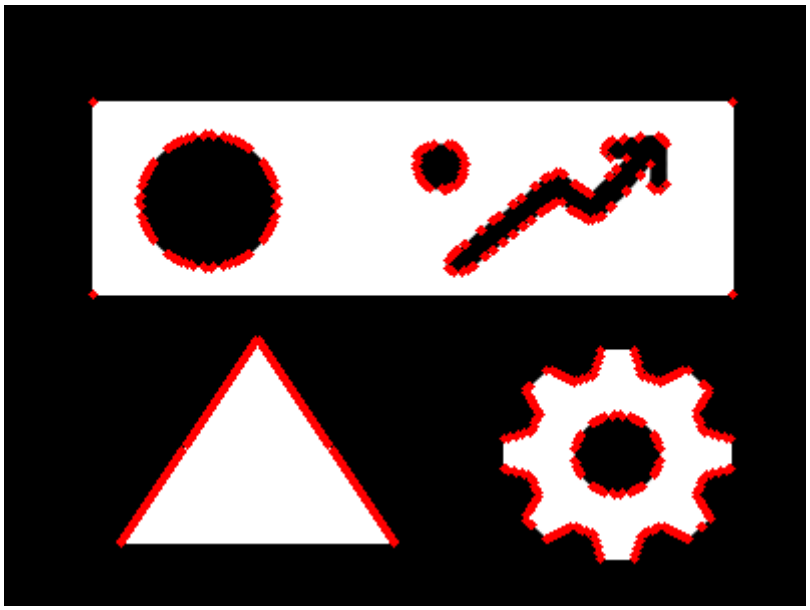
Further information

This type indicates whether and if so how the points describing a contour are to be simplified.

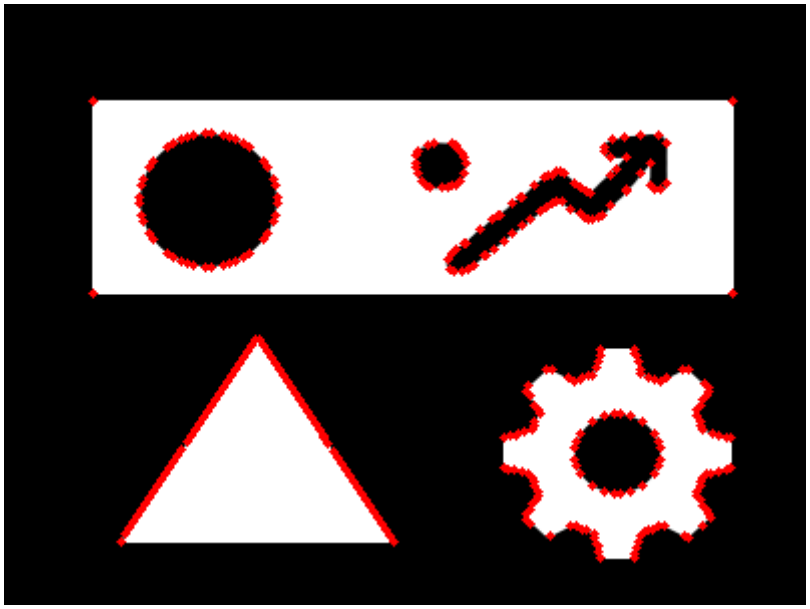
- **TCVN_CAM_NONE**
  No simplification of the contour description, the contours are described by all enclosing points.



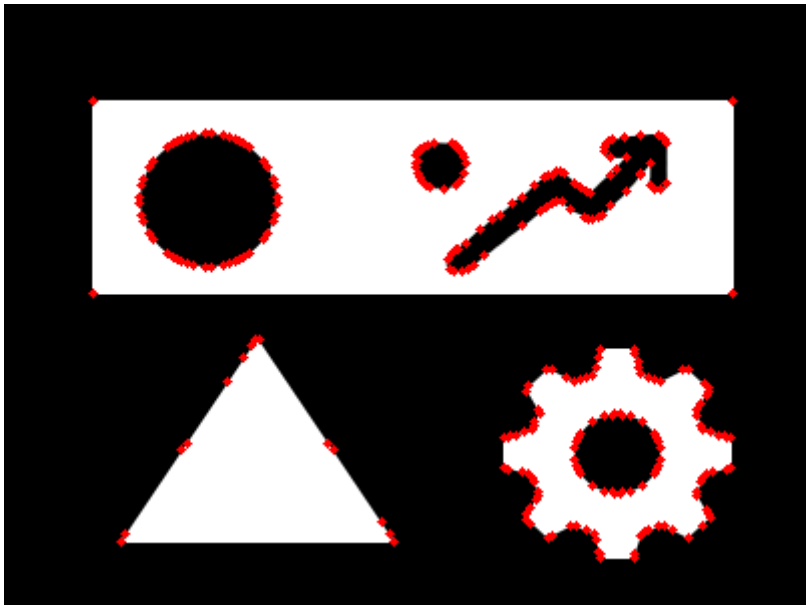
- **TCVN_CAM_SIMPLE**
  Lossless simplification



- **TCVN_CAM_TC89_L1**
  Simplification of contour points using the algorithm Teh-Chin chain Approximation (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).



- **TCVN_CAM_TC89_KCOS**
  Simplification of contour points using the algorithm Teh-Chin chain Approximation (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).



Related functions

- F_VN_FindContoursExp [▶ 1190]
- F_VN_FindContourHierarchyExp [▶ 1183]

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
```

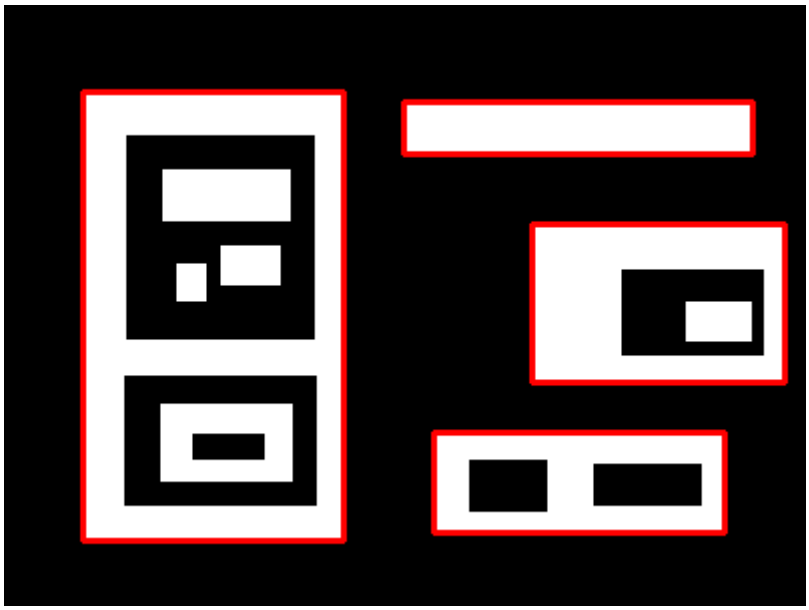
Values

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).

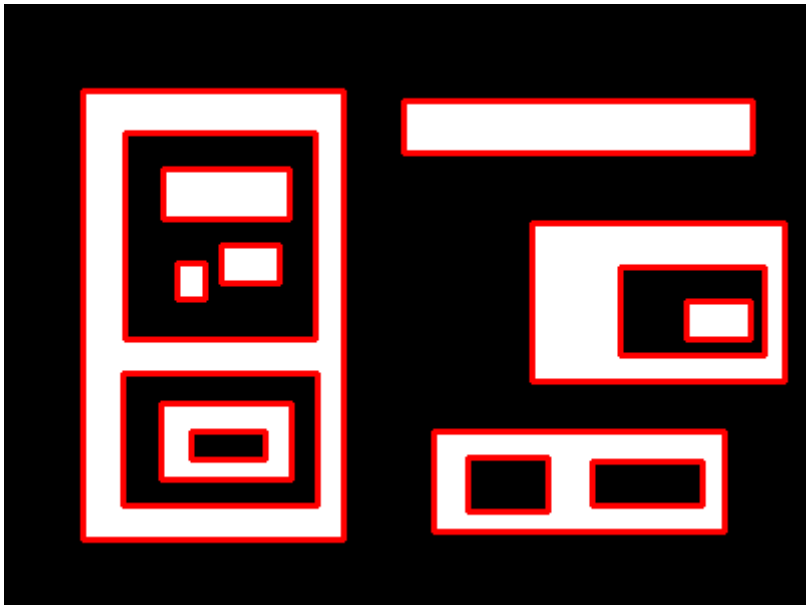
Further information

This type specifies how the contour hierarchy is to be taken into account in the contour search.

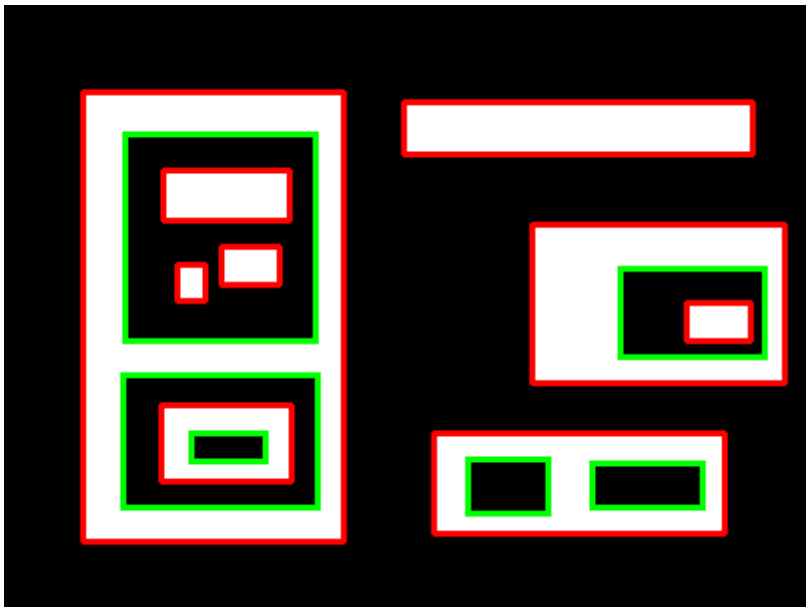
- **TCVN_CRM_EXTERNAL**
  Only the outer contours are returned; inner contours are not taken into account.



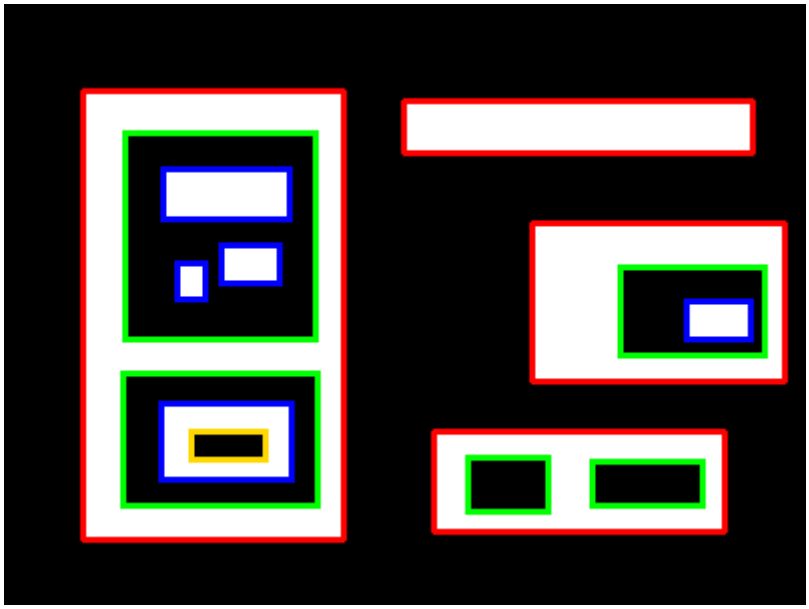
- **TCVN_CRM_LIST**
  All contours are returned. The contour hierarchy is not taken into account.



- **TCVN_CRM_CONNECTED_COMPONENTS**
  All contours are returned, and a 2-level hierarchy is created. In the graphic, all contours at level 1 are shown in red, level 2 is shown in green. It is evident that the next inner contour after a level 2 contour is again on level 1.



- **TCVN_CRM_TREE**
  All contours are returned, and a hierarchy is created based on a tree structure. In the image level 1 is red, level 2 green, level 3 blue and level 4 yellow.



Related functions

- F_VN_FindContoursExp [▶ 1190]
- F_VN_FindContourHierarchyExp [▶ 1183]

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.

Further information

The available methods are based on Hu invariants. The following algorithms are used, with A representing the first contour and B the second contour.

$$I_1(A,B) = \sum_{i=1..7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right|$$

$$I_2(A,B) = \sum_{i=1..7} \left| m_i^A - m_i^B \right|$$

$$I_3(A,B) = \max_{i=1..7} \frac{\left| m_i^A - m_i^B \right|}{\left| m_i^A \right|}$$

$$m_i^A = sign(h_i^A) \cdot \log(h_i^A)$$

$$m_i^B = sign(h_i^B) \cdot \log(h_i^B)$$

h_i^A and h_i^B with `i` in `[1, 7]` are each Hu moments of A and B.

Related functions

- F_VN_MatchContours(Exp) [▶ 983]
- F_VN_MatchContours1vsN(Exp) [▶ 984]
- F_VN_MatchImageHuMoments [▶ 1198]

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.

Further information

The enum `ETcVnDiffusivityTypeKAZE` is used in the structure TcVnParamsKAZE [▶ 240].

Related functions

- F_VN_KeyPointsAndDescriptorsKAZEExp [▶ 1363]

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).

Related functions

- F_VN_DistanceTransformationExp [▶ 1215]

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.

Related functions

- F_VN_DistanceTransformation(Exp) [▶ 1213]

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

Related functions

- F_VN_DistanceTransformation(Exp) [▶ 1213]
- F_VN_FitLineExp [▶ 981]

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.

Related functions

- F_VN_DrawKeypointsExp [▶ 1015]
- F_VN_DrawMatchesExp [▶ 1027]

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.

Related functions

- F_VN_DrawPoint(Exp) [▶ 1031]
- F_VN_DrawPoints(Exp) [▶ 1033]

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.

Related functions

- Measurement [▶ 1481]
  - F_VN_LocateCircularArcExp [▶ 1492]
  - F_VN_LocateEdgeExp [▶ 1498]
  - F_VN_LocateEdgesExp [▶ 1504]
  - F_VN_LocateEllipseExp [▶ 1510]
  - F_VN_MeasureAngleBetweenEdgesExp [▶ 1515]
  - F_VN_MeasureEdgeDistanceExp [▶ 1520]
  - F_VN_MeasureMinEdgeDistanceExp [▶ 1525]

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

Related functions

- Measurement [▶ 1481]

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)

Further information

The enum `ETcVnElementType` is mainly used to specify the pixel type or bit depth of images. Furthermore, it is also contained in the structure TcVnMatrix [▶ 231].

Related functions

- F_VN_ConvertElementType(Exp) [▶ 775]
- F_VN_CreateImage [▶ 784]
- F_VN_CreateImageAndSetPixels [▶ 785]
- F_VN_CreateImageFromArray [▶ 786]
- F_VN_CustomFilter(Exp) [▶ 1287]
- F_VN_InitMatrixStruct [▶ 1539]
- F_VN_LaplacianFilter(Exp) [▶ 1299]
- F_VN_NormalizeImageExp [▶ 1252]
- F_VN_ScharrFilter(Exp) [▶ 1313]
- F_VN_SeparableCustomFilter(Exp) [▶ 1317]
- F_VN_SobelFilter(Exp) [▶ 1321]

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

Related functions

- F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1352]
- F_VN_HomographyExp [▶ 1148]

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).

Related functions

- F_VN_ContourExtremePoint [▶ 964]

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).

Further information

The enum `ETcVnFeatureDescriptorTypAKAZE` is used in the structure TcVnParamsAKAZE [▶ 234].

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).

Further information

The enum `ETcVnFeatureScoreTypeORB` is used in the structure TcVnParamsORB [▶ 241].

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

Related functions

- F_VN_ScharrFilter(Exp) [▶ 1313]

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

Related functions

- F_VN_FlipImage [▶ 1141]

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).

Further information

The following fonts are available:



Related functions

- F_VN_PutLabelExp [▶ 1052]
- F_VN_PutText(Exp) [▶ 1053]

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.

Related functions

- F_VN_HoughCirclesExp [▶ 1193]

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

Related functions

- F_VN_MatchTemplateAndEvaluateExp [▶ 1201]
- F_VN_RemapImageToLogPolarSpaceExp [▶ 1154]
- F_VN_RemapImageToPolarSpaceExp [▶ 1157]
- F_VN_ResizeImage [▶ 1159]
- F_VN_WarpAffineExp [▶ 1169]
- F_VN_WarpPerspectiveExp [▶ 1174]

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

Further information

The enum `ETcVnKeypointDetectionTypeAGAST` is used in the structure TcVnParamsAGAST [▶ 233].

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

Further information

The enum `ETcVnKeypointDetectionTypeFAST` is used in the structure TcVnParamsFAST [▶ 239].

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).

Related functions

- F_VN_DrawCircleExp [▶ 1001]
- F_VN_DrawCirclesExp [▶ 1003]
- F_VN_DrawCircularArcExp [▶ 1005]
- F_VN_DrawContoursExp [▶ 1010]
- F_VN_DrawEllipseExp [▶ 1013]
- F_VN_DrawLineExp [▶ 1020]
- F_VN_DrawLineExp_TcVnVector4_DINT [▶ 1021]
- F_VN_DrawLineExp_TcVnVector4_LREAL [▶ 1022]
- F_VN_DrawLinesExp [▶ 1025]
- F_VN_DrawPointExp [▶ 1032]
- F_VN_DrawPointsExp [▶ 1034]
- F_VN_DrawRotatedRectangleExp [▶ 1041]
- F_VN_PutLabelExp [▶ 1052]
- F_VN_PutTextExp [▶ 1055]

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.

Related functions

- F_VN_MorphologicalOperator [▶ 1309]

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.

Related functions

- F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1354]
- F_VN_MatchDescriptorsBFExp [▶ 1375]
- F_VN_MatchDescriptorsKnnBFExp [▶ 1379]
- F_VN_NormalizeImageExp [▶ 1252]

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).

Further information

The enum `ETcVnPixelEncoding` is used in the structure TcVnPixelFormat [▶ 243]. Information about the pixel encoding can be obtained by querying the pixel format information of an image. A frequent use case is the query for a Bayer pattern in order to find out whether a picture still needs to be converted into an RGB format, for example.

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

Further information

The enum `ETcVnPixelPackMode` is used in the structure TcVnPixelFormat [▶ 243].

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

Related functions

- F_VN_RotateImage [▶ 1162]

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.

Related functions

- F_VN_NormalizeImageForDisplay [▶ 1253]

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

Related functions

- F_VN_CreateStructuringElement [▶ 1285]

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

Related functions

- F_VN_MatchTemplateAndEvaluateExp [▶ 1201]
- F_VN_MatchTemplateExp [▶ 1203]

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

Further information

The Enum `ETcVnThresholdType` offers a choice of the methods implemented to calculate the threshold value. It is also used in the structure TcVnParamsBlobDetection [▶ 234].

Related functions

- F_VN_AdaptiveThresholdExp [▶ 1329]
- F_VN_Threshold [▶ 1334]

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

Related functions

- F_VN_MaxPixelValueExp [▶ 1228]
- F_VN_MinPixelValueExp [▶ 1230]

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).

Further information

With this enum you can set how a watchdog [▶ 146] calculates the fraction of the executed functions (`nFractionProcessed`) when several functions are concatenated.

As an example to illustrate the mode of operation, it is assumed that three functions were monitored. The first was executed 100%, the second 50% and the third 0%.

WD_ACC_TYPE_MEAN	The average value of the execution fractions of all functions is returned as the result. In the example case the result would be: **(100% + 50% + 0%) / 3 = 50%**.
WD_ACC_TYPE_PRODUCT	The product of the execution fractions of all functions is returned as the result. In the example case the result would be: **100% * 50% * 0% = 0%**.

The use cases differ depending on the program structure and the safe states of the machine.

Related functions

- F_VN_StartAbsWatchdogExp [▶ 990]
- F_VN_StartRelWatchdogExp [▶ 992]

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	Unknown
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

6.1.2.5 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 [▶ 225]	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 [▶ 226]	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 [▶ 160]	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 [▶ 243]	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 [▶ 160]	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 [▶ 199]	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 [▶ 206]	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 [▶ 201]	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 [▶ 194]	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;
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 [▶ 220]	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 [▶ 165]	TCVN_BC_MEDIAN_THRESHOLD	Selects, which of the multi-threshold blob contours should be returned.

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 [▶ 207]	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 [▶ 194]	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 [▶ 202]	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 [▶ 213]	Pixel encoding
ePixelPackMode	ETcVnPixelPackMode [▶ 213]	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 [▶ 160]	Center point
stSize	TcVnSize2_REAL [▶ 245]	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 do not require a license.

Grouping of interfaces

All interfaces of the TwinCAT Vision library are sorted thematically into the following groups:

- Images [▶ 376] ()
- Container [▶ 246] ()
- Machine Learning [▶ 418] ()
- Miscellaneous [▶ 422] ()

6.1.3.1 Containers

This group contains interfaces for handling containers [▶ 152].

General information

A container is basically a vector of several elements. The number of elements can be changed dynamically. The corresponding memory in the router memory [▶ 54] is automatically allocated and released for this purpose.

Container types

All elements of a container have the same type. Containers are characterized by a unique GUID based on the type of elements they contain. For ease of use, these GUIDs are defined by constants [▶ 160]. All type designations for containers begin with "ContainerType_Vector_" (with the exception of "ContainerType_String_SINT“). Type requirements are documented for all API elements that make use of containers.

The container type can be determined as follows:

```
ipContainer.GetElementTypeGuid(nTypeGuid);
```

6.1.3.1.1 Function Blocks

This group contains interfaces for the implementation of certain function blocks.

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 [▶ 247]	ITcVnCustomContainerOperation_ITcVnContainer	Executes the custom operation on the container elements.

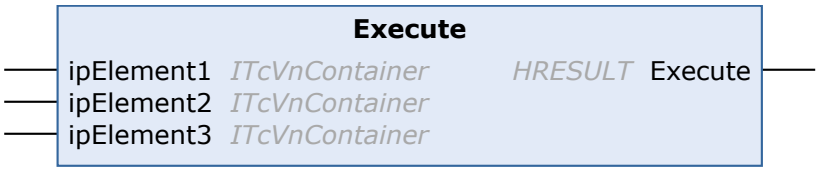
Related functions

- F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 974]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Current element of the 1st container.
ipElement2	ITcVnContainer [▶ 370]	Current element of the 2nd container.
ipElement3	ITcVnContainer [▶ 370]	Current element of the 3rd container.

Return value

HRESULT [▶ 141]

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 [▶ 248]	ITcVnCustomContainerOperation_ITcVnForwardIterator	Executes the custom operation on the container elements.

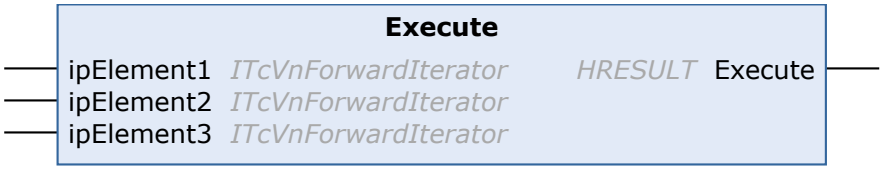
Related functions

- F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 975]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Current element of the 1st container.
ipElement2	ITcVnForwardIterator [▶ 360]	Current element of the 2nd container.
ipElement3	ITcVnForwardIterator [▶ 360]	Current element of the 3rd container.

Return value

HRESULT [▶ 141]

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 [▶ 249]	ITcVnCustomElementCondition_ITcVnContainer	Evaluates the condition for the container.

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750]

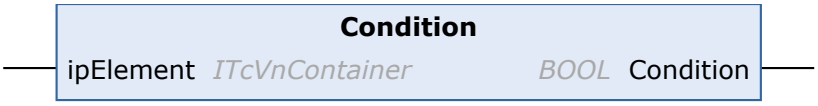
Samples

- Selections of container elements [▶ 2755]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 250]	ITcVnCustomElementCondition_ITcVnForwardIterator	Evaluates the condition for the provided element.

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751]

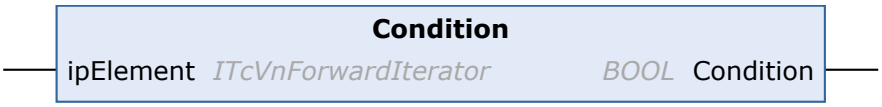
Samples

- Selections of container elements [▶ 2755]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Element for which the condition is evaluated.

Return value

BOOL

6.1.3.1.2 ITcVnAccess

The interface `ITcVnAccess` is for accessing elements of a container [▶ 153] and is provided by an iterator [▶ 359]. Unlike ITcVnRandomAccess [▶ 301], only that element to which the associated iterator is pointing in that moment can be addressed.

6.1.3.1.2.1 ITcVnAccess_DINT

Offers an access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 251]	ITcVnAccess_DINT	Gets the value
Set [▶ 251]	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 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
```

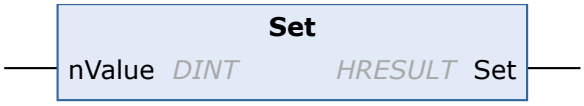
Inputs

Name	Type	Description
nValue	Reference To DINT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.2.2 ITcVnAccess_INT

Offers an access interface for INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 252]	ITcVnAccess_INT	Gets the value
Set [▶ 253]	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 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
```

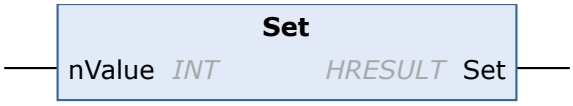
Inputs

Name	Type	Description
nValue	Reference To INT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.2.3 ITcVnAccess_ITcVnImage

Offers an access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 254]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 254]	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 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 [▶ 411]	Returns the image.

Return value

HRESULT [▶ 141]

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 [▶ 411]	The image to set.

Return value

HRESULT [▶ 141]

6.1.3.1.2.4 ITcVnAccess_LREAL

Offers an access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 255]	ITcVnAccess_LREAL	Gets the value
Set [▶ 255]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.2.5 ITcVnAccess_REAL

Offers an access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 256]	ITcVnAccess_REAL	Gets the value
Set [▶ 257]	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 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
```

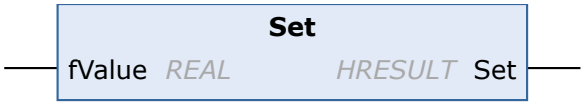
Inputs

Name	Type	Description
fValue	Reference To REAL	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.2.6 ITcVnAccess_SINT

Offers an access interface for SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 258]	ITcVnAccess_SINT	Gets the value
Set [▶ 258]	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 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
```

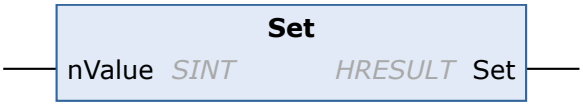
Inputs

Name	Type	Description
nValue	Reference To SINT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.2.7 ITcVnAccess_TcVnDMatch

Offers an access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 259]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 259]	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 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 [▶ 230]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 230]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.8 ITcVnAccess_TcVnKeyPoint

Offers an access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnKeyPoint

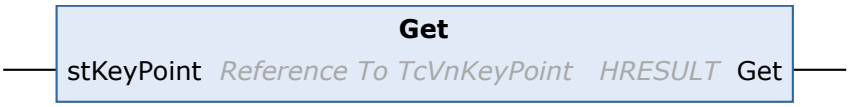
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 260]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 261]	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 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 [▶ 231]	Returns the value.

Return value

HRESULT [▶ 141]

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 [▶ 231]	The value to set.

Return value

HRESULT [▶ 141]

6.1.3.1.2.9 ITcVnAccess_TcVnPoint2_DINT

Offers an access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 262]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 262]	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 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 [▶ 160]	Returns the value.

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set.

Return value

HRESULT [▶ 141]

6.1.3.1.2.10 ITcVnAccess_TcVnPoint2_LREAL

Offers an access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 263]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 264]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.11 ITcVnAccess_TcVnPoint2_REAL

Offers an access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 265]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 265]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.12 ITcVnAccess_TcVnPoint3_LREAL

Offers an access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 266]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 266]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.13 ITcVnAccess_TcVnPoint3_REAL

Offers an access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 267]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 268]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.14 ITcVnAccess_TcVnRectangle_DINT

Offers an access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnRectangle_DINT

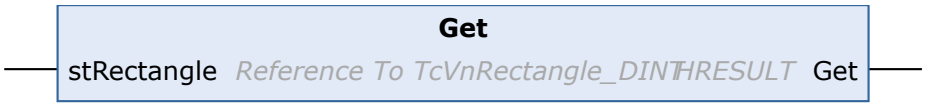
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 269]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 269]	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 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 [▶ 244]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 244]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.15 ITcVnAccess_TcVnVector2_DINT

Offers an access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 270]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 271]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.16 ITcVnAccess_TcVnVector2_INT

Offers an access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 272]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 272]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.17 ITcVnAccess_TcVnVector2_REAL

Offers an access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 273]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 273]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.18 ITcVnAccess_TcVnVector2_SINT

Offers an access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 274]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 275]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.19 ITcVnAccess_TcVnVector2_UINT

Offers an access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 276]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 276]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.20 ITcVnAccess_TcVnVector2_USINT

Offers an access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 277]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 278]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.21 ITcVnAccess_TcVnVector3_INT

Offers an access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 279]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 279]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.22 ITcVnAccess_TcVnVector3_REAL

Offers an access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 280]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 281]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.23 ITcVnAccess_TcVnVector3_SINT

Offers an access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 282]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 282]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.24 ITcVnAccess_TcVnVector3_UINT

Offers an access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 283]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 283]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.25 ITcVnAccess_TcVnVector3_USINT

Offers an access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 284]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 285]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.26 ITcVnAccess_TcVnVector4_DINT

Offers an access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 286]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 286]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.27 ITcVnAccess_TcVnVector4_INT

Offers an access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 287]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 288]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.28 ITcVnAccess_TcVnVector4_LREAL

Offers an access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 289]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 289]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.29 ITcVnAccess_TcVnVector4_SINT

Offers an access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 290]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 291]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.30 ITcVnAccess_TcVnVector4_UINT

Offers an access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 291]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 292]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.31 ITcVnAccess_TcVnVector4_USINT

Offers an access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 293]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 293]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.2.32 ITcVnAccess_UDINT

Offers an access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 294]	ITcVnAccess_UDINT	Gets the value
Set [▶ 295]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.2.33 ITcVnAccess_UINT

Offers an access interface for UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 296]	ITcVnAccess_UINT	Gets the value
Set [▶ 296]	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 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
```

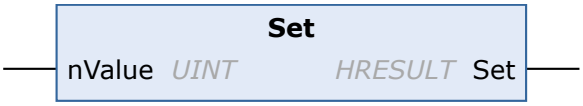
Inputs

Name	Type	Description
nValue	Reference To UINT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.2.34 ITcVnAccess_ULINT

Offers an access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 297]	ITcVnAccess_ULINT	Gets the value
Set [▶ 297]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.2.35 ITcVnAccess_USINT

Offers an access interface for USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 298]	ITcVnAccess_USINT	Gets the value
Set [▶ 299]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.2.36 ITcVnAccess_TcVnVector4_REAL

Offers an access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 300]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 300]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3 ITcVnRandomAccess

The interface `ITcVnRandomAccess` is for accessing any elements of a container [▶ 153] and is provided by an ITcVnRandomAccessIterator [▶ 367]. Unlike ITcVnAccess [▶ 250], any element in a container can be addressed.

6.1.3.1.3.1 ITcVnRandomAccess_DINT

Offers a random access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_DINT [▶ 250]
    ITcVnRandomAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 251]	ITcVnAccess_DINT	Gets the value
Set [▶ 251]	ITcVnAccess_DINT	Sets the value
GetAt [▶ 301]	ITcVnRandomAccess_DINT	Gets the value
SetAt [▶ 302]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.3.2 ITcVnRandomAccess_INT

Offers a random access interface for INT values

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_INT [▶ 252]
    ITcVnRandomAccess_INT

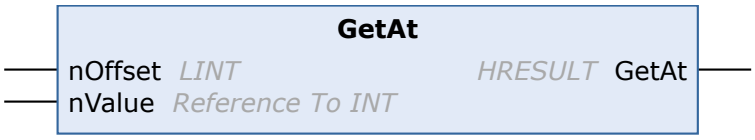
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 252]	ITcVnAccess_INT	Gets the value
Set [▶ 253]	ITcVnAccess_INT	Sets the value
GetAt [▶ 303]	ITcVnRandomAccess_INT	Gets the value
SetAt [▶ 303]	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 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
```

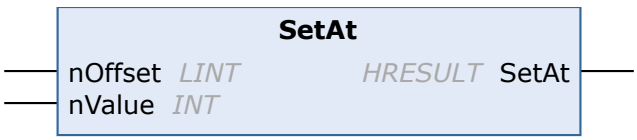
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To INT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.3.3 ITcVnRandomAccess_ITcVnImage

Offers an random access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_ITcVnImage [▶ 253]
    ITcVnRandomAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 254]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 254]	ITcVnAccess_ITcVnImage	Sets the image.
GetAt [▶ 304]	ITcVnRandomAccess_ITcVnImage	Gets the image.
SetAt [▶ 305]	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 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 [▶ 411]	Returns the image.

Return value

HRESULT [▶ 141]

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 [▶ 411]	The image to set.

Return value

HRESULT [▶ 141]

6.1.3.1.3.4 ITcVnRandomAccess_LREAL

Offers a random access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_LREAL [▶ 255]
    ITcVnRandomAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 255]	ITcVnAccess_LREAL	Gets the value
Set [▶ 255]	ITcVnAccess_LREAL	Sets the value
GetAt [▶ 306]	ITcVnRandomAccess_LREAL	Gets the value
SetAt [▶ 307]	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 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
```

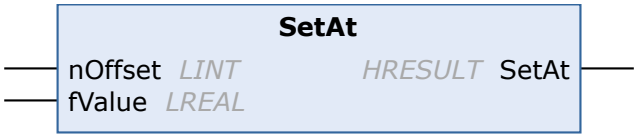
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
fValue	Reference To LREAL	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.3.5 ITcVnRandomAccess_REAL

Offers a random access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_REAL [▶ 256]
    ITcVnRandomAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 256]	ITcVnAccess_REAL	Gets the value
Set [▶ 257]	ITcVnAccess_REAL	Sets the value
GetAt [▶ 308]	ITcVnRandomAccess_REAL	Gets the value
SetAt [▶ 308]	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 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
```

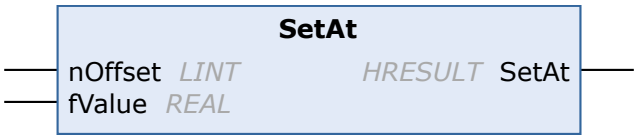
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
fValue	Reference To REAL	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.3.6 ITcVnRandomAccess_SINT

Offers a random access interface for SINT values

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_SINT [▶ 257]
    ITcVnRandomAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 258]	ITcVnAccess_SINT	Gets the value
Set [▶ 258]	ITcVnAccess_SINT	Sets the value
GetAt [▶ 309]	ITcVnRandomAccess_SINT	Gets the value
SetAt [▶ 310]	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 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
```

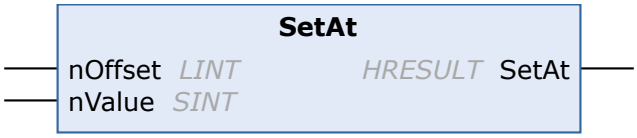
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To SINT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.3.7 ITcVnRandomAccess_TcVnDMatch

Offers an random access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnDMatch [▶ 259]
    ITcVnRandomAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 259]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 259]	ITcVnAccess_TcVnDMatch	Sets the value
GetAt [▶ 311]	ITcVnRandomAccess_TcVnDMatch	Gets the value
SetAt [▶ 311]	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 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 [▶ 230]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 230]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.8 ITcVnRandomAccess_TcVnKeyPoint

Offers a random access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnKeyPoint [▶ 260]
    ITcVnRandomAccess_TcVnKeyPoint

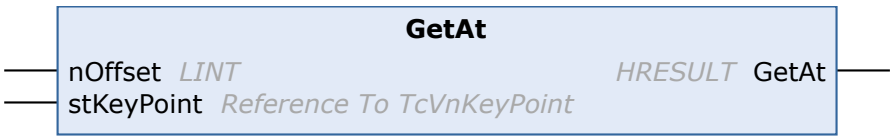
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 260]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 261]	ITcVnAccess_TcVnKeyPoint	Sets the value.
GetAt [▶ 312]	ITcVnRandomAccess_TcVnKeyPoint	Gets the value
SetAt [▶ 313]	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 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
```

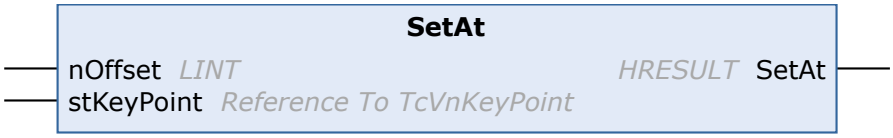
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stKeyPoint	Reference To TcVnKeyPoint [▶ 231]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 231]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.9 ITcVnRandomAccess_TcVnPoint2_DINT

Offers a random access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_DINT [▶ 261]
    ITcVnRandomAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 262]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 262]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.
GetAt [▶ 314]	ITcVnRandomAccess_TcVnPoint2_DINT	Gets the value
SetAt [▶ 314]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.10 ITcVnRandomAccess_TcVnPoint2_LREAL

Offers a random access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_LREAL [▶ 263]
    ITcVnRandomAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 263]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 264]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value
GetAt [▶ 316]	ITcVnRandomAccess_TcVnPoint2_LREAL	Gets the value
SetAt [▶ 316]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.11 ITcVnRandomAccess_TcVnPoint2_REAL

Offers a random access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint2_REAL [▶ 264]
    ITcVnRandomAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 265]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 265]	ITcVnAccess_TcVnPoint2_REAL	Sets the value
GetAt [▶ 317]	ITcVnRandomAccess_TcVnPoint2_REAL	Gets the value
SetAt [▶ 318]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.12 ITcVnRandomAccess_TcVnPoint3_LREAL

Offers a random access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint3_LREAL [▶ 266]
    ITcVnRandomAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 266]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 266]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value
GetAt [▶ 319]	ITcVnRandomAccess_TcVnPoint3_LREAL	Gets the value
SetAt [▶ 319]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.13 ITcVnRandomAccess_TcVnPoint3_REAL

Offers a random access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnPoint3_REAL [▶ 267]
    ITcVnRandomAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 267]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 268]	ITcVnAccess_TcVnPoint3_REAL	Sets the value
GetAt [▶ 321]	ITcVnRandomAccess_TcVnPoint3_REAL	Gets the value
SetAt [▶ 321]	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 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 [▶ 160]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 160]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.14 ITcVnRandomAccess_TcVnRectangle_DINT

Offers a random access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnRectangle_DINT [▶ 268]
    ITcVnRandomAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 269]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 269]	ITcVnAccess_TcVnRectangle_DINT	Sets the value
GetAt [▶ 322]	ITcVnRandomAccess_TcVnRectangle_DINT	Gets the value
SetAt [▶ 323]	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 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 [▶ 244]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 244]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.15 ITcVnRandomAccess_TcVnVector2_DINT

Offers a random access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_DINT [▶ 270]
    ITcVnRandomAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 270]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 271]	ITcVnAccess_TcVnVector2_DINT	Sets the value.
GetAt [▶ 324]	ITcVnRandomAccess_TcVnVector2_DINT	Gets the value
SetAt [▶ 324]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.16 ITcVnRandomAccess_TcVnVector2_INT

Offers a random access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_INT [▶ 271]
    ITcVnRandomAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 272]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 272]	ITcVnAccess_TcVnVector2_INT	Sets the value.
GetAt [▶ 326]	ITcVnRandomAccess_TcVnVector2_INT	Gets the value
SetAt [▶ 326]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.17 ITcVnRandomAccess_TcVnVector2_REAL

Offers a random access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_REAL [▶ 273]
    ITcVnRandomAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 273]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 273]	ITcVnAccess_TcVnVector2_REAL	Sets the value
GetAt [▶ 327]	ITcVnRandomAccess_TcVnVector2_REAL	Gets the value
SetAt [▶ 328]	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 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
```

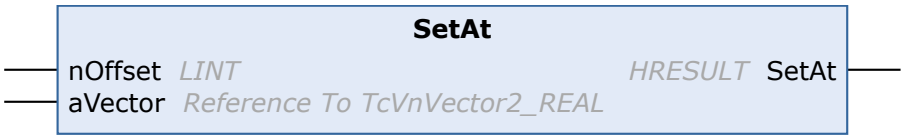
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_REAL [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.18 ITcVnRandomAccess_TcVnVector2_SINT

Offers a random access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_SINT [▶ 274]
    ITcVnRandomAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 274]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 275]	ITcVnAccess_TcVnVector2_SINT	Sets the value.
GetAt [▶ 329]	ITcVnRandomAccess_TcVnVector2_SINT	Gets the value
SetAt [▶ 329]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.19 ITcVnRandomAccess_TcVnVector2_UINT

Offers a random access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_UINT [▶ 275]
    ITcVnRandomAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 276]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 276]	ITcVnAccess_TcVnVector2_UINT	Sets the value.
GetAt [▶ 331]	ITcVnRandomAccess_TcVnVector2_UINT	Gets the value
SetAt [▶ 331]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.20 ITcVnRandomAccess_TcVnVector2_USINT

Offers a random access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector2_USINT [▶ 277]
    ITcVnRandomAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 277]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 278]	ITcVnAccess_TcVnVector2_USINT	Sets the value.
GetAt [▶ 332]	ITcVnRandomAccess_TcVnVector2_USINT	Gets the value
SetAt [▶ 333]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.21 ITcVnRandomAccess_TcVnVector3_INT

Offers a random access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_INT [▶ 278]
    ITcVnRandomAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 279]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 279]	ITcVnAccess_TcVnVector3_INT	Sets the value
GetAt [▶ 334]	ITcVnRandomAccess_TcVnVector3_INT	Gets the value
SetAt [▶ 334]	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 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
```

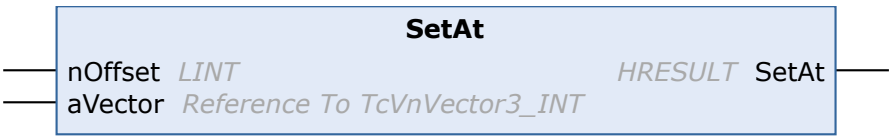
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_INT [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.22 ITcVnRandomAccess_TcVnVector3_REAL

Offers a random access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_REAL [▶ 280]
    ITcVnRandomAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 280]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 281]	ITcVnAccess_TcVnVector3_REAL	Sets the value
GetAt [▶ 336]	ITcVnRandomAccess_TcVnVector3_REAL	Gets the value
SetAt [▶ 336]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.23 ITcVnRandomAccess_TcVnVector3_SINT

Offers a random access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_SINT [▶ 281]
    ITcVnRandomAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 282]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 282]	ITcVnAccess_TcVnVector3_SINT	Sets the value
GetAt [▶ 337]	ITcVnRandomAccess_TcVnVector3_SINT	Gets the value
SetAt [▶ 338]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.24 ITcVnRandomAccess_TcVnVector3_UINT

Offers a random access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_UINT [▶ 283]
    ITcVnRandomAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 283]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 283]	ITcVnAccess_TcVnVector3_UINT	Sets the value
GetAt [▶ 339]	ITcVnRandomAccess_TcVnVector3_UINT	Gets the value
SetAt [▶ 339]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.25 ITcVnRandomAccess_TcVnVector3_USINT

Offers a random access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector3_USINT [▶ 284]
    ITcVnRandomAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 284]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 285]	ITcVnAccess_TcVnVector3_USINT	Sets the value
GetAt [▶ 341]	ITcVnRandomAccess_TcVnVector3_USINT	Gets the value
SetAt [▶ 341]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.26 ITcVnRandomAccess_TcVnVector4_DINT

Offers a random access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_DINT [▶ 285]
    ITcVnRandomAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 286]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 286]	ITcVnAccess_TcVnVector4_DINT	Sets the value
GetAt [▶ 342]	ITcVnRandomAccess_TcVnVector4_DINT	Gets the value
SetAt [▶ 343]	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 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
```

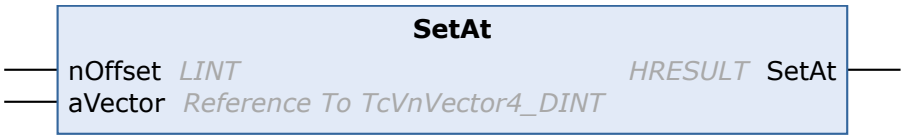
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_DINT [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.27 ITcVnRandomAccess_TcVnVector4_INT

Offers a random access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_INT [▶ 287]
    ITcVnRandomAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 287]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 288]	ITcVnAccess_TcVnVector4_INT	Sets the value
GetAt [▶ 344]	ITcVnRandomAccess_TcVnVector4_INT	Gets the value
SetAt [▶ 344]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.28 ITcVnRandomAccess_TcVnVector4_LREAL

Offers a random access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_LREAL [▶ 288]
    ITcVnRandomAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 289]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 289]	ITcVnAccess_TcVnVector4_LREAL	Sets the value
GetAt [▶ 346]	ITcVnRandomAccess_TcVnVector4_LREAL	Gets the value
SetAt [▶ 346]	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 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
```

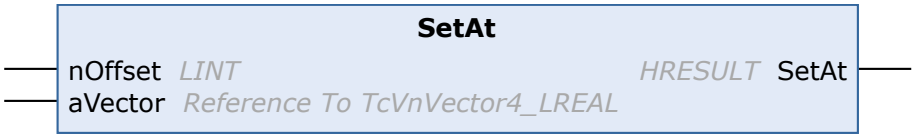
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_LREAL [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.29 ITcVnRandomAccess_TcVnVector4_SINT

Offers a random access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_SINT [▶ 290]
    ITcVnRandomAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 290]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 291]	ITcVnAccess_TcVnVector4_SINT	Sets the value
GetAt [▶ 347]	ITcVnRandomAccess_TcVnVector4_SINT	Gets the value
SetAt [▶ 348]	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 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
```

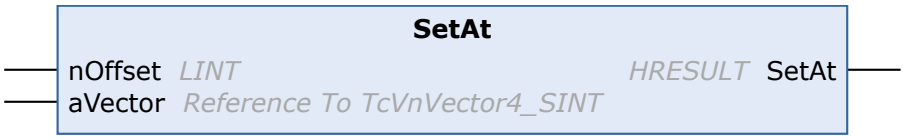
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_SINT [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.30 ITcVnRandomAccess_TcVnVector4_UINT

Offers a random access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_UINT [▶ 291]
    ITcVnRandomAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 291]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 292]	ITcVnAccess_TcVnVector4_UINT	Sets the value
GetAt [▶ 349]	ITcVnRandomAccess_TcVnVector4_UINT	Gets the value
SetAt [▶ 349]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.31 ITcVnRandomAccess_TcVnVector4_USINT

Offers a random access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_USINT [▶ 292]
    ITcVnRandomAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 293]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 293]	ITcVnAccess_TcVnVector4_USINT	Sets the value
GetAt [▶ 351]	ITcVnRandomAccess_TcVnVector4_USINT	Gets the value
SetAt [▶ 351]	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 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
```

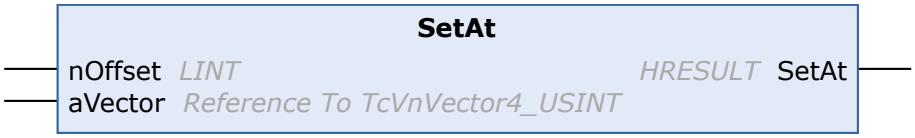
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_USINT [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.3.32 ITcVnRandomAccess_UDINT

Offers a random access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_UDINT [▶ 294]
    ITcVnRandomAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 294]	ITcVnAccess_UDINT	Gets the value
Set [▶ 295]	ITcVnAccess_UDINT	Sets the value
GetAt [▶ 352]	ITcVnRandomAccess_UDINT	Gets the value
SetAt [▶ 353]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.3.33 ITcVnRandomAccess_UINT

Offers a random access interface for UINT values

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_UINT [▶ 295]
    ITcVnRandomAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 296]	ITcVnAccess_UINT	Gets the value
Set [▶ 296]	ITcVnAccess_UINT	Sets the value
GetAt [▶ 354]	ITcVnRandomAccess_UINT	Gets the value
SetAt [▶ 354]	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 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
```

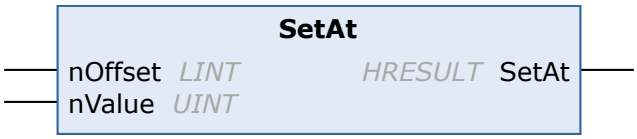
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To UINT	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.1.3.34 ITcVnRandomAccess_ULINT

Offers a randowm access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_ULINT [▶ 297]
    ITcVnRandomAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 297]	ITcVnAccess_ULINT	Gets the value
Set [▶ 297]	ITcVnAccess_ULINT	Sets the value
GetAt [▶ 355]	ITcVnRandomAccess_ULINT	Gets the value
SetAt [▶ 356]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.3.35 ITcVnRandomAccess_USINT

Offers a random access interface for USINT values

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_USINT [▶ 298]
    ITcVnRandomAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 298]	ITcVnAccess_USINT	Gets the value
Set [▶ 299]	ITcVnAccess_USINT	Sets the value
GetAt [▶ 357]	ITcVnRandomAccess_USINT	Gets the value
SetAt [▶ 357]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.1.3.36 ITcVnRandomAccess_TcVnVector4_REAL

Offers a random access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnAccess_TcVnVector4_REAL [▶ 299]
    ITcVnRandomAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Get [▶ 300]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 300]	ITcVnAccess_TcVnVector4_REAL	Sets the value
GetAt [▶ 358]	ITcVnRandomAccess_TcVnVector4_REAL	Gets the value
SetAt [▶ 359]	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 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 [▶ 162]	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 162]	The value to set

Return value

HRESULT [▶ 141]

6.1.3.1.4 Iterators

This group contains interfaces for the handling of iterators in order to access elements of containers [▶ 152].

6.1.3.1.4.1 ITcVnBidirectionalIterator

Offers an interface for a bidirectional iterator.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnIteratorBase [▶ 363]
    ITcVnForwardIterator [▶ 360]
      ITcVnBidirectionalIterator

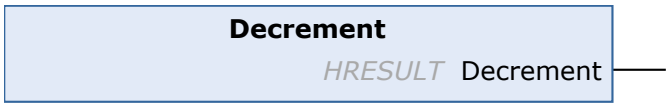
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 364]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 365]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 365]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 365]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 361]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 362]	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 [▶ 362]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 363]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 360]	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 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 [▶ 141]

6.1.3.1.4.2 ITcVnForwardIterator

Offers an interface for a forward iterator.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnIteratorBase [▶ 363]
    ITcVnForwardIterator

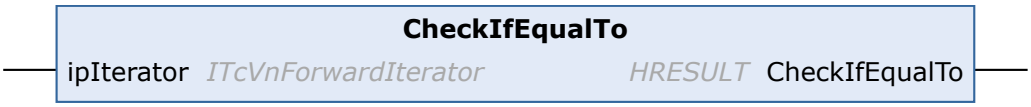
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 364]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 365]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 365]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 365]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 361]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 362]	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 [▶ 362]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 363]	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 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
```

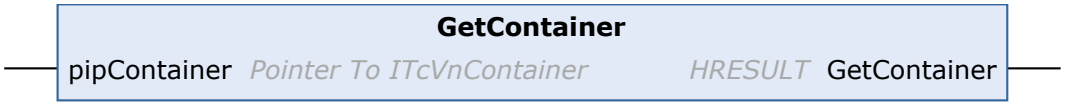
Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 360]	Iterator interface to compare with

Return value

HRESULT [▶ 141]

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
```

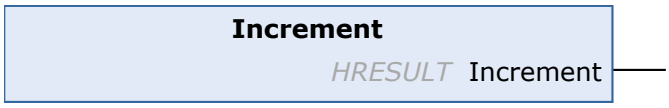
Inputs

Name	Type	Description
pipContainer	Pointer To ITcVnContainer [▶ 370]	Returns the container interface.

Return value

HRESULT [▶ 141]

6.1.3.1.4.2.3 Increment



Increments the iterator. (Alternatively use F_VN_IncrementIterator.)

Syntax

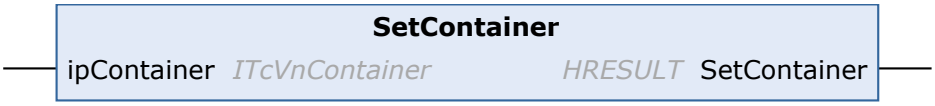
Definition:

```
METHOD Increment : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 370]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 141]

6.1.3.1.4.3 ITcVnIteratorBase

Offers a base interface for iterators

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnIteratorBase

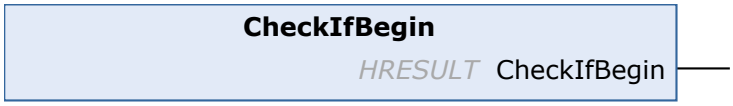
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 364]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 365]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 365]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 365]	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 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

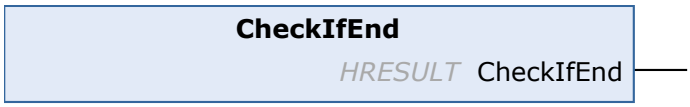
Definition:

```
METHOD CheckIfBegin : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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
```

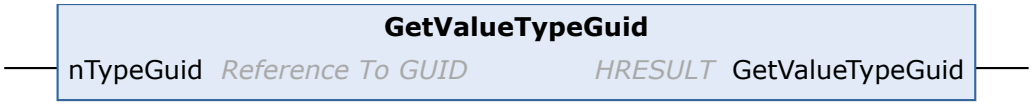
Inputs

Name	Type	Description
nSize	Reference To ULINT	Returns the element size in bytes.

Return value

HRESULT [▶ 141]

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
```

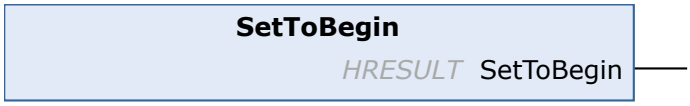
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the type GUID.

Return value

HRESULT [▶ 141]

6.1.3.1.4.3.5 SetToBegin



Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).

Syntax

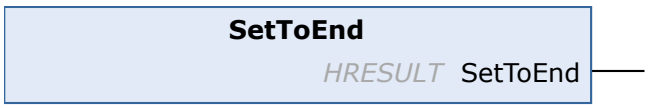
Definition:

```
METHOD SetToBegin : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 428]
  ITcVnIteratorCopyCreator

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
Create [▶ 366]	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 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 [▶ 360]	Returns the created iterator.

Return value

HRESULT [▶ 141]

6.1.3.1.4.5 ITcVnRandomAccessIterator

Offers an interface for a random access iterator.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnIteratorBase [▶ 363]
    ITcVnForwardIterator [▶ 360]
      ITcVnBidirectionalIterator [▶ 359]
        ITcVnRandomAccessIterator

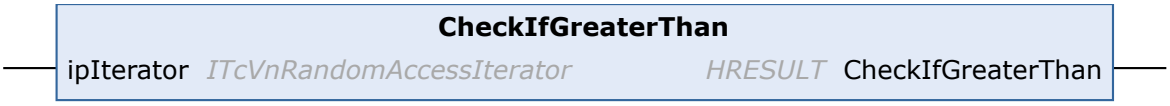
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 364]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 364]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 365]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 365]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 365]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 361]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 362]	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 [▶ 362]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 363]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 360]	ITcVnBidirectionalIterator	Decrements the iterator.
CheckIfGreaterThan [▶ 368]	ITcVnRandomAccessIterator	Checks if the iterator is greater than another iterator.
CheckIfLessThan [▶ 369]	ITcVnRandomAccessIterator	Checks if the iterator is less than another iterator.
GetContainerAt [▶ 369]	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 [▶ 370]	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 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
```

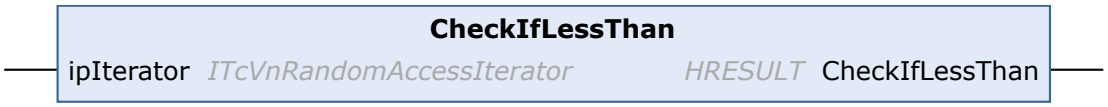
Inputs

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator [▶ 367]	Iterator interface to compare with.

Return value

HRESULT [▶ 141]

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
```

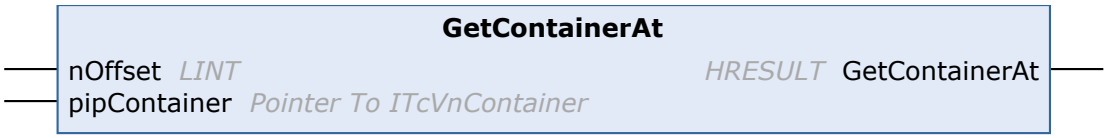
Inputs

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator [▶ 367]	Iterator interface to compare with.

Return value

HRESULT [▶ 141]

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
```

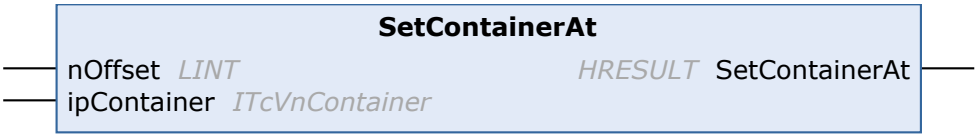
Inputs

Name	Type	Description
nOffset	LINT	Offset from the current element.
pipContainer	Pointer To ITcVnContainer [▶ 370]	Returns the pointer to the container interface.

Return value

HRESULT [▶ 141]

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 [▶ 370]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 141]

6.1.3.1.5 ITcVnContainer

Offers an interface for an object container.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnContainer

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBasicContainer [▶ 372]	ITcVnContainer	Checks if the container contains only basic (non-container) elements.
CheckIfEmpty [▶ 372]	ITcVnContainer	Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)
GetBidirectionalIterator [▶ 372]	ITcVnContainer	Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.
GetElementNum [▶ 373]	ITcVnContainer	Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)
GetElementSize [▶ 373]	ITcVnContainer	Gets the size (in byte) of each element in the container.
GetElementTypeGuid [▶ 374]	ITcVnContainer	Gets the GUID of the container elements.
GetExportSize [▶ 374]	ITcVnContainer	Gets combined size (in byte) of all elements in the container.
GetForwardIterator [▶ 374]	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 [▶ 375]	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 [▶ 375]	ITcVnContainer	Gets GUID of the container.
GetTypeName [▶ 376]	ITcVnContainer	Gets the container type name as a string.

Further information

The interface `ITcVnContainer` is for handling containers [▶ 152]. Observe the notes in the chapter Interface pointers [▶ 137].

Overwriting containers

Generally, containers with API functions can simply be overwritten. At present, however, overwriting is not possible if the existing container and the container to be written are of different types. This is signaled by the error code 70E (INCOMPATIBLE).

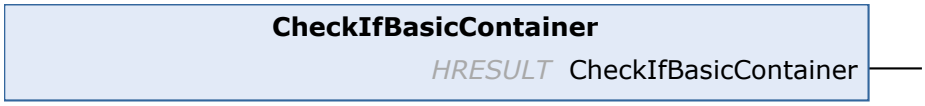
Related functions

- Basic Container Operations [▶ 547]
- Algebraic Container Operations [▶ 464]
- Container statistics [▶ 861]
- Contour analysis [▶ 955]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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

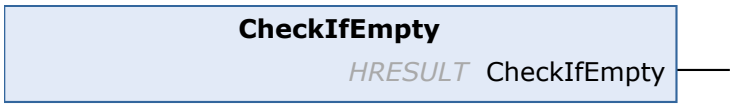
Definition:

```
METHOD CheckIfBasicContainer : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.1.5.2 CheckIfEmpty



Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)

Syntax

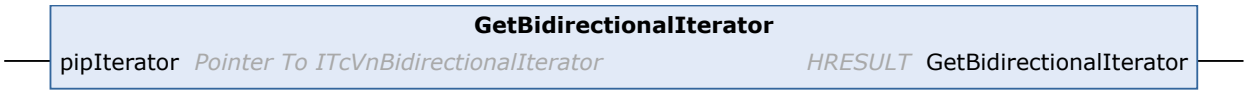
Definition:

```
METHOD CheckIfEmpty : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 359]	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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
```

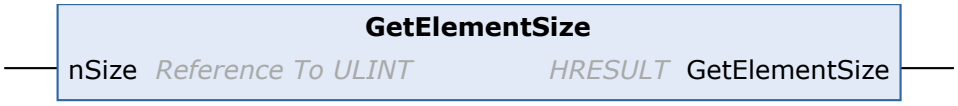
Inputs

Name	Type	Description
nElementNum	Reference To ULINT	Returns the number of elements in the container.

Return value

HRESULT [▶ 141]

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
```

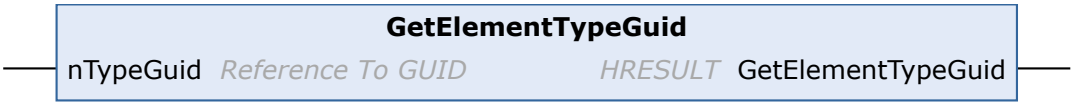
Inputs

Name	Type	Description
nSize	Reference To ULINT	Returns the size in byte of a single element in the container.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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
```

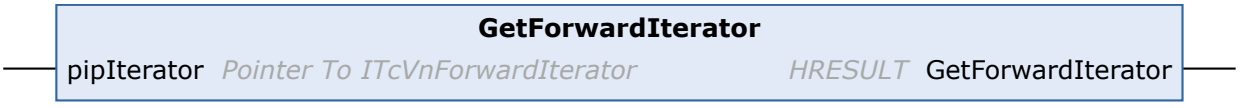
Inputs

Name	Type	Description
nExportSize	Reference To ULINT	Returns the combined size (in byte) of all elements in the container.

Return value

HRESULT [▶ 141]

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
```

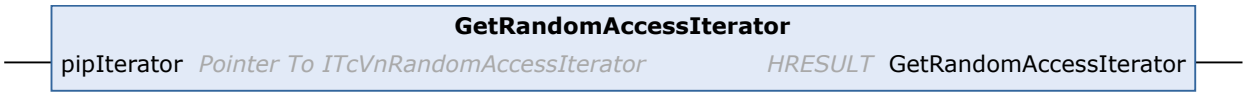
Inputs

Name	Type	Description
pipIterator	Pointer To ITcVnForwardIterator [▶ 360]	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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 [▶ 367]	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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
```

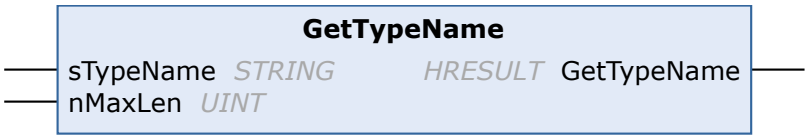
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the GUID of the container.

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.2 Images

This group contains interfaces for recording and handling images [▶ 149].

6.1.3.2.1 Acquisition

This group contains interfaces for handling the image acquisition.

6.1.3.2.1.1 ITcIoFileImageAcquisition

Interface for file image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcIoImageAcquisition [▶ 381]
    ITcIoFileImageAcquisition

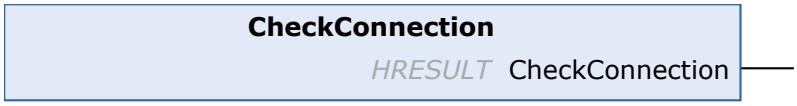
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 381]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 382]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 382]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 382]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 383]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 383]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 383]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 384]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 377]	ITcIoFileImageAcquisition	Checks the connection
TriggerImage [▶ 378]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 378]	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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.1.1 CheckConnection



Checks the connection

Syntax

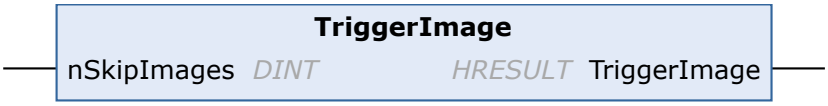
Definition:

```
METHOD CheckConnection : HRESULT
```

Return value

HRESULT [▶ 141]

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
```

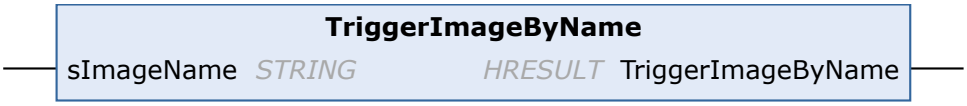
Inputs

Name	Type	Description
nSkipImages	DINT	Number of images to skip

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.2.1.2 ITcIoFileImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcIoImageRecv [▶ 384]
    ITcIoFileImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 385]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 385]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.
ReceiveImage [▶ 379]	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 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 [▶ 414]	Interface pointer to the image.
sFileName	STRING	Returns the filename to the image.
hrAcquisitionResult	HRESULT [▶ 141]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 141]

6.1.3.2.1.3 ITcIoGevImageAcquisition

Interface for GigE Vision image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcIoImageAcquisition [▶ 381]
    ITcIoGevImageAcquisition

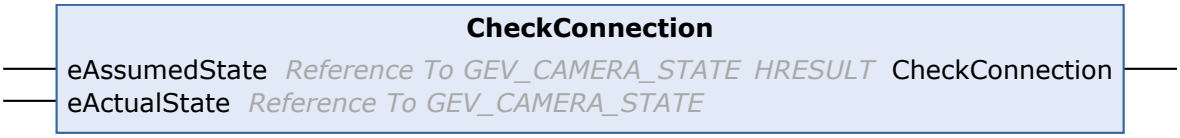
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 381]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 382]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 382]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 382]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 383]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 383]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 383]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 384]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 380]	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 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 [▶ 224]	The internally assumed state of the camera.
eActualState	Reference To GEV_CAMERA_STATE [▶ 224]	The actually observable state of the camera.

Return value

HRESULT [▶ 141]

6.1.3.2.1.4 ITcIoImageAcquisition

Interface for image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcIoImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 381]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 382]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 382]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 382]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 383]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 383]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 383]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 384]	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 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
```

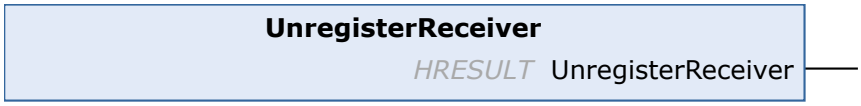
Inputs

Name	Type	Description
ipRecv	ITcIoImageRecv [▶ 384]	Pointer to an interface containing the callback function.

Return value

HRESULT [▶ 141]

6.1.3.2.1.4.2 UnregisterReceiver



Unregister callback interface.

Syntax

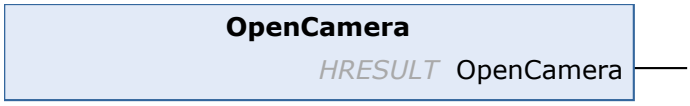
Definition:

```
METHOD UnregisterReceiver : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.4.3 OpenCamera



Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

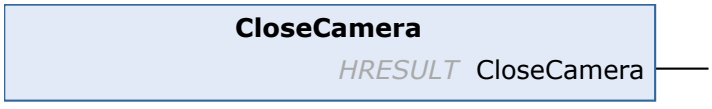
Definition:

```
METHOD OpenCamera : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.4.4 CloseCamera



Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax

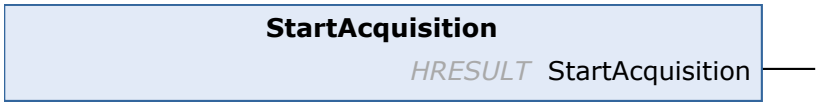
Definition:

```
METHOD CloseCamera : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.4.5 StartAcquisition



Start the image acquisition.

Syntax

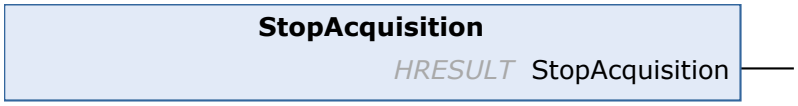
Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.4.6 StopAcquisition



Stop the image acquisition.

Syntax

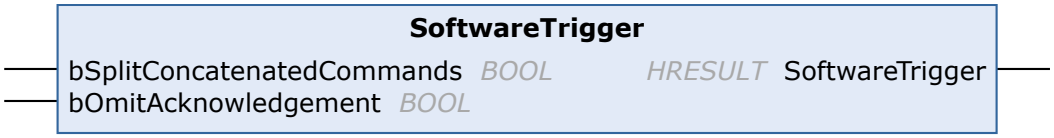
Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 141]

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
```

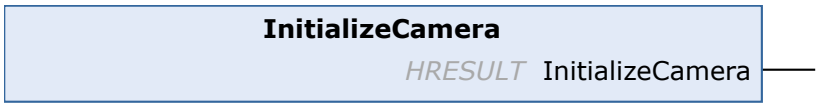
Inputs

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 [▶ 141]

6.1.3.2.1.4.8 InitializeCamera



Send initialization commands to the camera.

Syntax

Definition:

```
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.5 ITcIoImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcIoImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 385]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 385]	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 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 [▶ 414]	Interface pointer to the image.
hrAcquisitionResult	HRESULT [▶ 141]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 141]

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 [▶ 141]	HRESULT indicating the status of the performed operation.

Return value

HRESULT [▶ 141]

6.1.3.2.1.6 ITcVnFileImageProvider

Interface for an image provider for images from outside realtime environment.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageProvider [▶ 390]
    ITcVnFileImageProvider

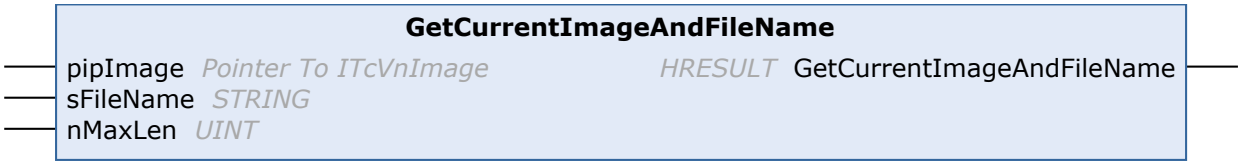
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 391]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 391]	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 [▶ 392]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 392]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 393]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 393]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 393]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 393]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 394]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 394]	ITcVnImageProvider	Set camera to initial state.
GetCurrentImageAndFileName [▶ 387]	ITcVnFileImageProvider	Get the current image by detaching its internal reference.
TriggerImage [▶ 387]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 388]	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 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
```

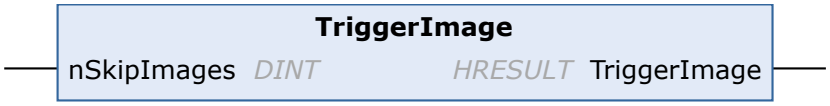
Inputs

Name	Type	Description
pipImage	Pointer To ITcVnImage [▶ 411]	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 [▶ 141]

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
```

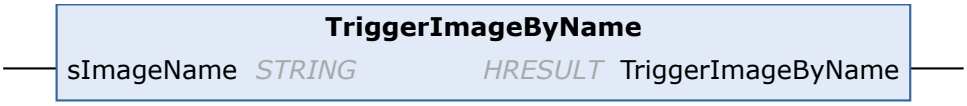
Inputs

Name	Type	Description
nSkipImages	DINT	Number of images to skip

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.2.1.7 ITcVnGevImageProvider

Interface for a GigE Vision image provider.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageProvider [▶ 390]
    ITcVnGevImageProvider

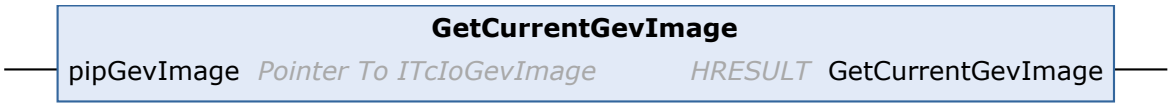
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 391]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 391]	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 [▶ 392]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 392]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 393]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 393]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 393]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 393]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 394]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 394]	ITcVnImageProvider	Set camera to initial state.
GetCurrentGevImage [▶ 389]	ITcVnGevImageProvider	Gets the current GigE Vision image by detaching its internal reference.
GetCurrentImageWithGvspInfo [▶ 390]	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 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 [▶ 407]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 141]

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 [▶ 411]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.
stGvspInfo	Reference To GVSP_IMAGE_INFO [▶ 225]	Contains useful meta information like image id, time stamp etc.

Return value

HRESULT [▶ 141]

6.1.3.2.1.8 ITcVnImageProvider

Interface for an image provider.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageProvider

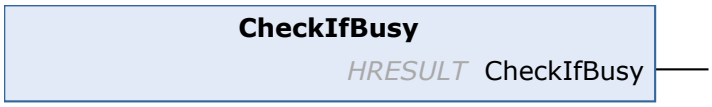
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 391]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 391]	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 [▶ 392]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 392]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 393]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 393]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 393]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 393]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 394]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 394]	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 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

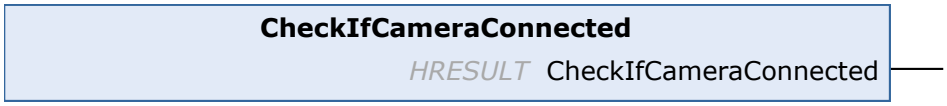
Definition:

```
METHOD CheckIfBusy : HRESULT
```

Return value

HRESULT [▶ 141]

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

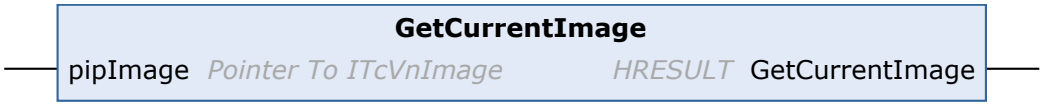
Definition:

```
METHOD CheckIfCameraConnected : HRESULT
```

Return value

HRESULT [▶ 141]

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
```

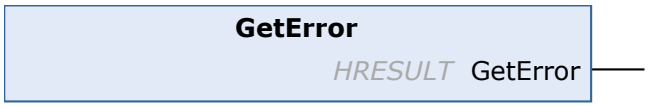
Inputs

Name	Type	Description
pipImage	Pointer To ITcVnImage [▶ 411]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 141]

6.1.3.2.1.8.4 GetError



Gets result of the last operation performed.

Syntax

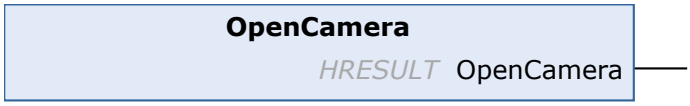
Definition:

```
METHOD GetError : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.8.5 OpenCamera



Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

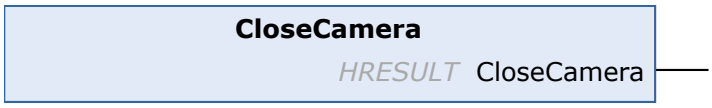
Definition:

```
METHOD OpenCamera : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.2.1.8.7 StartAcquisition



Start image acquisition.

Syntax

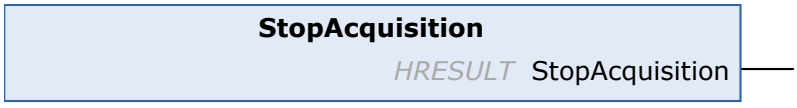
Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.8.8 StopAcquisition



Stop image acquisition.

Syntax

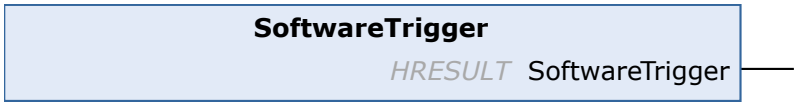
Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 141]

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

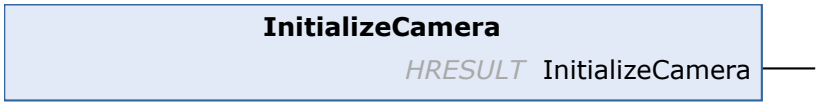
Definition:

```
METHOD SoftwareTrigger : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.1.8.10 InitializeCamera



Set camera to initial state.

Syntax

Definition:

```
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.3.2.2 Export

This group contains interfaces for handling the data export in connection with images.

6.1.3.2.2.1 ITcVnBitmapExport

Interface for exporting an image as a Windows Bitmap (BMP).

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnBitmapExport

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 395]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 396]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.

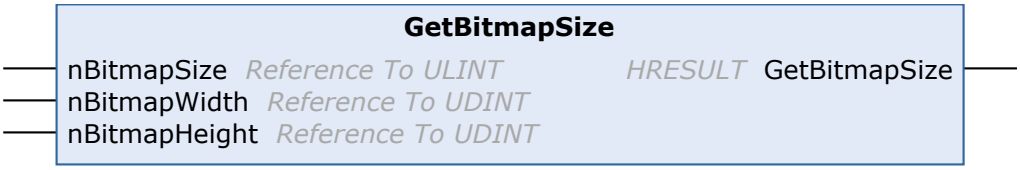
Further information

By definition, a bitmap can have only 8 bits (for 1-channel images) or 24 bits (for 3 and 4-channel images). Therefore, 16-bit images as well as images with the element type REAL or LREAL are converted for the bitmap export. In the case of 16-bit images the value range is simply scaled down to 8-bit (0 corresponds to 0 and 255 corresponds to 65,280). In the case of (L)REAL images the value range [-1, +1] is scaled to [-127, +127] and transferred as a signed integer. Values outside these limits will be interpreted as -1 or +1 respectively.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
```

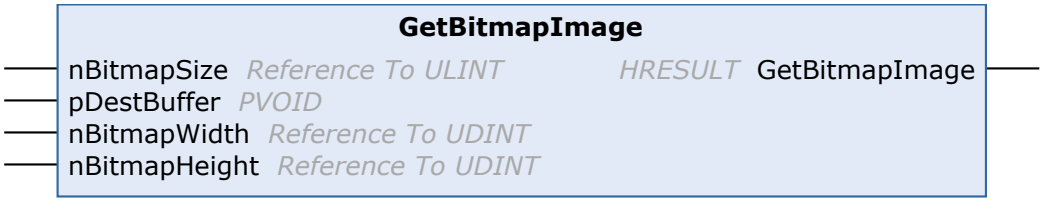
Inputs

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 [▶ 141]

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 [▶ 141]

6.1.3.2.2.2 ITcVnBitmapExportNotification

Interface required for sending displayable images as ADS notifications

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnBitmapExportNotification

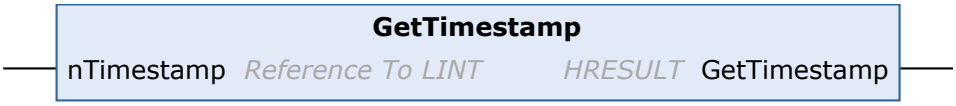
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 397]	ITcVnBitmapExportNotification	Gets the timestamp of the latest image change.
GetBitmapImageRpcUnlocked [▶ 397]	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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.1 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 [▶ 141]

6.1.3.2.2.2 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 [▶ 141]

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 [▶ 428]
  ITcVnBitmapExport [▶ 394]
    ITcVnBitmapExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 395]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 396]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.
GetBitmapImageRpcUnlocked [▶ 399]	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.

Further information

By definition, a bitmap can have only 8 bits (for 1-channel images) or 24 bits (for 3 and 4-channel images). Therefore, 16-bit images as well as images with the element type REAL or LREAL are converted for the bitmap export. In the case of 16-bit images the value range is simply scaled down to 8-bit (0 corresponds to 0 and 255 corresponds to 65,280). In the case of (L)REAL images the value range [-1, +1] is scaled to [-127, +127] and transferred as a signed integer. Values outside these limits will be interpreted as -1 or +1 respectively.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]

6.1.3.2.2.4 ITcVnHistogramExport

Interface for exporting a histogram for an image as an array.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnHistogramExport

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetHistogramSize [▶ 400]	ITcVnHistogramExport	Gets the size of the histogram array.
GetHistogramArray [▶ 400]	ITcVnHistogramExport	Gets the histogram array.
GetCustomHistogramSize [▶ 401]	ITcVnHistogramExport	Gets the size of a custom histogram array.
GetCustomHistogramArray [▶ 402]	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 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 [▶ 141]

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
```

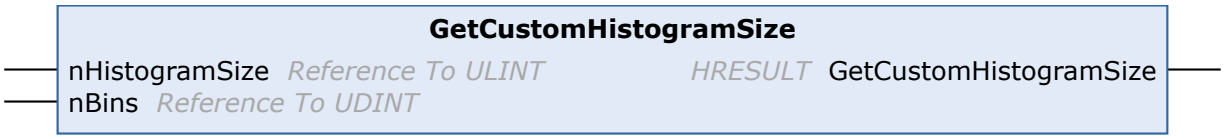
Inputs

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 [▶ 141]

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
```

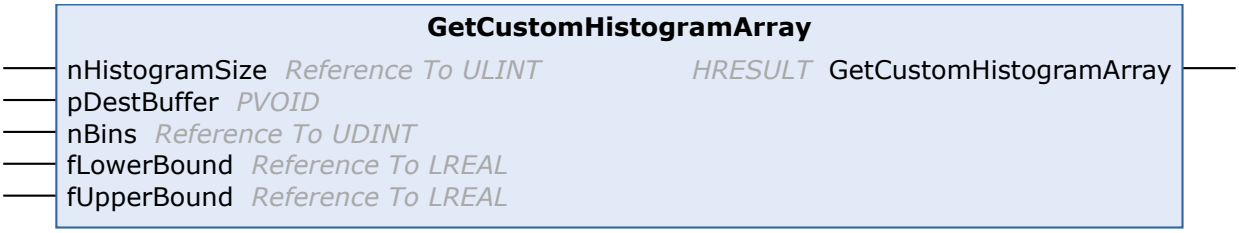
Inputs

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 [▶ 141]

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 [▶ 141]

6.1.3.2.2.5 ITcVnTiffExport

Interface for exporting an image as tiff.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnTiffExport

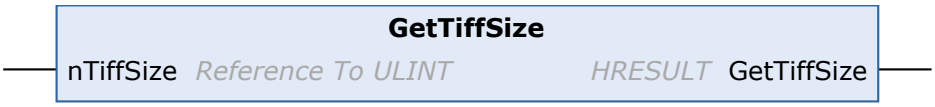
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 403]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 403]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.2.2.6 ITcVnTiffExportNotification

Interface for sending displayable images as tiff via ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnTiffExportNotification

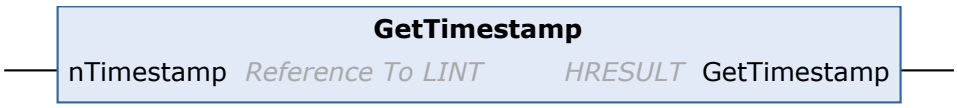
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 404]	ITcVnTiffExportNotification	Gets the timestamp of the latest image change.
GetTiffImageRpcUnlocked [▶ 405]	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 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
```

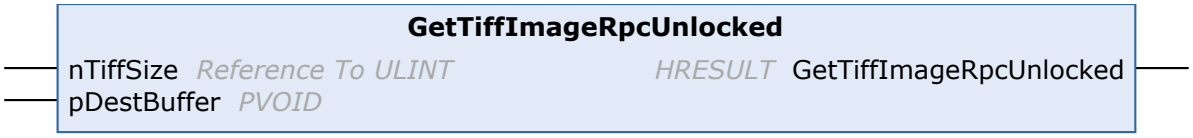
Inputs

Name	Type	Description
nTimestamp	Reference To LINT	Returns the timestamp.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 428]
  ITcVnTiffExport [▶ 402]
    ITcVnTiffExportRpcUnlocked

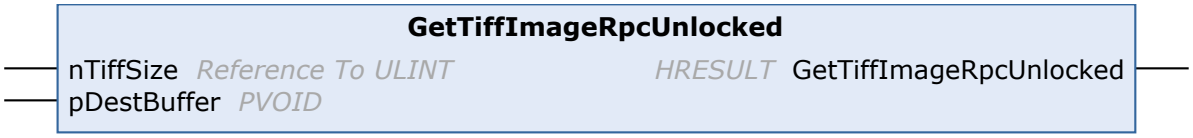
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 403]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 403]	ITcVnTiffExport	Export the image as tiff into a given buffer.
GetTiffImageRpcUnlocked [▶ 406]	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 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 [▶ 141]

6.1.3.2.3 Import

This group contains interfaces for handling the data import in connection with images.

6.1.3.2.3.1 ITcVnTiffImport

Interface for importing a tiff image.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnTiffImport

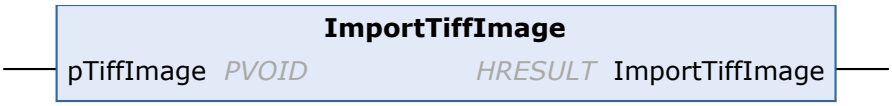
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
ImportTiffImage [▶ 407]	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 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 [▶ 141]

6.1.3.2.4 ITcIoGevImage

Interface for GigE Vision images.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageBase [▶ 414]
    ITcIoGevImage

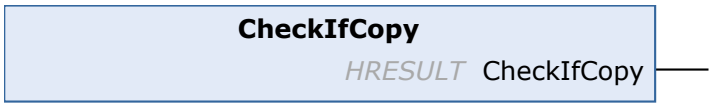
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 415]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 415]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 416]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 416]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 417]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 417]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 417]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 418]	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 [▶ 408]	ITcIoGevImage	Check if image data is a copy.
DecreaseHeight [▶ 409]	ITcIoGevImage	Sets image height to newHeight if it is smaller than the current value.
GetBlockId [▶ 409]	ITcIoGevImage	Gets the GVSP block ID.
GetGevStatus [▶ 409]	ITcIoGevImage	Gets the block status.
GetGvspImageInfo [▶ 410]	ITcIoGevImage	Gets a pointer to the GVSP leader payload.
SetGevStatus [▶ 410]	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 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

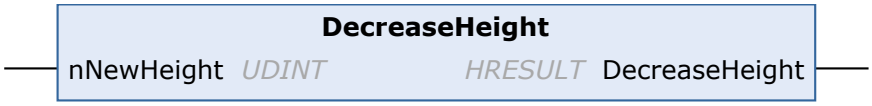
Definition:

```
METHOD CheckIfCopy : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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
```

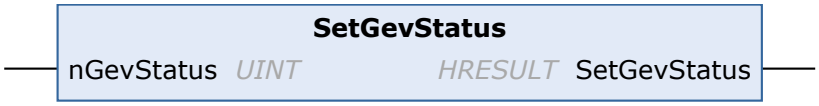
Inputs

Name	Type	Description
ppGvspImageInfo	Pointer To Pointer To GVSP_IMAGE_INFO [▶ 225]	Pointer to the GVSP meta information.

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.2.5 ITcVnDisplayableImage

Interface for displayable images.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnDisplayableImage

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.

Further information

This interface is used to display images via ADS (e.g. with ADS Image Watch). It is not suitable for processing images. The ITcVnImage [▶ 411] interface is required to process images. For more information, see Distinction between processable and displayable images [▶ 149].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.6 ITcVnImage

Basic interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageBase [▶ 414]
    ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 415]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 415]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 416]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 416]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 417]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 417]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 417]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 418]	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 [▶ 412]	ITcVnImage	Initialize an image with an ITcVnImageBase interface.
GetRowPointer [▶ 413]	ITcVnImage	Gets a pointer to a specific row of an image.
ReleaseRowPointer [▶ 414]	ITcVnImage	Release the pointer to a specific row of an image.

Further information

The interface `ITcVnImage` is for handling images [▶ 149]. It provides methods for obtaining metadata such as image size, number of channels, pixel type etc. and for accessing the raw image data.

This interface is not suitable for image display via ADS (e.g. with ADS Image Watch [▶ 131]). The ITcVnDisplayableImage [▶ 411] interface is required for image display via ADS. A conversion can be carried out with the following functions:

- F_VN_CopyIntoDisplayableImage [▶ 781]
- F_VN_TransformIntoDisplaybleImage [▶ 812]

Related functions

- Basic Image Operations [▶ 773]
- Algebraic Image Operations [▶ 506]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 414]	Image from which to obtain the data.

Return value

HRESULT [▶ 141]

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 [▶ 141]

Further information

Make sure that the describing pointer is of the same elementary type as the pixel type of the image.

NOTICE

Enabling an Row Pointer

If you create a pointer with GetRowPointer, you must also release it again with the method ReleaseRowPointer!

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 [▶ 141]

6.1.3.2.7 ITcVnImageBase

Base interface for all image types.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnImageBase

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 415]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 415]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 416]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 416]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 417]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 417]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 417]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 418]	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.)

Further information

The interface `ITcVnImageBase` is used for internal purposes and is not normally needed by the user.

For handling images [▶ 149] use the ITcVnImage [▶ 411] interface instead.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 423]	Returns a pointer to the image data interface.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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
```

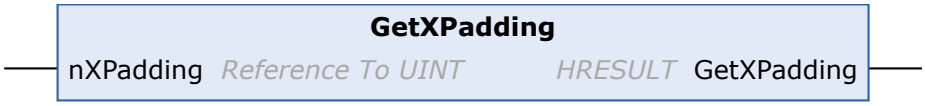
Inputs

Name	Type	Description
nHeight	Reference To UDINT	Returns the image height in pixels.

Return value

HRESULT [▶ 141]

6.1.3.2.7.5 GetXPadding



Gets the horizontal padding.

Syntax

Definition:

```
METHOD GetXPadding : HRESULT
VAR_INPUT
    nXPadding : Reference To UINT;
END_VAR
```

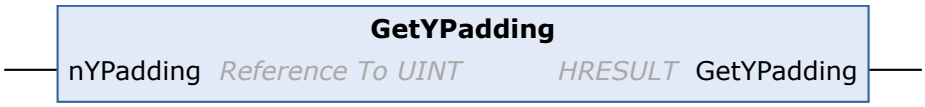
Inputs

Name	Type	Description
nXPadding	Reference To UINT	Returns the horizontal padding in bytes.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 243]	Returns a struct describing the pixel format.

Return value

HRESULT [▶ 141]

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 [▶ 230]	Returns a struct describing the image.

Return value

HRESULT [▶ 141]

6.1.3.3 Machine Learning

This group contains interfaces for machine learning.

6.1.3.3.1 ITcVnColorModel

Interface to train and classify an image color.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnMlModel [▶ 420]
    ITcVnColorModel

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 421]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 421]	ITcVnMlModel	Gets the model type name as a string.
TrainImageColor [▶ 419]	ITcVnColorModel	Train the color of the provided image.
ClassifyImageColor [▶ 420]	ITcVnColorModel	Compute the similarity of each image pixel to the trained reference color.

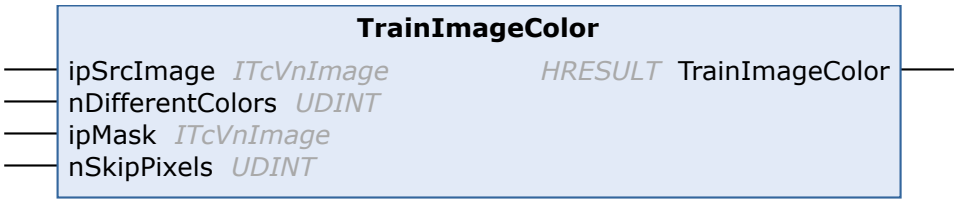
Further information

The interface `ITcVnColorModel` is for describing a color area. The model can be created or adjusted with the function F_VN_TrainImageColor [▶ 1261] and compared with the colors of an image using the function F_VN_ReferenceColorSimilarity_ITcVnColorModel [▶ 1254].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image.
nDifferentColors	UDINT	Number of different colors to differentiate.
ipMask	ITcVnImage [▶ 411]	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 [▶ 141]

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 [▶ 411]	Source image.
pipDestImage	Pointer To ITcVnImage [▶ 411]	Returns the color similarity.
fVariance	REAL	Allowed color variance.
fLuminanceWeight	REAL	Weight the impact of the luminance.

Return value

HRESULT [▶ 141]

6.1.3.3.2 ITcVnMlModel

Interface for a machine learning model.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnMlModel

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 421]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 421]	ITcVnMlModel	Gets the model type name as a string.

Further information

The interface `ITcVnMlModel` is for handling Machine Learning models. At present this only includes the use of the interface ITcVnColorModel [▶ 418].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
```

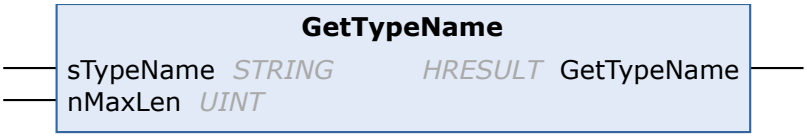
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the GUID of the model.

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.4 Miscellaneous

This group contains other interfaces.

6.1.3.4.1 ITcSerializableNotification

Interface required for sending serialized objects as ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcSerializableNotification

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetComparisonValue [▶ 422]	ITcSerializableNotification	Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).
SerializeRpcUnlocked [▶ 423]	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 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
```

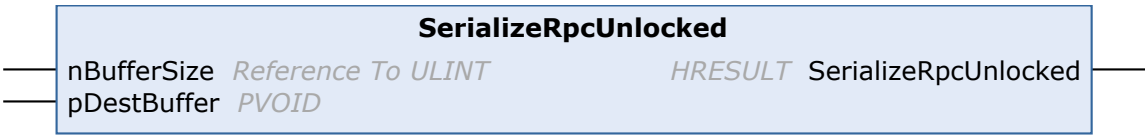
Inputs

Name	Type	Description
nComparisonValue	Reference To LINT	Returns the comparison value.

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.1.3.4.2 ITcVnData

Interface for accessing data.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnData

Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
CheckIfCopy [▶ 424]	ITcVnData	Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).
CheckIfImage [▶ 424]	ITcVnData	Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).
GetSize [▶ 425]	ITcVnData	Gets the size of the data.
CheckDataPointer [▶ 425]	ITcVnData	Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).
GetDataPointer [▶ 425]	ITcVnData	Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).
ReleaseDataPointer [▶ 426]	ITcVnData	Release the data pointer.

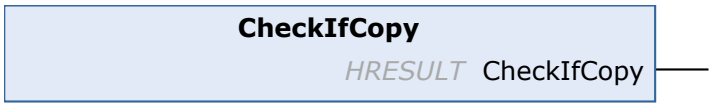
Further information

The interface `ITcVnData` is used for internal purposes and is not normally needed by the user. For access to raw image data, use the function `GetRowPointer` of the interface ITcVnImage [▶ 411] instead.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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

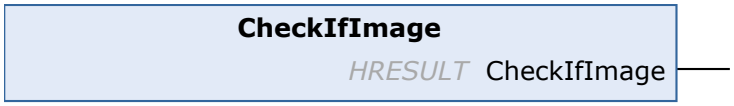
Definition:

```
METHOD CheckIfCopy : HRESULT
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.1.3.4.3 ITcVnTimestamp

Offers an interface for timestamps.

Inheritance Hierarchy

ITcUnknown [▶ 428]
    ITcVnTimestamp

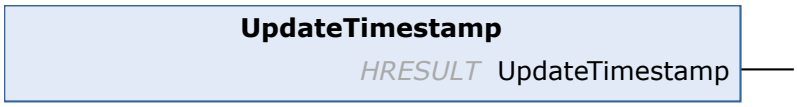
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
UpdateTimestamp [▶ 427]	ITcVnTimestamp	Updates the timestamp to the current time
GetTimestamp [▶ 427]	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 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 [▶ 141]

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 [▶ 141]

6.1.3.4.4 Interface ITcUnknown

ITcUnknown defines the reference count as well as the querying of a reference to a more specific interface.

Syntax

```
TCOM_DECL_INTERFACE("00000001-0000-0000-e000-000000000064", ITcUnknown)
```

Declared in: TcInterfaces.h

Required include: -

Methods

Name	Description
TcAddRef [▶ 428]	Increments the reference counter.
TcQueryInterface [▶ 428]	Query of the reference to an implemented interface via the IID.
TcRelease [▶ 429]	Decrements the reference counter.

Every TcCOM interface is derived directly or indirectly from ITcUnknown. As a result, every TcCOM module class implements ITcUnknown because it is derived from ITComObject.

The standard implementation of ITcUnknown ensures that the object is deleted after the last reference is released. For this reason, an interface pointer must not be dereferenced after calling TcRelease().

6.1.3.4.4.1 Method ITcUnknown:TcAddRef

Increments the reference counter and returns the new value.

Syntax

```
ULONG TcAddRef( )
```

Return value

Resulting reference count value.

6.1.3.4.4.2 Method ITcUnknown:TcQueryInterface

Query of an interface pointer with regard to an interface that is given by interface ID (IID).

Syntax

```
HRESULT TcQueryInterface(RITCID iid, PPVOID pipItf )
```

Return value

If successful, S_OK ("0") or another positive value will be returned, cf. Return values. Extended messages refer in particular to the column HRESULT in ADS Return Codes [▶ 2879].

If the demanded interface is not available, the method returns ADSERR_DEVICE_NOINTERFACE.

Parameter

Name	Type	Description
iid	RITCID	Interface IID.
pipItf	PPVOID	Pointer to interface pointer. Is set when the requested interface type is available from the corresponding instance.

Description

Query of the reference to an implemented interface via the IID. It is recommended to use Smart Pointers to initialize and hold interface pointers.

Variant 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;
}
}
```

The interface ID associated with the Smart Pointer can be used as a parameter in TcQueryInterface. The "&" operator will return the pointer to the internal interface pointer member variable of the Smart Pointer. Variant 1 assumes that the interface pointer is initialized when TcQueryInterface indicates success. If the range remains, then the destructor of the Smart Pointer spObj releases the reference.

Variant 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;
}
```

If the interface pointer ip is assigned to the Smart Pointer spObj, then the TcQueryInterface method is called implicitly with IID_ITComObject on the instance to which ip refers. This results in a shorter code, but the original return code of TcQueryInterface is lost.

6.1.3.4.4.3 Method ITcUnknown:TcRelease

This method decrements the reference counter.

Syntax

```
ULONG TcRelease( )
```

Return value

Resulting reference count value.

Description

Decrements the reference counter and returns the new value.

If the reference counter is 0, the object deletes itself.

6.1.3.5 Neural Network

6.1.3.5.1 ITcVnNeuralNetwork

Interface for neural networks.

Inheritance Hierarchy

ITcUnknown [▶ 428]
  ITcVnMlModel [▶ 420]
    ITcVnNeuralNetwork

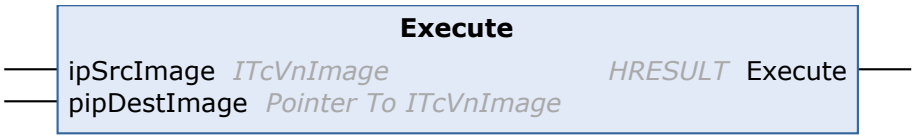
Methods

Name	Origin	Description
TcAddRef [▶ 428]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 428]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 429]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 421]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 421]	ITcVnMlModel	Gets the model type name as a string.
Execute [▶ 430]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from the last output layer. The method doesn't perform any pre-processing or transformation.
Execute_SingleLayerOutput [▶ 431]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from a specified output layer. The method doesn't perform any pre-processing or transformation.
Execute_MultiLayerOutput [▶ 432]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from 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 [▶ 432]	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 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 from the last 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 [▶ 411]	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 [▶ 411]	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 [▶ 141]

6.1.3.5.1.2 Execute_SingleLayerOutput



Execute a neural network using the provided image as input and provide the output from a 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
```

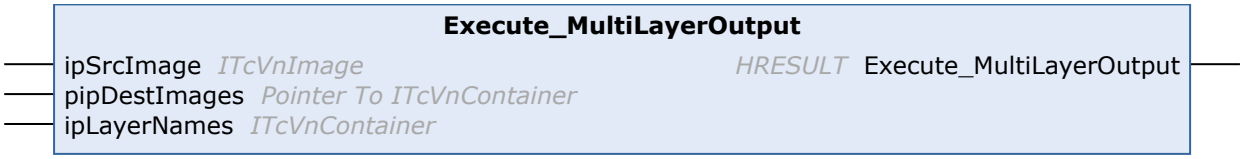
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 411]	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 [▶ 411]	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 the last output layer is provided.

Return value

HRESULT [▶ 141]

6.1.3.5.1.3 Execute_MultiLayerOutput



Execute a neural network using the provided image as input and provide the output from 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
```

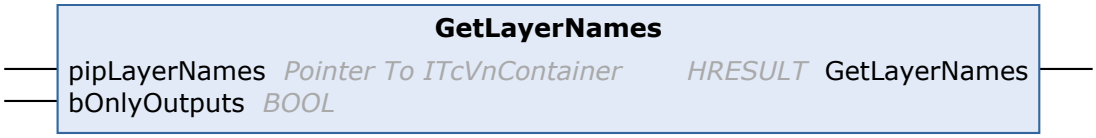
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 411]	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 [▶ 370]	Container of images containing the output from the specified layers
ipLayerNames	ITcVnContainer [▶ 370]	Container of layer names whose outputs are required (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.

Return value

HRESULT [▶ 141]

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 [▶ 370]	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	BOOL	If true, only the unconnected output layer names are returned

Return value

HRESULT [▶ 141]

6.1.4 Functions

The image processing algorithms are encapsulated in functions. More complex algorithms have been implemented as function blocks and can be found at ImageProcessing [▶ 1652].

Structure of a function call

Functions are structured as follows:

```
hr := F_VN_ProcessImage(<...>, hr);
```

Status variable of type **HRESULT**
Naming convention of vision API functions
Parameters and return values as references

hr	Status variable [▶ 141] of the type HRESULT. The execution status of every function can be followed with this.
F_VN_	Functions are identified by the prefix "F_VN_".
Parameter <...>	Function parameters essentially fulfill the following purpose: • Input parameters (e.g. images or containers) that are to be processed. • Parameters for setting the algorithm • Parameters as reference for returning the results

Grouping of functions

All functions of the TwinCAT Vision library are sorted thematically into the following groups:

Images

- Basic Image Operations [▶ 773]
- Algebraic Image Operations [▶ 506]
- Image Segmentation [▶ 1327]
- Image Color and Contrast Processing [▶ 1235]
- Image Analysis [▶ 1175]
- Image Filtering [▶ 1275]

Container

- Basic Container Operations [▶ 547]
- Algebraic Container Operations [▶ 464]
- Container Statistics [▶ 861]

Contours

- Contour Analysis [▶ 955]

Measurement

- Measurement [▶ 1481]
- Geometric and coordinate transformations [▶ 1071]

Code quality

- Code Quality [▶ 818]

Code reading

- Code Reading [▶ 837]

Optical character recognition (OCR)

- Optical Character Recognition [▶ 1547]

Drawing and text

- Drawing [▶ 994]

Machine Learning

- MachineLearning [▶ 1384]
- Neural Network [▶ 1557]

Advanced Functions

- Fourier Analysis [▶ 1056]
- Keypoint Features [▶ 1338]

Miscellaneous

- Control Functions [▶ 989]
- Miscellaneous [▶ 1528]

Expert functions

Expert versions exist for many functions. These contain additional parameters and are usually marked by the name suffix "`Exp`". In the case of functions with Expert versions, this is additionally noted under the heading "Expert Parameters".

For example, the Expert version of F_VN_MaxImage [▶ 530] is called F_VN_MaxImagesExp [▶ 531] and contains the additional parameter `ipMask`.

Runtime behavior

The execution duration of some image processing algorithms depends on their parameterization and also the image content. It is therefore not deterministic. In order to guarantee real-time behavior despite that, watchdogs [▶ 146] can be applied to these functions or function groups that abort the functions after a specified time.

⚠ WARNING

TwinCAT crash when functions are used incorrectly

If some functions are used incorrectly, this can lead to errors (and to resulting TwinCAT crashes). These are not intercepted within the function, as corresponding checks would considerably lengthen the execution time. Therefore, you must take care to use the functions properly! For example, we recommend carrying out a validation of the input parameters.

6.1.4.1 License Overview

The following list shows the assignment of TwinCAT Vision functions to available licenses (see Licensing model [▶ 12]). It shows which license is required for the application of certain functions.

TF7100

The basic license TC3 Vision Base (TF7100) is always required.

TC3 Vision Base

F_VN_AdaptiveThreshold [▶ 1327]

F_VN_AdaptiveThresholdExp [▶ 1329]

F_VN_AddContainers [▶ 492]

F_VN_AddImages [▶ 507]

F_VN_AddImagesWeighted [▶ 508]

F_VN_AddImagesWeightedExp [▶ 509]

F_VN_AddScalarToImage [▶ 510]

F_VN_AddToContainerElements_DINT [▶ 465]

F_VN_AddToContainerElements_INT [▶ 466]

F_VN_AddToContainerElements_LREAL [▶ 466]

F_VN_AddToContainerElements_REAL [▶ 467]

F_VN_AddToContainerElements_SINT [▶ 468]

F_VN_AddToContainerElements_TcVnKeyPoint [▶ 468]

F_VN_AddToContainerElements_TcVnPoint2_DINT [▶ 469]

F_VN_AddToContainerElements_TcVnPoint2_LREAL [▶ 470]

F_VN_AddToContainerElements_TcVnPoint2_REAL [▶ 471]

F_VN_AddToContainerElements_TcVnPoint3_LREAL [▶ 472]

F_VN_AddToContainerElements_TcVnPoint3_REAL [▶ 473]

F_VN_AddToContainerElements_TcVnRectangle_DINT [▶ 473]

F_VN_AddToContainerElements_TcVnVector2_DINT [▶ 474]

F_VN_AddToContainerElements_TcVnVector2_INT [▶ 475]

F_VN_AddToContainerElements_TcVnVector2_REAL [▶ 476]

F_VN_AddToContainerElements_TcVnVector2_SINT [▶ 477]

F_VN_AddToContainerElements_TcVnVector2_UINT [▶ 478]

F_VN_AddToContainerElements_TcVnVector2_USINT [▶ 478]

F_VN_AddToContainerElements_TcVnVector3_INT [▶ 479]

F_VN_AddToContainerElements_TcVnVector3_REAL [▶ 480]

F_VN_AddToContainerElements_TcVnVector3_SINT [▶ 481]

F_VN_AddToContainerElements_TcVnVector3_UINT [▶ 482]

F_VN_AddToContainerElements_TcVnVector3_USINT [▶ 483]

F_VN_AddToContainerElements_TcVnVector4_DINT [▶ 483]

F_VN_AddToContainerElements_TcVnVector4_INT [▶ 484]

F_VN_AddToContainerElements_TcVnVector4_LREAL [▶ 485]

F_VN_AddToContainerElements_TcVnVector4_REAL [▶ 491]

F_VN_AddToContainerElements_TcVnVector4_SINT [▶ 486]

F_VN_AddToContainerElements_TcVnVector4_UINT [▶ 487]

F_VN_AddToContainerElements_TcVnVector4_USINT [▶ 488]

F_VN_AddToContainerElements_UDINT [▶ 488]

F_VN_AddToContainerElements_UINT [▶ 489]

F_VN_AddToContainerElements_ULINT [▶ 490]

F_VN_AddToContainerElements_USINT [▶ 491]

F_VN_AddVectorToImage [▶ 511]

F_VN_AdjustActiveContour [▶ 1177]

F_VN_AdvanceIterator [▶ 745]

F_VN_AlignRotatedImageRegion [▶ 1113]

F_VN_AlignRotatedImageRegionExp [▶ 1114]

F_VN_AppendToContainer_DINT [▶ 549]

F_VN_AppendToContainer_INT [▶ 549]

F_VN_AppendToContainer_ITcVnContainer [▶ 550]

F_VN_AppendToContainer_ITcVnForwardIterator [▶ 551]

F_VN_AppendToContainer_ITcVnImage [▶ 552]

F_VN_AppendToContainer_LREAL [▶ 552]

F_VN_AppendToContainer_REAL [▶ 553]

F_VN_AppendToContainer_SINT [▶ 554]

F_VN_AppendToContainer_TcVnDMatch [▶ 554]

F_VN_AppendToContainer_TcVnKeyPoint [▶ 555]

F_VN_AppendToContainer_TcVnPoint2_DINT [▶ 556]

F_VN_AppendToContainer_TcVnPoint2_LREAL [▶ 557]

F_VN_AppendToContainer_TcVnPoint2_REAL [▶ 558]

F_VN_AppendToContainer_TcVnPoint3_LREAL [▶ 559]

F_VN_AppendToContainer_TcVnPoint3_REAL [▶ 559]

F_VN_AppendToContainer_TcVnRectangle_DINT [▶ 560]

F_VN_AppendToContainer_TcVnVector2_DINT [▶ 561]

F_VN_AppendToContainer_TcVnVector2_INT [▶ 562]

F_VN_AppendToContainer_TcVnVector2_REAL [▶ 563]

F_VN_AppendToContainer_TcVnVector2_SINT [▶ 564]

F_VN_AppendToContainer_TcVnVector2_UINT [▶ 564]

F_VN_AppendToContainer_TcVnVector2_USINT [▶ 565]

F_VN_AppendToContainer_TcVnVector3_INT [▶ 566]

F_VN_AppendToContainer_TcVnVector3_REAL [▶ 567]

F_VN_AppendToContainer_TcVnVector3_SINT [▶ 568]

F_VN_AppendToContainer_TcVnVector3_UINT [▶ 569]

F_VN_AppendToContainer_TcVnVector3_USINT [▶ 569]

F_VN_AppendToContainer_TcVnVector4_DINT [▶ 570]

F_VN_AppendToContainer_TcVnVector4_INT [▶ 571]

F_VN_AppendToContainer_TcVnVector4_LREAL [▶ 572]

F_VN_AppendToContainer_TcVnVector4_REAL [▶ 578]

F_VN_AppendToContainer_TcVnVector4_SINT [▶ 573]

F_VN_AppendToContainer_TcVnVector4_UINT [▶ 574]

F_VN_AppendToContainer_TcVnVector4_USINT [▶ 574]

F_VN_AppendToContainer_UDINT [▶ 575]

F_VN_AppendToContainer_UINT [▶ 576]

F_VN_AppendToContainer_ULINT [▶ 577]

F_VN_AppendToContainer_USINT [▶ 577]

F_VN_ApplyColorMap [▶ 1236]

F_VN_ApplyLut [▶ 1238]

F_VN_ApplyRotationToAffineTransformation [▶ 1115]

F_VN_ApplyRotationToAffineTransformationExp [▶ 1116]

F_VN_ApplyScalingToAffineTransformation [▶ 1117]

F_VN_ApplyScalingToAffineTransformationExp [▶ 1118]

F_VN_ApplyTranslationToAffineTransformation [▶ 1119]

F_VN_ApplyTranslationToAffineTransformationExp [▶ 1120]

F_VN_ApplyYAxisInversionToAffineTransformation [▶ 1121]

F_VN_ApplyYAxisInversionToAffineTransformationExp [▶ 1122]

F_VN_ApproximatePolygon [▶ 956]

F_VN_BilateralFilter [▶ 1276]

F_VN_BilateralFilterExp [▶ 1280]

F_VN_BitwiseAndContainers [▶ 493]

F_VN_BitwiseAndImages [▶ 512]

F_VN_BitwiseAndImagesExp [▶ 512]

F_VN_BitwiseAndScalarWithImage [▶ 513]

F_VN_BitwiseAndVectorWithImage [▶ 514]

F_VN_BitwiseNotContainer [▶ 494]

F_VN_BitwiseNotImage [▶ 515]

F_VN_BitwiseNotImageExp [▶ 516]

F_VN_BitwiseOrContainers [▶ 494]

F_VN_BitwiseOrImages [▶ 517]

F_VN_BitwiseOrImagesExp [▶ 517]

F_VN_BitwiseOrScalarWithImage [▶ 518]

F_VN_BitwiseOrVectorWithImage [▶ 519]

F_VN_BitwiseXorContainers [▶ 495]

F_VN_BitwiseXorImages [▶ 520]

F_VN_BitwiseXorImagesExp [▶ 521]

F_VN_BitwiseXorScalarWithImage [▶ 522]

F_VN_BitwiseXorVectorWithImage [▶ 522]

F_VN_BlendImages [▶ 523]

F_VN_BoxFilter [▶ 1281]

F_VN_BoxFilterExp [▶ 1282]

F_VN_BrightBorderObjects [▶ 1283]

F_VN_CalibrateCameraPlanar [▶ 1086]

F_VN_CalibrateCameraPlanarExp [▶ 1087]

F_VN_CannyEdgeDetection [▶ 1204]

F_VN_CannyEdgeDetectionExp [▶ 1206]

F_VN_CheckColorRange [▶ 1331]

F_VN_CheckFunctionInitialization [▶ 1529]

F_VN_CheckIfEmpty [▶ 746]

F_VN_CheckIfIteratorIsAtEnd [▶ 747]

F_VN_CheckIfPointIsInsideContour [▶ 957]

F_VN_Clahe [▶ 1239]

F_VN_ClaheExp [▶ 1240]

F_VN_ClipLineToBoundary [▶ 1529]

F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531]

F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1532]

F_VN_CombineImageChannels [▶ 775]

F_VN_ConnectedComponents [▶ 1207]

F_VN_ConnectedComponentsExp [▶ 1208]

F_VN_ConnectedComponentsWithStats [▶ 1209]

F_VN_ConnectedComponentsWithStatsExp [▶ 1210]

F_VN_ContainerAverage [▶ 943]

F_VN_ContainerAverageElementwise2 [▶ 944]

F_VN_ContainerAverageElementwise3 [▶ 945]

F_VN_ContainerAverageElementwise4 [▶ 946]

F_VN_ContainerAverageVariance [▶ 947]

F_VN_ContainerAverageVarianceElementwise2 [▶ 948]

F_VN_ContainerAverageVarianceElementwise3 [▶ 950]

F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

F_VN_ContourArea [▶ 958]

F_VN_ContourCenterOfMass [▶ 959]

F_VN_ContourCircularity [▶ 960]

F_VN_ContourConvexity [▶ 961]

F_VN_ContourEccentricity [▶ 962]

F_VN_ContourElongation [▶ 963]

F_VN_ContourExtremePoint [▶ 964]

F_VN_ContourInertiaRatio [▶ 965]

F_VN_ContourMoments [▶ 966]

F_VN_ContourOrientation [▶ 967]

F_VN_ContourOrientationExp [▶ 968]

F_VN_ContourPerimeter [▶ 969]

F_VN_ContourRoundness [▶ 970]

F_VN_ConvertCartesianToPolarAngleImage [▶ 1123]

F_VN_ConvertCartesianToPolarAngleImageExp [▶ 1124]

F_VN_ConvertCartesianToPolarAngles [▶ 1124]

F_VN_ConvertCartesianToPolarAnglesExp [▶ 1125]

F_VN_ConvertCartesianToPolarImages [▶ 1126]

F_VN_ConvertCartesianToPolarImagesExp [▶ 1127]

F_VN_ConvertCartesianToPolarMagnitudeImage [▶ 1128]

F_VN_ConvertCartesianToPolarMagnitudes [▶ 1129]

F_VN_ConvertCartesianToPolarPoints [▶ 1130]

F_VN_ConvertCartesianToPolarPointsExp [▶ 1131]

F_VN_ConvertColorSpace [▶ 1241]

F_VN_ConvertContainerType [▶ 748]

F_VN_ConvertDataLayout [▶ 1558]

F_VN_ConvertElementType [▶ 775]

F_VN_ConvertElementTypeExp [▶ 776]

F_VN_ConvertITcUnknownToITcVnBitmapExport [▶ 1533]

F_VN_ConvertITcUnknownToITcVnContainer [▶ 1534]

F_VN_ConvertITcUnknownToITcVnImage [▶ 1535]

F_VN_ConvertITcUnknownToITcVnMlModel [▶ 1536]

F_VN_ConvertPolarToCartesianImages [▶ 1132]

F_VN_ConvertPolarToCartesianImagesExp [▶ 1133]

F_VN_ConvertPolarToCartesianPoints [▶ 1134]

F_VN_ConvertPolarToCartesianPointsExp [▶ 1135]

F_VN_ConvexHullPoints [▶ 971]

F_VN_ConvexHullPointsExp [▶ 972]

F_VN_ConvexityDefects [▶ 973]

F_VN_CopyContainer [▶ 749]

F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750]

F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751]

F_VN_CopyImage [▶ 777]

F_VN_CopyImageRegion [▶ 778]

F_VN_CopyImageRegionToRegion [▶ 780]

F_VN_CopyIntoDisplayableImage [▶ 781]

F_VN_CountNonZeroPixels [▶ 1212]

F_VN_CreateAssociatedImage [▶ 782]

F_VN_CreateBandpassButterworthFilter [▶ 1057]

F_VN_CreateBandpassGaussianFilter [▶ 1058]

F_VN_CreateBandrejectButterworthFilter [▶ 1059]

F_VN_CreateBandrejectGaussianFilter [▶ 1060]

F_VN_CreateContainer [▶ 753]

F_VN_CreateContainerFromArray [▶ 754]

F_VN_CreateEmptyImage [▶ 783]

F_VN_CreateHighpassButterworthFilter [▶ 1062]

F_VN_CreateHighpassGaussianFilter [▶ 1063]

F_VN_CreateImage [▶ 784]

F_VN_CreateImageAndSetPixels [▶ 785]

F_VN_CreateImageFromArray [▶ 786]

F_VN_CreateLowpassButterworthFilter [▶ 1064]

F_VN_CreateLowpassGaussianFilter [▶ 1065]

F_VN_CreateNDimensionalImage [▶ 814]

F_VN_CreateNDimensionalImageAndSetPixels [▶ 815]

F_VN_CreateNDimensionalImageFromArray [▶ 816]

F_VN_CreateStructuringElement [▶ 1285]

F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 974]

F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 975]

F_VN_CustomFilter [▶ 1287]

F_VN_CustomFilterExp [▶ 1290]

F_VN_DarkBorderObjects [▶ 1292]

F_VN_DecomposeAffineTransformation [▶ 1136]

F_VN_DecomposeAffineTransformationExp [▶ 1137]

F_VN_DecomposeHomography [▶ 1138]

F_VN_DecomposeHomographyExp [▶ 1140]

F_VN_DeinitializeFunction [▶ 1536]

F_VN_DetectBlobs [▶ 1178]

F_VN_DetectBlobsExp [▶ 1181]

F_VN_Dft [▶ 1066]

F_VN_DistanceTransformation [▶ 1213]

F_VN_DistanceTransformationExp [▶ 1215]

F_VN_DivideContainers [▶ 496]

F_VN_DivideContainersExp [▶ 497]

F_VN_DivideImageByScalar [▶ 524]

F_VN_DivideImageByVector [▶ 525]

F_VN_DivideImages [▶ 526]

F_VN_DivideScalarByImage [▶ 527]

F_VN_DivideVectorByImage [▶ 528]

F_VN_DoubleThreshold [▶ 1332]

F_VN_DrawArrow [▶ 995]

F_VN_DrawArrow_TcVnVector4_DINT [▶ 996]

F_VN_DrawArrowExp [▶ 997]

F_VN_DrawArrowExp_TcVnVector4_DINT [▶ 999]

F_VN_DrawCircle [▶ 1000]

F_VN_DrawCircleExp [▶ 1001]

F_VN_DrawCircles [▶ 1002]

F_VN_DrawCirclesExp [▶ 1003]

F_VN_DrawCircularArc [▶ 1004]

F_VN_DrawCircularArcExp [▶ 1005]

F_VN_DrawComponents [▶ 1006]

F_VN_DrawComponentsExp [▶ 1007]

F_VN_DrawContours [▶ 1009]

F_VN_DrawContoursExp [▶ 1010]

F_VN_DrawEllipse [▶ 1012]

F_VN_DrawEllipseExp [▶ 1013]

F_VN_DrawLine [▶ 1016]

F_VN_DrawLine_TcVnVector4_DINT [▶ 1017]

F_VN_DrawLine_TcVnVector4_LREAL [▶ 1018]

F_VN_DrawLineExp [▶ 1020]

F_VN_DrawLineExp_TcVnVector4_DINT [▶ 1021]

F_VN_DrawLineExp_TcVnVector4_LREAL [▶ 1022]

F_VN_DrawLines [▶ 1023]

F_VN_DrawLinesExp [▶ 1025]

F_VN_DrawOrientation [▶ 1028]

F_VN_DrawOrientationExp [▶ 1030]

F_VN_DrawPoint [▶ 1031]

F_VN_DrawPointExp [▶ 1032]

F_VN_DrawPoints [▶ 1033]

F_VN_DrawPointsExp [▶ 1034]

F_VN_DrawPolygon [▶ 1035]

F_VN_DrawPolygonExp [▶ 1036]

F_VN_DrawRectangle [▶ 1037]

F_VN_DrawRectangle_TcVnRectangle_DINT [▶ 1038]

F_VN_DrawRectangle_TcVnRectangle_UDINT [▶ 1039]

F_VN_DrawRotatedRectangle [▶ 1040]

F_VN_DrawRotatedRectangleExp [▶ 1041]

F_VN_ElementwiseExp [▶ 529]

F_VN_ElementwiseLog [▶ 529]

F_VN_EnclosingCircle [▶ 976]

F_VN_EnclosingRectangle [▶ 977]

F_VN_EnclosingTriangle [▶ 978]

F_VN_EraseFromContainer [▶ 755]

F_VN_ExportContainer [▶ 755]

F_VN_ExportContainer_String [▶ 757]

F_VN_ExportContainerSize [▶ 758]

F_VN_ExportImage [▶ 787]

F_VN_ExportImageAsBmp [▶ 788]

F_VN_ExportImageAsBmpExp [▶ 789]

F_VN_ExportImageAsBmpSize [▶ 790]

F_VN_ExportImageAsBmpSizeExp [▶ 790]

F_VN_ExportImageSize [▶ 791]

F_VN_ExportSubContainer [▶ 759]

F_VN_ExportSubContainer_String [▶ 760]

F_VN_ExportSubContainerSize [▶ 762]

F_VN_ExtractContainerRange [▶ 763]

F_VN_FillCircle [▶ 1043]

F_VN_FillContainer_DINT [▶ 579]

F_VN_FillContainer_INT [▶ 580]

F_VN_FillContainer_ITcVnImage [▶ 580]

F_VN_FillContainer_LREAL [▶ 581]

F_VN_FillContainer_REAL [▶ 582]

F_VN_FillContainer_SINT [▶ 583]

F_VN_FillContainer_TcVnDMatch [▶ 583]

F_VN_FillContainer_TcVnKeyPoint [▶ 584]

F_VN_FillContainer_TcVnPoint2_DINT [▶ 585]

F_VN_FillContainer_TcVnPoint2_LREAL [▶ 586]

F_VN_FillContainer_TcVnPoint2_REAL [▶ 587]

F_VN_FillContainer_TcVnPoint3_LREAL [▶ 588]

F_VN_FillContainer_TcVnPoint3_REAL [▶ 588]

F_VN_FillContainer_TcVnRectangle_DINT [▶ 589]

F_VN_FillContainer_TcVnVector2_DINT [▶ 590]

F_VN_FillContainer_TcVnVector2_INT [▶ 591]

F_VN_FillContainer_TcVnVector2_REAL [▶ 592]

F_VN_FillContainer_TcVnVector2_SINT [▶ 593]

F_VN_FillContainer_TcVnVector2_UINT [▶ 593]

F_VN_FillContainer_TcVnVector2_USINT [▶ 594]

F_VN_FillContainer_TcVnVector3_INT [▶ 595]

F_VN_FillContainer_TcVnVector3_REAL [▶ 596]

F_VN_FillContainer_TcVnVector3_SINT [▶ 597]

F_VN_FillContainer_TcVnVector3_UINT [▶ 598]

F_VN_FillContainer_TcVnVector3_USINT [▶ 599]

F_VN_FillContainer_TcVnVector4_DINT [▶ 599]

F_VN_FillContainer_TcVnVector4_INT [▶ 600]

F_VN_FillContainer_TcVnVector4_LREAL [▶ 601]

F_VN_FillContainer_TcVnVector4_REAL [▶ 607]

F_VN_FillContainer_TcVnVector4_SINT [▶ 602]

F_VN_FillContainer_TcVnVector4_UINT [▶ 603]

F_VN_FillContainer_TcVnVector4_USINT [▶ 604]

F_VN_FillContainer_UDINT [▶ 605]

F_VN_FillContainer_UINT [▶ 605]

F_VN_FillContainer_ULINT [▶ 606]

F_VN_FillContainer_USINT [▶ 607]

F_VN_FillContainerExp_DINT [▶ 608]

F_VN_FillContainerExp_INT [▶ 609]

F_VN_FillContainerExp_ITcVnImage [▶ 610]

F_VN_FillContainerExp_LREAL [▶ 611]

F_VN_FillContainerExp_REAL [▶ 612]

F_VN_FillContainerExp_SINT [▶ 613]

F_VN_FillContainerExp_TcVnDMatch [▶ 613]

F_VN_FillContainerExp_TcVnKeyPoint [▶ 614]

F_VN_FillContainerExp_TcVnPoint2_DINT [▶ 615]

F_VN_FillContainerExp_TcVnPoint2_LREAL [▶ 616]

F_VN_FillContainerExp_TcVnPoint2_REAL [▶ 617]

F_VN_FillContainerExp_TcVnPoint3_LREAL [▶ 618]

F_VN_FillContainerExp_TcVnPoint3_REAL [▶ 619]

F_VN_FillContainerExp_TcVnRectangle_DINT [▶ 620]

F_VN_FillContainerExp_TcVnVector2_DINT [▶ 621]

F_VN_FillContainerExp_TcVnVector2_INT [▶ 622]

F_VN_FillContainerExp_TcVnVector2_REAL [▶ 623]

F_VN_FillContainerExp_TcVnVector2_SINT [▶ 624]

F_VN_FillContainerExp_TcVnVector2_UINT [▶ 625]

F_VN_FillContainerExp_TcVnVector2_USINT [▶ 626]

F_VN_FillContainerExp_TcVnVector3_INT [▶ 627]

F_VN_FillContainerExp_TcVnVector3_REAL [▶ 628]

F_VN_FillContainerExp_TcVnVector3_SINT [▶ 629]

F_VN_FillContainerExp_TcVnVector3_UINT [▶ 630]

F_VN_FillContainerExp_TcVnVector3_USINT [▶ 631]

F_VN_FillContainerExp_TcVnVector4_DINT [▶ 632]

F_VN_FillContainerExp_TcVnVector4_INT [▶ 633]

F_VN_FillContainerExp_TcVnVector4_LREAL [▶ 634]

F_VN_FillContainerExp_TcVnVector4_REAL [▶ 642]

F_VN_FillContainerExp_TcVnVector4_SINT [▶ 635]

F_VN_FillContainerExp_TcVnVector4_UINT [▶ 636]

F_VN_FillContainerExp_TcVnVector4_USINT [▶ 637]

F_VN_FillContainerExp_UDINT [▶ 638]

F_VN_FillContainerExp_UINT [▶ 639]

F_VN_FillContainerExp_ULINT [▶ 640]

F_VN_FillContainerExp_USINT [▶ 641]

F_VN_FillContours [▶ 1044]

F_VN_FillEllipse [▶ 1045]

F_VN_FillHoles [▶ 1293]

F_VN_FillPolygon [▶ 1046]

F_VN_FillRectangle [▶ 1047]

F_VN_FillRotatedRectangle [▶ 1048]

F_VN_FindContourHierarchyExp [▶ 1183]

F_VN_FindContours [▶ 1187]

F_VN_FindContoursExp [▶ 1190]

F_VN_FitEllipse [▶ 979]

F_VN_FitLine [▶ 980]

F_VN_FitLineExp [▶ 981]

F_VN_FlipImage [▶ 1141]

F_VN_FourierDescriptors [▶ 982]

F_VN_FuseImages [▶ 792]

F_VN_FuseImagesArray [▶ 794]

F_VN_GaussianFilter [▶ 1295]

F_VN_GaussianFilterExp [▶ 1298]

F_VN_GenerateAffineTransformationUnitMatrix2D [▶ 1142]

F_VN_GenerateColorMap [▶ 1241]

F_VN_GenerateCustomColorMap [▶ 1243]

F_VN_GetAffineTransformation [▶ 1143]

F_VN_GetAffineTransformation2D [▶ 1144]

F_VN_GetAffineTransformation2DExp [▶ 1145]

F_VN_GetAt_DINT [▶ 643]

F_VN_GetAt_INT [▶ 644]

F_VN_GetAt_ITcVnContainer [▶ 645]

F_VN_GetAt_ITcVnImage [▶ 645]

F_VN_GetAt_LREAL [▶ 646]

F_VN_GetAt_REAL [▶ 647]

F_VN_GetAt_SINT [▶ 648]

F_VN_GetAt_TcVnDMatch [▶ 649]

F_VN_GetAt_TcVnKeyPoint [▶ 650]

F_VN_GetAt_TcVnPoint2_DINT [▶ 651]

F_VN_GetAt_TcVnPoint2_LREAL [▶ 652]

F_VN_GetAt_TcVnPoint2_REAL [▶ 653]

F_VN_GetAt_TcVnPoint3_LREAL [▶ 654]

F_VN_GetAt_TcVnPoint3_REAL [▶ 655]

F_VN_GetAt_TcVnRectangle_DINT [▶ 656]

F_VN_GetAt_TcVnVector2_DINT [▶ 657]

F_VN_GetAt_TcVnVector2_INT [▶ 658]

F_VN_GetAt_TcVnVector2_REAL [▶ 659]

F_VN_GetAt_TcVnVector2_SINT [▶ 660]

F_VN_GetAt_TcVnVector2_UINT [▶ 661]

F_VN_GetAt_TcVnVector2_USINT [▶ 662]

F_VN_GetAt_TcVnVector3_INT [▶ 663]

F_VN_GetAt_TcVnVector3_REAL [▶ 664]

F_VN_GetAt_TcVnVector3_SINT [▶ 665]

F_VN_GetAt_TcVnVector3_UINT [▶ 666]

F_VN_GetAt_TcVnVector3_USINT [▶ 667]

F_VN_GetAt_TcVnVector4_DINT [▶ 668]

F_VN_GetAt_TcVnVector4_INT [▶ 669]

F_VN_GetAt_TcVnVector4_LREAL [▶ 670]

F_VN_GetAt_TcVnVector4_REAL [▶ 678]

F_VN_GetAt_TcVnVector4_SINT [▶ 671]

F_VN_GetAt_TcVnVector4_UINT [▶ 672]

F_VN_GetAt_TcVnVector4_USINT [▶ 673]

F_VN_GetAt_UDINT [▶ 674]

F_VN_GetAt_UINT [▶ 675]

F_VN_GetAt_ULINT [▶ 676]

F_VN_GetAt_USINT [▶ 677]

F_VN_GetConnectedComponent [▶ 1216]

F_VN_GetConnectedComponentExp [▶ 1217]

F_VN_GetContainer [▶ 764]

F_VN_GetContainerExp [▶ 765]

F_VN_GetForwardIterator [▶ 766]

F_VN_GetImageChannel [▶ 795]

F_VN_GetImageHeight [▶ 796]

F_VN_GetImageInfo [▶ 797]

F_VN_GetImageWidth [▶ 798]

F_VN_GetLayoutInfo [▶ 817]

F_VN_GetNumberOfElements [▶ 766]

F_VN_GetPerspectiveTransformation [▶ 1146]

F_VN_GetPixel [▶ 799]

F_VN_GetPixelFormat [▶ 800]

F_VN_GetRandomAccessIterator [▶ 767]

F_VN_GetRoi [▶ 801]

F_VN_GetTimestamp [▶ 1537]

F_VN_Histogram [▶ 1244]

F_VN_HistogramEqualization [▶ 1245]

F_VN_HistogramEqualizationExp [▶ 1246]

F_VN_HistogramExp [▶ 1247]

F_VN_Homography [▶ 1147]

F_VN_HomographyExp [▶ 1148]

F_VN_HoughCircles [▶ 1192]

F_VN_HoughCirclesExp [▶ 1193]

F_VN_HoughLines [▶ 1194]

F_VN_HoughLinesExp [▶ 1195]

F_VN_HoughLinesP [▶ 1196]

F_VN_HoughLinesPExp [▶ 1197]

F_VN_HuMomentInvariants [▶ 1538]

F_VN_ImageAbsoluteAverage [▶ 1233]

F_VN_ImageAverage [▶ 1218]

F_VN_ImageAverageExp [▶ 1218]

F_VN_ImageAverageStdDev [▶ 1219]

F_VN_ImageAverageStdDevExp [▶ 1220]

F_VN_ImageCenterOfMass [▶ 1221]

F_VN_ImageCenterOfMassExp [▶ 1223]

F_VN_ImageMedian [▶ 1224]

F_VN_ImageMedianExp [▶ 1225]

F_VN_ImageMoments [▶ 1226]

F_VN_IncrementIterator [▶ 768]

F_VN_InitMatrixStruct [▶ 1539]

F_VN_InsertIntoContainer_DINT [▶ 679]

F_VN_InsertIntoContainer_INT [▶ 680]

F_VN_InsertIntoContainer_ITcVnContainer [▶ 681]

F_VN_InsertIntoContainer_ITcVnForwardIterator [▶ 682]

F_VN_InsertIntoContainer_ITcVnImage [▶ 682]

F_VN_InsertIntoContainer_LREAL [▶ 683]

F_VN_InsertIntoContainer_REAL [▶ 684]

F_VN_InsertIntoContainer_SINT [▶ 685]

F_VN_InsertIntoContainer_TcVnDMatch [▶ 686]

F_VN_InsertIntoContainer_TcVnKeyPoint [▶ 686]

F_VN_InsertIntoContainer_TcVnPoint2_DINT [▶ 687]

F_VN_InsertIntoContainer_TcVnPoint2_LREAL [▶ 688]

F_VN_InsertIntoContainer_TcVnPoint2_REAL [▶ 689]

F_VN_InsertIntoContainer_TcVnPoint3_LREAL [▶ 690]

F_VN_InsertIntoContainer_TcVnPoint3_REAL [▶ 691]

F_VN_InsertIntoContainer_TcVnRectangle_DINT [▶ 692]

F_VN_InsertIntoContainer_TcVnVector2_DINT [▶ 692]

F_VN_InsertIntoContainer_TcVnVector2_INT [▶ 693]

F_VN_InsertIntoContainer_TcVnVector2_REAL [▶ 694]

F_VN_InsertIntoContainer_TcVnVector2_SINT [▶ 695]

F_VN_InsertIntoContainer_TcVnVector2_UINT [▶ 696]

F_VN_InsertIntoContainer_TcVnVector2_USINT [▶ 697]

F_VN_InsertIntoContainer_TcVnVector3_INT [▶ 698]

F_VN_InsertIntoContainer_TcVnVector3_REAL [▶ 699]

F_VN_InsertIntoContainer_TcVnVector3_SINT [▶ 699]

F_VN_InsertIntoContainer_TcVnVector3_UINT [▶ 700]

F_VN_InsertIntoContainer_TcVnVector3_USINT [▶ 701]

F_VN_InsertIntoContainer_TcVnVector4_DINT [▶ 702]

F_VN_InsertIntoContainer_TcVnVector4_INT [▶ 703]

F_VN_InsertIntoContainer_TcVnVector4_LREAL [▶ 704]

F_VN_InsertIntoContainer_TcVnVector4_REAL [▶ 710]

F_VN_InsertIntoContainer_TcVnVector4_SINT [▶ 705]

F_VN_InsertIntoContainer_TcVnVector4_UINT [▶ 706]

F_VN_InsertIntoContainer_TcVnVector4_USINT [▶ 706]

F_VN_InsertIntoContainer_UDINT [▶ 707]

F_VN_InsertIntoContainer_UINT [▶ 708]

F_VN_InsertIntoContainer_ULINT [▶ 709]

F_VN_InsertIntoContainer_USINT [▶ 710]

F_VN_InverseDft [▶ 1067]

F_VN_InvertAffineTransformation [▶ 1149]

F_VN_InvertImageColor [▶ 1249]

F_VN_InvertImageColorExp [▶ 1250]

F_VN_InvertMatrix3x3 [▶ 1540]

F_VN_IteratorDistance [▶ 769]

F_VN_LaplacianFilter [▶ 1299]

F_VN_LaplacianFilterExp [▶ 1301]

F_VN_LineIntersectionPoint [▶ 1541]

F_VN_LineIntersectionPointAndAngle [▶ 1542]

F_VN_LocalMaxima [▶ 1303]

F_VN_LocalMinima [▶ 1305]

F_VN_MaxContainer [▶ 498]

F_VN_MaxElement [▶ 952]

F_VN_MaxElementElementwise_DINT [▶ 862]

F_VN_MaxElementElementwise_INT [▶ 862]

F_VN_MaxElementElementwise_LREAL [▶ 863]

F_VN_MaxElementElementwise_REAL [▶ 864]

F_VN_MaxElementElementwise_SINT [▶ 865]

F_VN_MaxElementElementwise_TcVnPoint2_DINT [▶ 866]

F_VN_MaxElementElementwise_TcVnPoint2_LREAL [▶ 867]

F_VN_MaxElementElementwise_TcVnPoint2_REAL [▶ 868]

F_VN_MaxElementElementwise_TcVnPoint3_LREAL [▶ 868]

F_VN_MaxElementElementwise_TcVnPoint3_REAL [▶ 869]

F_VN_MaxElementElementwise_TcVnVector2_DINT [▶ 870]

F_VN_MaxElementElementwise_TcVnVector2_INT [▶ 871]

F_VN_MaxElementElementwise_TcVnVector2_REAL [▶ 872]

F_VN_MaxElementElementwise_TcVnVector2_SINT [▶ 873]

F_VN_MaxElementElementwise_TcVnVector2_UINT [▶ 873]

F_VN_MaxElementElementwise_TcVnVector2_USINT [▶ 874]

F_VN_MaxElementElementwise_TcVnVector3_INT [▶ 875]

F_VN_MaxElementElementwise_TcVnVector3_REAL [▶ 876]

F_VN_MaxElementElementwise_TcVnVector3_SINT [▶ 877]

F_VN_MaxElementElementwise_TcVnVector3_UINT [▶ 878]

F_VN_MaxElementElementwise_TcVnVector3_USINT [▶ 878]

F_VN_MaxElementElementwise_TcVnVector4_DINT [▶ 879]

F_VN_MaxElementElementwise_TcVnVector4_INT [▶ 880]

F_VN_MaxElementElementwise_TcVnVector4_LREAL [▶ 881]

F_VN_MaxElementElementwise_TcVnVector4_REAL [▶ 888]

F_VN_MaxElementElementwise_TcVnVector4_SINT [▶ 882]

F_VN_MaxElementElementwise_TcVnVector4_UINT [▶ 883]

F_VN_MaxElementElementwise_TcVnVector4_USINT [▶ 883]

F_VN_MaxElementElementwise_UDINT [▶ 884]

F_VN_MaxElementElementwise_UINT [▶ 885]

F_VN_MaxElementElementwise_ULINT [▶ 886]

F_VN_MaxElementElementwise_USINT [▶ 887]

F_VN_MaxImage [▶ 530]

F_VN_MaxImageExp [▶ 531]

F_VN_MaxImageWithScalar [▶ 532]

F_VN_MaxImageWithScalarExp [▶ 533]

F_VN_MaxImageWithVector [▶ 534]

F_VN_MaxImageWithVectorExp [▶ 535]

F_VN_MaxPixelValue [▶ 1227]

F_VN_MaxPixelValueExp [▶ 1228]

F_VN_MedianElement [▶ 953]

F_VN_MedianElementElementwise_DINT [▶ 889]

F_VN_MedianElementElementwise_INT [▶ 890]

F_VN_MedianElementElementwise_LREAL [▶ 890]

F_VN_MedianElementElementwise_REAL [▶ 891]

F_VN_MedianElementElementwise_SINT [▶ 892]

F_VN_MedianElementElementwise_TcVnPoint2_DINT [▶ 893]

F_VN_MedianElementElementwise_TcVnPoint2_LREAL [▶ 894]

F_VN_MedianElementElementwise_TcVnPoint2_REAL [▶ 895]

F_VN_MedianElementElementwise_TcVnPoint3_LREAL [▶ 895]

F_VN_MedianElementElementwise_TcVnPoint3_REAL [▶ 896]

F_VN_MedianElementElementwise_TcVnVector2_DINT [▶ 897]

F_VN_MedianElementElementwise_TcVnVector2_INT [▶ 898]

F_VN_MedianElementElementwise_TcVnVector2_REAL [▶ 899]

F_VN_MedianElementElementwise_TcVnVector2_SINT [▶ 900]

F_VN_MedianElementElementwise_TcVnVector2_UINT [▶ 900]

F_VN_MedianElementElementwise_TcVnVector2_USINT [▶ 901]

F_VN_MedianElementElementwise_TcVnVector3_INT [▶ 902]

F_VN_MedianElementElementwise_TcVnVector3_REAL [▶ 903]

F_VN_MedianElementElementwise_TcVnVector3_SINT [▶ 904]

F_VN_MedianElementElementwise_TcVnVector3_UINT [▶ 905]

F_VN_MedianElementElementwise_TcVnVector3_USINT [▶ 905]

F_VN_MedianElementElementwise_TcVnVector4_DINT [▶ 906]

F_VN_MedianElementElementwise_TcVnVector4_INT [▶ 907]

F_VN_MedianElementElementwise_TcVnVector4_LREAL [▶ 908]

F_VN_MedianElementElementwise_TcVnVector4_REAL [▶ 915]

F_VN_MedianElementElementwise_TcVnVector4_SINT [▶ 909]

F_VN_MedianElementElementwise_TcVnVector4_UINT [▶ 910]

F_VN_MedianElementElementwise_TcVnVector4_USINT [▶ 910]

F_VN_MedianElementElementwise_UDINT [▶ 911]

F_VN_MedianElementElementwise_UINT [▶ 912]

F_VN_MedianElementElementwise_ULINT [▶ 913]

F_VN_MedianElementElementwise_USINT [▶ 914]

F_VN_MedianFilter [▶ 1306]

F_VN_MinContainer [▶ 499]

F_VN_MinElement [▶ 954]

F_VN_MinElementElementwise_DINT [▶ 916]

F_VN_MinElementElementwise_INT [▶ 916]

F_VN_MinElementElementwise_LREAL [▶ 917]

F_VN_MinElementElementwise_REAL [▶ 918]

F_VN_MinElementElementwise_SINT [▶ 919]

F_VN_MinElementElementwise_TcVnPoint2_DINT [▶ 920]

F_VN_MinElementElementwise_TcVnPoint2_LREAL [▶ 921]

F_VN_MinElementElementwise_TcVnPoint2_REAL [▶ 922]

F_VN_MinElementElementwise_TcVnPoint3_LREAL [▶ 922]

F_VN_MinElementElementwise_TcVnPoint3_REAL [▶ 923]

F_VN_MinElementElementwise_TcVnVector2_DINT [▶ 924]

F_VN_MinElementElementwise_TcVnVector2_INT [▶ 925]

F_VN_MinElementElementwise_TcVnVector2_REAL [▶ 926]

F_VN_MinElementElementwise_TcVnVector2_SINT [▶ 927]

F_VN_MinElementElementwise_TcVnVector2_UINT [▶ 927]

F_VN_MinElementElementwise_TcVnVector2_USINT [▶ 928]

F_VN_MinElementElementwise_TcVnVector3_INT [▶ 929]

F_VN_MinElementElementwise_TcVnVector3_REAL [▶ 930]

F_VN_MinElementElementwise_TcVnVector3_SINT [▶ 931]

F_VN_MinElementElementwise_TcVnVector3_UINT [▶ 932]

F_VN_MinElementElementwise_TcVnVector3_USINT [▶ 932]

F_VN_MinElementElementwise_TcVnVector4_DINT [▶ 933]

F_VN_MinElementElementwise_TcVnVector4_INT [▶ 934]

F_VN_MinElementElementwise_TcVnVector4_LREAL [▶ 935]

F_VN_MinElementElementwise_TcVnVector4_REAL [▶ 942]

F_VN_MinElementElementwise_TcVnVector4_SINT [▶ 936]

F_VN_MinElementElementwise_TcVnVector4_UINT [▶ 937]

F_VN_MinElementElementwise_TcVnVector4_USINT [▶ 937]

F_VN_MinElementElementwise_UDINT [▶ 938]

F_VN_MinElementElementwise_UINT [▶ 939]

F_VN_MinElementElementwise_ULINT [▶ 940]

F_VN_MinElementElementwise_USINT [▶ 941]

F_VN_MinImage [▶ 536]

F_VN_MinImageExp [▶ 536]

F_VN_MinImageWithScalar [▶ 537]

F_VN_MinImageWithScalarExp [▶ 538]

F_VN_MinImageWithVector [▶ 539]

F_VN_MinImageWithVectorExp [▶ 540]

F_VN_MinPixelValue [▶ 1229]

F_VN_MinPixelValueExp [▶ 1230]

F_VN_MixImageChannels [▶ 802]

F_VN_MorphologicalOperator [▶ 1309]

F_VN_MultiplyContainers [▶ 500]

F_VN_MultiplyImages [▶ 541]

F_VN_MultiplyImageWithScalar [▶ 542]

F_VN_MultiplyImageWithVector [▶ 542]

F_VN_MultiplyMatrices [▶ 1543]

F_VN_MultiplyWithContainerElements1 [▶ 501]

F_VN_MultiplyWithContainerElements2 [▶ 502]

F_VN_MultiplyWithContainerElements3 [▶ 502]

F_VN_MultiplyWithContainerElements4 [▶ 503]

F_VN_NegateContainer [▶ 504]

F_VN_NonMaxSuppression [▶ 1562]

F_VN_NonMaxSuppressionExp [▶ 1563]

F_VN_NormalizeImage [▶ 1251]

F_VN_NormalizeImageExp [▶ 1252]

F_VN_NormalizeImageForDisplay [▶ 1253]

F_VN_OptimalDftSize [▶ 1068]

F_VN_PadImageBorder [▶ 1069]

F_VN_PadImageBorderExp [▶ 1070]

F_VN_PerspectiveTransformation [▶ 1150]

F_VN_PlotIntensityProfile [▶ 1049]

F_VN_PlotIntensityProfileExp [▶ 1050]

F_VN_PutLabel [▶ 1051]

F_VN_PutLabelExp [▶ 1052]

F_VN_PutText [▶ 1053]

F_VN_PutTextExp [▶ 1055]

F_VN_PyramidDown [▶ 1151]

F_VN_PyramidUp [▶ 1152]

F_VN_Reduce [▶ 1565]

F_VN_ReduceArg [▶ 1566]

F_VN_ReferenceColorSimilarity_ITcVnColorModel [▶ 1254]

F_VN_ReferenceColorSimilarity_ITcVnMlModel [▶ 1255]

F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL [▶ 1256]

F_VN_ReferenceColorSimilarityExp_ITcVnColorModel [▶ 1257]

F_VN_ReferenceColorSimilarityExp_ITcVnMlModel [▶ 1258]

F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1260]

F_VN_RegionOrientation [▶ 1231]

F_VN_RegionOrientationExp [▶ 1232]

F_VN_RemapImageToLogPolarSpace [▶ 1153]

F_VN_RemapImageToLogPolarSpaceExp [▶ 1154]

F_VN_RemapImageToLogPolarSpaceExp2 [▶ 1155]

F_VN_RemapImageToPolarSpace [▶ 1156]

F_VN_RemapImageToPolarSpaceExp [▶ 1157]

F_VN_RemapImageToPolarSpaceExp2 [▶ 1158]

F_VN_RemoveLocalMaxima [▶ 1310]

F_VN_RemoveLocalMinima [▶ 1312]

F_VN_ReserveContainerMemory [▶ 770]

F_VN_ResetRoi [▶ 803]

F_VN_ReshapeImage [▶ 1566]

F_VN_ResizeImage [▶ 1159]

F_VN_ResizeImageExp [▶ 1160]

F_VN_ReverseContainer [▶ 771]

F_VN_RotatedRectangleCorners [▶ 1544]

F_VN_RotatedRectangleIntersection [▶ 1545]

F_VN_RotateImage [▶ 1162]

F_VN_RotateImageExp [▶ 1162]

F_VN_RotateImageExp2 [▶ 1163]

F_VN_SauvolaThreshold [▶ 1337]

F_VN_ScharrFilter [▶ 1313]

F_VN_ScharrFilterExp [▶ 1315]

F_VN_SeparableCustomFilter [▶ 1317]

F_VN_SeparableCustomFilterExp [▶ 1319]

F_VN_SetAt_DINT [▶ 711]

F_VN_SetAt_INT [▶ 712]

F_VN_SetAt_ITcVnContainer [▶ 713]

F_VN_SetAt_ITcVnImage [▶ 714]

F_VN_SetAt_LREAL [▶ 714]

F_VN_SetAt_REAL [▶ 715]

F_VN_SetAt_SINT [▶ 716]

F_VN_SetAt_TcVnDMatch [▶ 717]

F_VN_SetAt_TcVnKeyPoint [▶ 718]

F_VN_SetAt_TcVnPoint2_DINT [▶ 719]

F_VN_SetAt_TcVnPoint2_LREAL [▶ 720]

F_VN_SetAt_TcVnPoint2_REAL [▶ 721]

F_VN_SetAt_TcVnPoint3_LREAL [▶ 722]

F_VN_SetAt_TcVnPoint3_REAL [▶ 723]

F_VN_SetAt_TcVnRectangle_DINT [▶ 723]

F_VN_SetAt_TcVnVector2_DINT [▶ 724]

F_VN_SetAt_TcVnVector2_INT [▶ 725]

F_VN_SetAt_TcVnVector2_REAL [▶ 726]

F_VN_SetAt_TcVnVector2_SINT [▶ 727]

F_VN_SetAt_TcVnVector2_UINT [▶ 728]

F_VN_SetAt_TcVnVector2_USINT [▶ 729]

F_VN_SetAt_TcVnVector3_INT [▶ 730]

F_VN_SetAt_TcVnVector3_REAL [▶ 731]

F_VN_SetAt_TcVnVector3_SINT [▶ 732]

F_VN_SetAt_TcVnVector3_UINT [▶ 733]

F_VN_SetAt_TcVnVector3_USINT [▶ 734]

F_VN_SetAt_TcVnVector4_DINT [▶ 735]

F_VN_SetAt_TcVnVector4_INT [▶ 736]

F_VN_SetAt_TcVnVector4_LREAL [▶ 737]

F_VN_SetAt_TcVnVector4_REAL [▶ 744]

F_VN_SetAt_TcVnVector4_SINT [▶ 738]

F_VN_SetAt_TcVnVector4_UINT [▶ 739]

F_VN_SetAt_TcVnVector4_USINT [▶ 740]

F_VN_SetAt_UDINT [▶ 741]

F_VN_SetAt_UINT [▶ 742]

F_VN_SetAt_ULINT [▶ 743]

F_VN_SetAt_USINT [▶ 744]

F_VN_SetContainer [▶ 772]

F_VN_SetImageChannel [▶ 804]

F_VN_SetIteratorToBegin [▶ 772]

F_VN_SetPixel [▶ 805]

F_VN_SetPixels [▶ 806]

F_VN_SetPixelsExp [▶ 806]

F_VN_SetRngSeed [▶ 1546]

F_VN_SetRoi [▶ 807]

F_VN_SetRoi_TcVnRectangle_DINT [▶ 809]

F_VN_SetRoi_TcVnRectangle_UDINT [▶ 809]

F_VN_Sigmoid [▶ 1567]

F_VN_SigmoidExp [▶ 1568]

F_VN_SobelFilter [▶ 1321]

F_VN_SobelFilterExp [▶ 1323]

F_VN_SoftMax [▶ 1569]

F_VN_SplitImageChannels [▶ 811]

F_VN_StartAbsWatchdog [▶ 990]

F_VN_StartAbsWatchdogExp [▶ 990]

F_VN_StartRelWatchdog [▶ 991]

F_VN_StartRelWatchdogExp [▶ 992]

F_VN_StopWatchdog [▶ 993]

F_VN_SubtractContainers [▶ 505]

F_VN_SubtractImageFromScalar [▶ 543]

F_VN_SubtractImageFromVector [▶ 544]

F_VN_SubtractImages [▶ 545]

F_VN_SubtractScalarFromImage [▶ 546]

F_VN_SubtractVectorFromImage [▶ 547]

F_VN_Threshold [▶ 1334]

F_VN_TrainImageColor [▶ 1261]

F_VN_TrainImageColor_ITcVnMlModel [▶ 1263]

F_VN_TrainImageColorExp [▶ 1264]

F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1267]

F_VN_TrainImageColorExp2 [▶ 1269]

F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1272]

F_VN_TransformCoordinatesPlanar_Container [▶ 1109]

F_VN_TransformCoordinatesPlanar_Point [▶ 1110]

F_VN_TransformIntoDisplayableImage [▶ 812]

F_VN_TransformIntoDisplayableImageExp [▶ 813]

F_VN_UpdateTimestamp [▶ 1547]

F_VN_UprightBoundingRectangle [▶ 988]

F_VN_UprightInnerRectangle [▶ 1233]

F_VN_UprightInnerRectangleExp [▶ 1234]

F_VN_VarianceFilter [▶ 1325]

F_VN_VarianceFilterExp [▶ 1326]

F_VN_WarpAffine [▶ 1165]

F_VN_WarpAffine_Container [▶ 1166]

F_VN_WarpAffine_Point [▶ 1167]

F_VN_WarpAffine_Rectangle [▶ 1168]

F_VN_WarpAffineExp [▶ 1169]

F_VN_WarpPerspective [▶ 1170]

F_VN_WarpPerspective_Container [▶ 1171]

F_VN_WarpPerspective_Point [▶ 1172]

F_VN_WarpPerspective_Rectangle [▶ 1173]

F_VN_WarpPerspectiveExp [▶ 1174]

F_VN_WatershedSegmentationExp [▶ 1336]

F_VN_WhiteBalance [▶ 1275]

TC3 Vision Code Reading

F_VN_ReadBarcode [▶ 839]

F_VN_ReadBarcodeExp [▶ 841]

F_VN_ReadBarcodeExp2 [▶ 843]

F_VN_ReadDataMatrixCode [▶ 845]

F_VN_ReadDataMatrixCodeExp [▶ 847]

F_VN_ReadDataMatrixCodeExp2 [▶ 849]

F_VN_ReadPharmaCode [▶ 850]

F_VN_ReadPharmaCodeExp [▶ 852]

F_VN_ReadPharmaCodeExp2 [▶ 854]

F_VN_ReadQRCode [▶ 855]

F_VN_ReadQRCodeExp [▶ 858]

F_VN_ReadQRCodeExp2 [▶ 859]

TC3 Vision Code Quality

F_VN_GradeBarcode [▶ 819]

F_VN_GradeBarcodeExp [▶ 822]

F_VN_GradeDataMatrixCode [▶ 826]

F_VN_GradeDataMatrixCodeExp [▶ 828]

F_VN_GradeQRCode [▶ 831]

F_VN_GradeQRCodeExp [▶ 834]

TC3 Vision Metrology 2D

F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1486]

F_VN_CalibrateCamera [▶ 1073]

F_VN_CalibrateCameraExp [▶ 1076]

F_VN_CalibrateCameraExp2 [▶ 1079]

F_VN_CalibrateCameraExp3 [▶ 1081]

F_VN_CalibrateCameraManually [▶ 1083]

F_VN_CalibrateCameraManuallyExp [▶ 1085]

F_VN_CalibrateLinescanCamera [▶ 1088]

F_VN_CalibrateLinescanCameraExp [▶ 1090]

F_VN_ClosestPointsBF [▶ 1487]

F_VN_CompensateLensDistortion [▶ 1092]

F_VN_CompensateLensDistortionExp1 [▶ 1093]

F_VN_CompensateLensDistortionExp2 [▶ 1094]

F_VN_CompensateLensDistortionForPoints [▶ 1095]

F_VN_CompensateLensDistortionForPointsExp1 [▶ 1096]

F_VN_CompensateLensDistortionForPointsExp2 [▶ 1097]

F_VN_DetectPatternPoints [▶ 1098]

F_VN_DetectPatternPointsExp [▶ 1099]

F_VN_GenerateCalibrationPatternReferencePoints [▶ 1101]

F_VN_ImagePointsWorldDistance [▶ 1102]

F_VN_LocateCircularArc [▶ 1489]

F_VN_LocateCircularArcExp [▶ 1492]

F_VN_LocateEdge [▶ 1495]

F_VN_LocateEdgeExp [▶ 1498]

F_VN_LocateEdges [▶ 1501]

F_VN_LocateEdgesExp [▶ 1504]

F_VN_LocateEllipse [▶ 1507]

F_VN_LocateEllipseExp [▶ 1510]

F_VN_MeasureAngleBetweenEdges [▶ 1513]

F_VN_MeasureAngleBetweenEdgesExp [▶ 1515]

F_VN_MeasureEdgeDistance [▶ 1518]

F_VN_MeasureEdgeDistanceExp [▶ 1520]

F_VN_MeasureMinEdgeDistance [▶ 1523]

F_VN_MeasureMinEdgeDistanceExp [▶ 1525]

F_VN_SolvePnP [▶ 1103]

F_VN_SolvePnPExp [▶ 1105]

F_VN_SortDetectedPatternPoints [▶ 1106]

F_VN_TransformCoordinatesImageToWorld_Container [▶ 1107]

F_VN_TransformCoordinatesImageToWorld_Point [▶ 1108]

F_VN_TransformCoordinatesWorldToImage_Container [▶ 1111]

F_VN_TransformCoordinatesWorldToImage_Point [▶ 1112]

TC3 Vision Matching

F_VN_DrawKeypoints [▶ 1014]

F_VN_DrawKeypointsExp [▶ 1015]

F_VN_DrawMatches [▶ 1026]

F_VN_DrawMatchesExp [▶ 1027]

F_VN_FilterGoodMatches [▶ 1339]

F_VN_FindReferenceKeyPointsInImage [▶ 1340]

F_VN_FindReferenceKeyPointsInImageAKAZE [▶ 1342]

F_VN_FindReferenceKeyPointsInImageAKAZEExp [▶ 1344]

F_VN_FindReferenceKeyPointsInImageBRISK [▶ 1346]

F_VN_FindReferenceKeyPointsInImageBRISKExp [▶ 1348]

F_VN_FindReferenceKeyPointsInImageExp [▶ 1350]

F_VN_FindReferenceKeyPointsInImageORB [▶ 1352]

F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1354]

F_VN_GetMatchCoordinates [▶ 1356]

F_VN_KeyPointsAGAST [▶ 1357]

F_VN_KeyPointsAGASTExp [▶ 1358]

F_VN_KeyPointsAndDescriptorsAKAZE [▶ 1359]

F_VN_KeyPointsAndDescriptorsAKAZEExp [▶ 1359]

F_VN_KeyPointsAndDescriptorsBRISK [▶ 1360]

F_VN_KeyPointsAndDescriptorsBRISKExp [▶ 1361]

F_VN_KeyPointsAndDescriptorsKAZE [▶ 1362]

F_VN_KeyPointsAndDescriptorsKAZEExp [▶ 1363]

F_VN_KeyPointsAndDescriptorsORB [▶ 1364]

F_VN_KeyPointsAndDescriptorsORBExp [▶ 1365]

F_VN_KeyPointsFAST [▶ 1366]

F_VN_KeyPointsFASTExp [▶ 1367]

F_VN_KeyPointsGFTT [▶ 1368]

F_VN_KeyPointsGFTTExp [▶ 1369]

F_VN_KeyPointsMSER [▶ 1370]

F_VN_KeyPointsMSERExp [▶ 1371]

F_VN_KeyPointsSB [▶ 1372]

F_VN_KeyPointsSBExp [▶ 1373]

F_VN_MatchContours [▶ 983]

F_VN_MatchContours1vsN [▶ 984]

F_VN_MatchContours1vsNExp [▶ 986]

F_VN_MatchContoursExp [▶ 987]

F_VN_MatchDescriptorsBF [▶ 1374]

F_VN_MatchDescriptorsBFExp [▶ 1375]

F_VN_MatchDescriptorsFlannLsh [▶ 1376]

F_VN_MatchDescriptorsFlannLshExp [▶ 1377]

F_VN_MatchDescriptorsKnnBF [▶ 1378]

F_VN_MatchDescriptorsKnnBFExp [▶ 1379]

F_VN_MatchDescriptorsKnnFlannLsh [▶ 1380]

F_VN_MatchDescriptorsKnnFlannLshExp [▶ 1381]

F_VN_MatchImageHuMoments [▶ 1198]

F_VN_MatchTemplate [▶ 1199]

F_VN_MatchTemplateAndEvaluate [▶ 1200]

F_VN_MatchTemplateAndEvaluateExp [▶ 1201]

F_VN_MatchTemplateExp [▶ 1203]

F_VN_RegionsMSER [▶ 1382]

F_VN_RegionsMSERExp [▶ 1383]

TC3 Vision Machine Learning

F_VN_CreateBoostClassifier [▶ 1385]

F_VN_CreateBoostClassifierExp [▶ 1387]

F_VN_CreateBoostClassifierExp2 [▶ 1388]

F_VN_CreateKmppModel [▶ 1390]

F_VN_CreateKmppModelExp [▶ 1392]

F_VN_CreateKnnModel [▶ 1394]

F_VN_CreateLbgModel [▶ 1395]

F_VN_CreateLbgModelExp [▶ 1397]

F_VN_CreateLdaTransform [▶ 1399]

F_VN_CreateLdaTransformViaComponentNum [▶ 1401]

F_VN_CreateNbcModel [▶ 1402]

F_VN_CreatePcaTransform [▶ 1404]

F_VN_CreatePcaTransformViaComponentNum [▶ 1405]

F_VN_CreatePcaTransformViaVariance [▶ 1407]

F_VN_CreateRTreesModel [▶ 1408]

F_VN_CreateRTreesModelExp [▶ 1410]

F_VN_CreateRTreesModelExp2 [▶ 1412]

F_VN_CreateStaModel [▶ 1414]

F_VN_CreateStaModelExp [▶ 1415]

F_VN_CreateStaModelExp2 [▶ 1417]

F_VN_CreateSvmModel [▶ 1419]

F_VN_CreateSvmModelExp [▶ 1422]

F_VN_CreateSvmModelExp2 [▶ 1425]

F_VN_CreateSvmSgdClassifier [▶ 1427]

F_VN_CreateSvmSgdClassifierExp [▶ 1429]

F_VN_FeatureScaling [▶ 1431]

F_VN_FeatureScalingExp [▶ 1433]

F_VN_FeatureTransform [▶ 1434]

F_VN_GetBatchClusters [▶ 1436]

F_VN_GetBatchClustersExp [▶ 1437]

F_VN_GetBatchNovelties [▶ 1439]

F_VN_GetClusterCenter [▶ 1440]

F_VN_GetClusterNum [▶ 1442]

F_VN_GetFeatureScales [▶ 1443]

F_VN_GetSampleCluster [▶ 1445]

F_VN_GetSampleClusterExp [▶ 1446]

F_VN_GetSampleNovelty [▶ 1448]

F_VN_Granulometry [▶ 1554]

F_VN_HaralickFeatures [▶ 1556]

F_VN_InverseFeatureScaling [▶ 1449]

F_VN_InverseFeatureScaling_REAL [▶ 1451]

F_VN_InverseFeatureScalingExp [▶ 1453]

F_VN_InverseFeatureScalingExp_REAL [▶ 1454]

F_VN_InverseFeatureTransform [▶ 1456]

F_VN_PredictBatch [▶ 1457]

F_VN_PredictBatchExp [▶ 1459]

F_VN_PredictSampleClass [▶ 1461]

F_VN_PredictSampleClassExp [▶ 1463]

F_VN_PredictSampleScalar [▶ 1464]

F_VN_PredictSampleScalarExp [▶ 1466]

F_VN_PredictSampleVector [▶ 1467]

F_VN_PredictSampleVectorExp [▶ 1469]

F_VN_TrainBatch [▶ 1470]

F_VN_TrainBatchClusters [▶ 1472]

F_VN_TrainSample [▶ 1474]

F_VN_TrainSampleClass [▶ 1475]

F_VN_TrainSampleCluster [▶ 1476]

F_VN_TrainSampleScalar [▶ 1478]

F_VN_TrainSampleVector [▶ 1479]

TC3 Vision OCR

F_VN_OCR [▶ 1550]

F_VN_OCRExp [▶ 1552]

TC3 Vision Neural Network

F_VN_ExecuteNeuralNetwork [▶ 1559]

F_VN_ExecuteNeuralNetwork_MultiLayerOutput [▶ 1560]

F_VN_ExecuteNeuralNetwork_SingleLayerOutput [▶ 1561]

F_VN_GetNeuralNetworkLayerNames [▶ 1562]

6.1.4.2 Algebraic Container Operations

This group contains functions for the element-wise application of algebraic operators to containers [▶ 152].

Functions

Algebra

- F_VN_AddContainers [▶ 492]
- F_VN_AddToContainerElements [▶ 465]
- F_VN_DivideContainers(Exp) [▶ 496]
- F_VN_MultiplyContainers [▶ 500]
- F_VN_MultiplyWithContainerElements [▶ 501]
- F_VN_NegateContainer [▶ 504]
- F_VN_SubtractContainers [▶ 505]

Bitwise operators

- F_VN_BitwiseAndContainers [▶ 493]
- F_VN_BitwiseNotContainer [▶ 494]
- F_VN_BitwiseOrContainers [▶ 494]
- F_VN_BitwiseXorContainers [▶ 495]

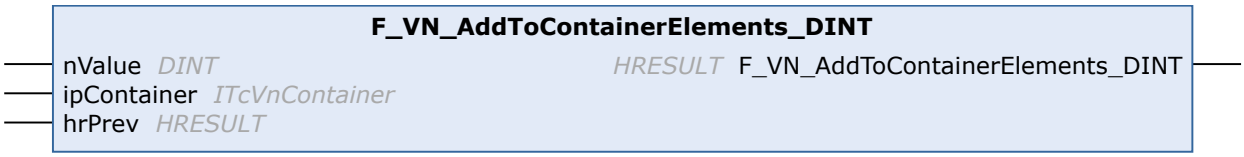
Statistical features

- F_VN_MaxContainer [▶ 498]
- F_VN_MinContainer [▶ 499]

6.1.4.2.1 F_VN_AddToContainerElements

This chapter contains functions to add a value to each container element according to the data type.

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 [▶ 370]	Container with DINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

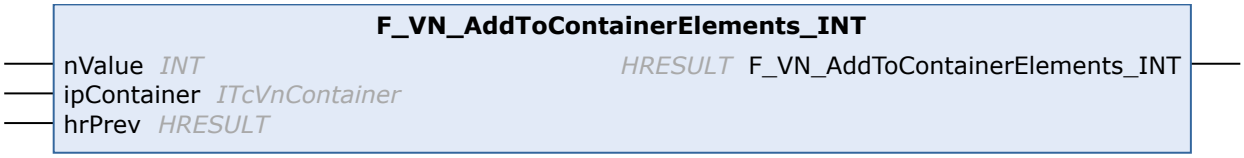
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with INT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

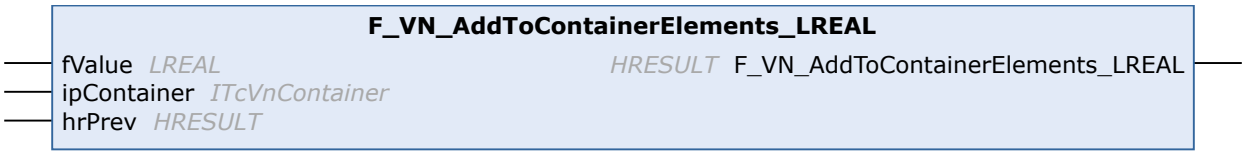
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with LREAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

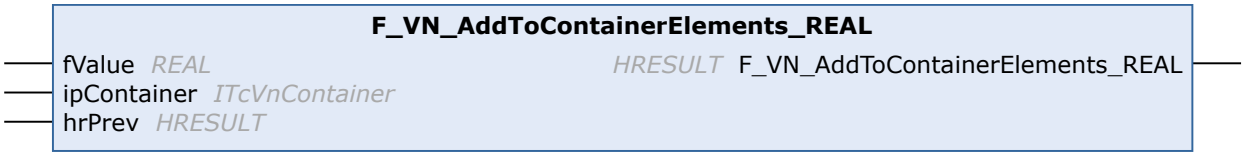
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with SINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

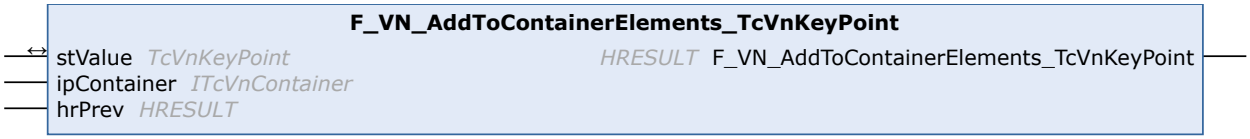
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnKeyPoint elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 231]	Value

Return value

HRESULT [▶ 141]

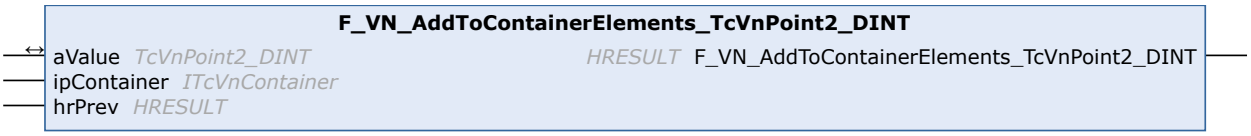
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 160]	Value

Return value

HRESULT [▶ 141]

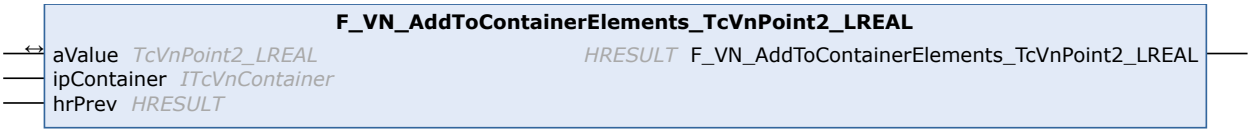
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_LREAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 160]	Value

Return value

HRESULT [▶ 141]

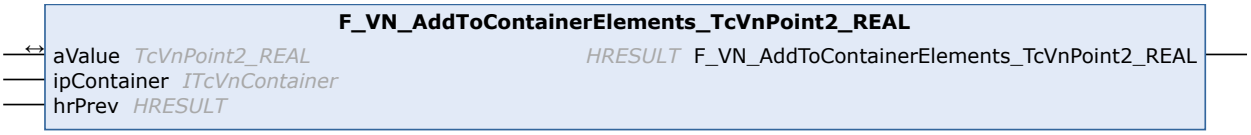
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 160]	Value

Return value

HRESULT [▶ 141]

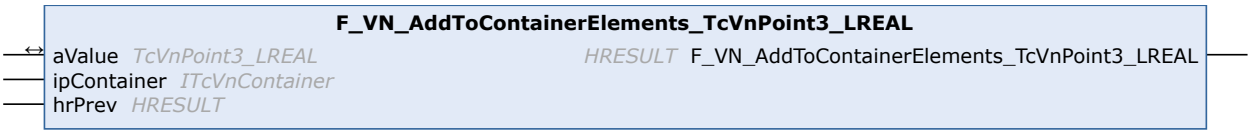
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_LREAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 160]	Value

Return value

HRESULT [▶ 141]

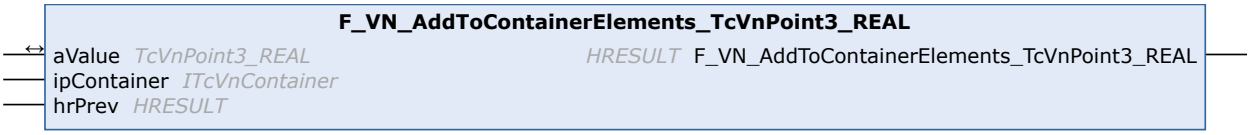
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 160]	Value

Return value

HRESULT [▶ 141]

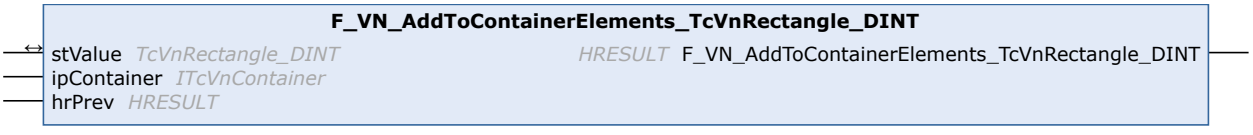
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnRectangle_DINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 244]	Value

Return value

HRESULT [▶ 141]

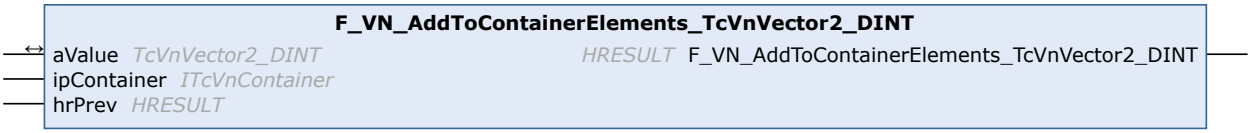
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_DINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

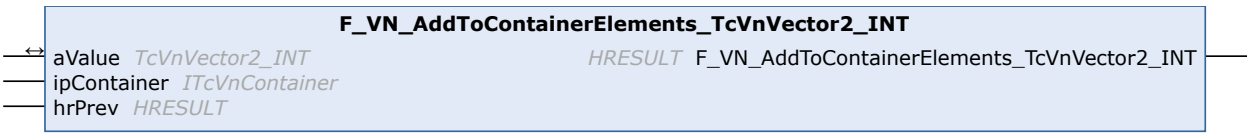
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_INT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 162]	Value

Return value

HRESULT [▶ 141]

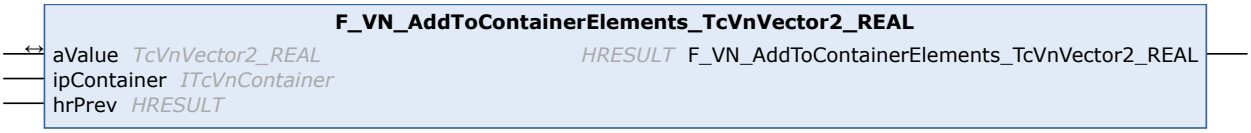
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

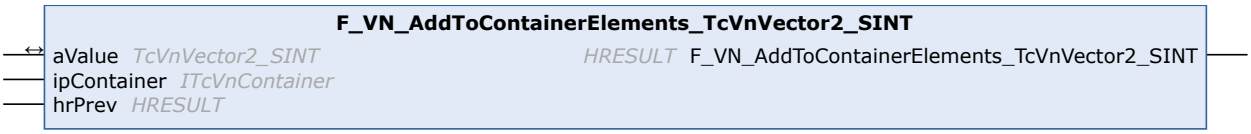
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_SINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

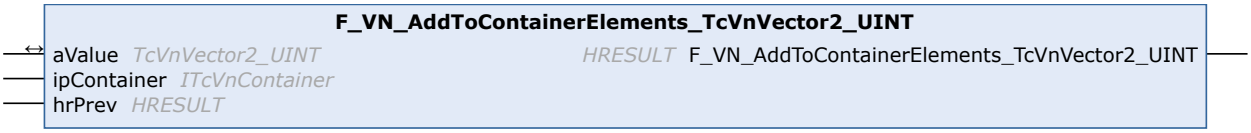
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_UINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

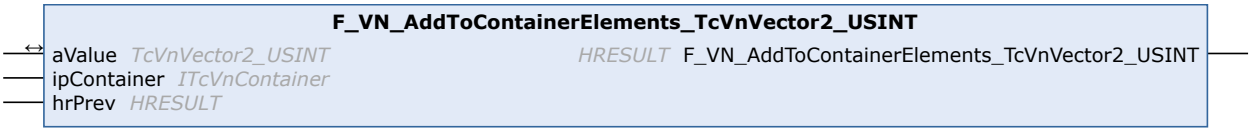
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector2_USINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

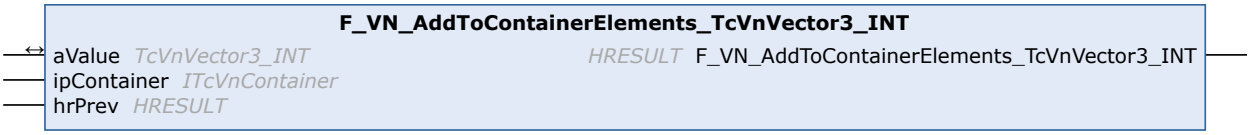
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector3_INT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 162]	Value

Return value

HRESULT [▶ 141]

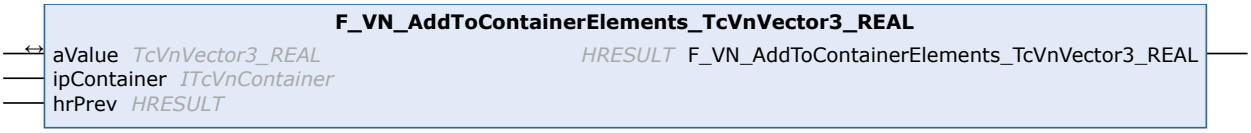
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector3_REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

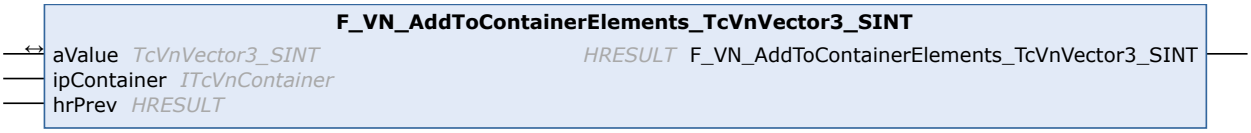
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector3_SINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

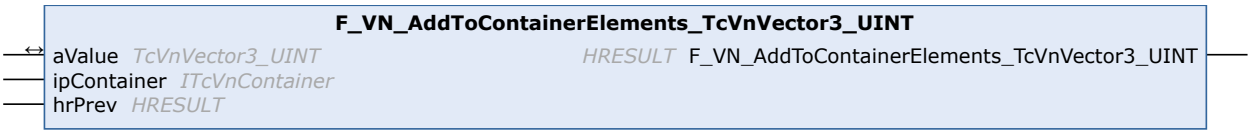
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector3_UINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

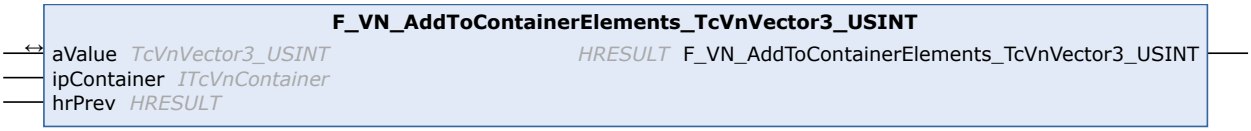
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector3_USINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

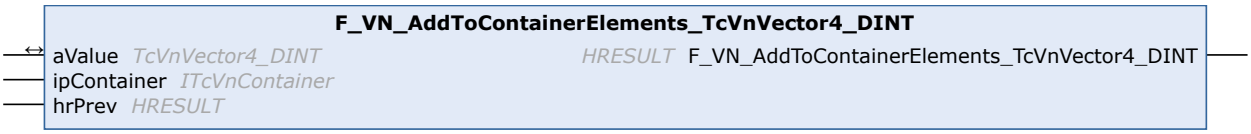
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_DINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

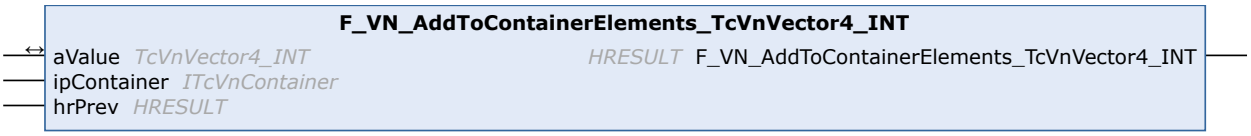
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_INT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 162]	Value

Return value

HRESULT [▶ 141]

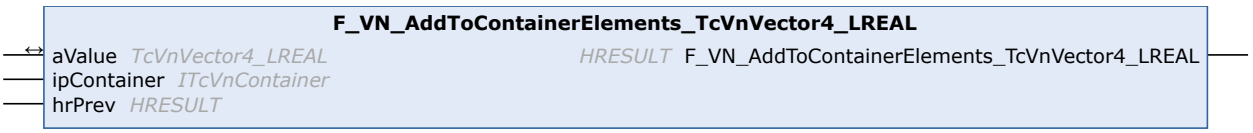
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_LREAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

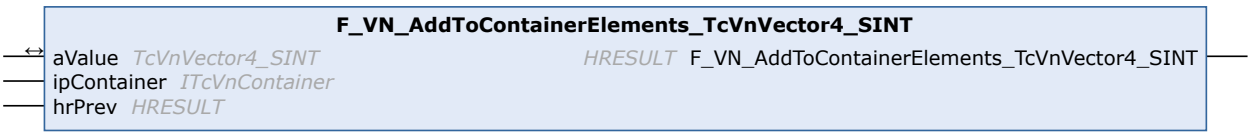
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_SINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

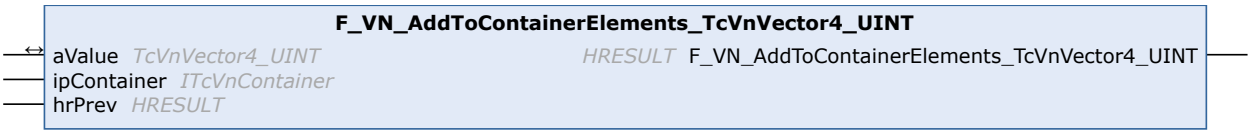
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_UINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

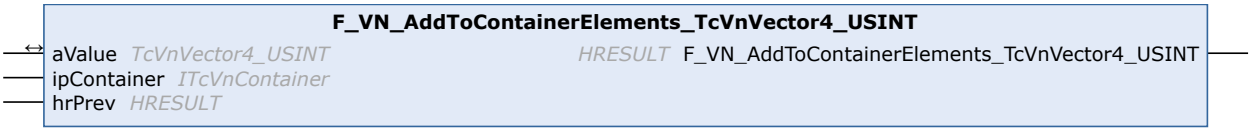
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_USINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 162]	Value

Return value

HRESULT [▶ 141]

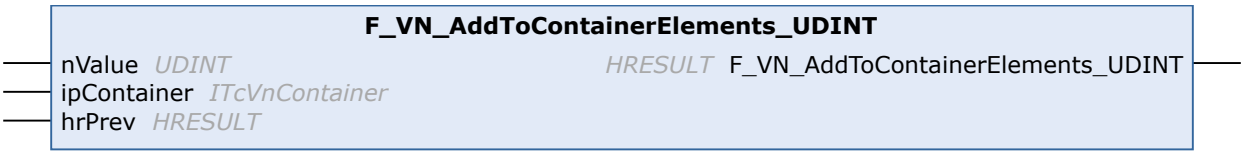
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with UDINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

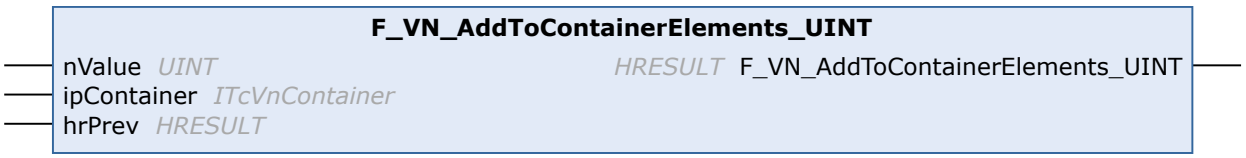
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with UINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

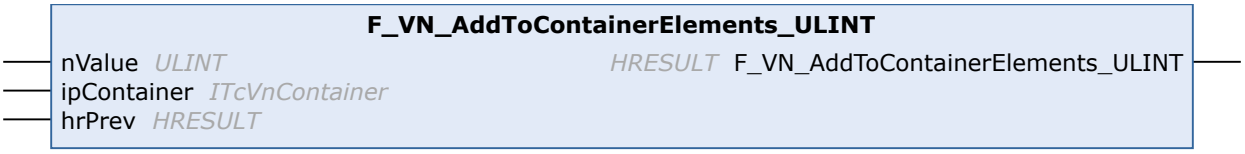
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with ULINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

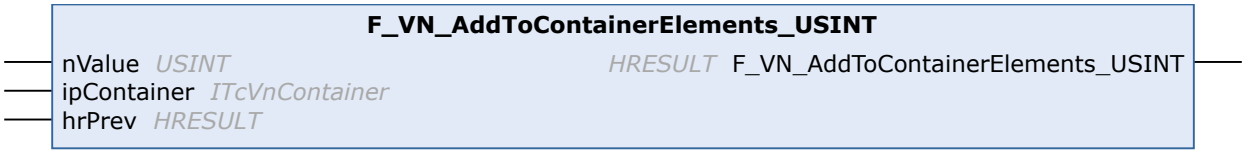
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with USINT elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

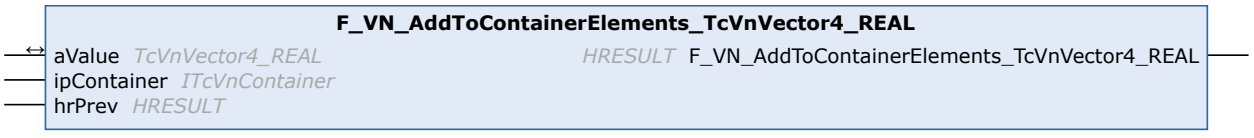
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnVector4_REAL elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

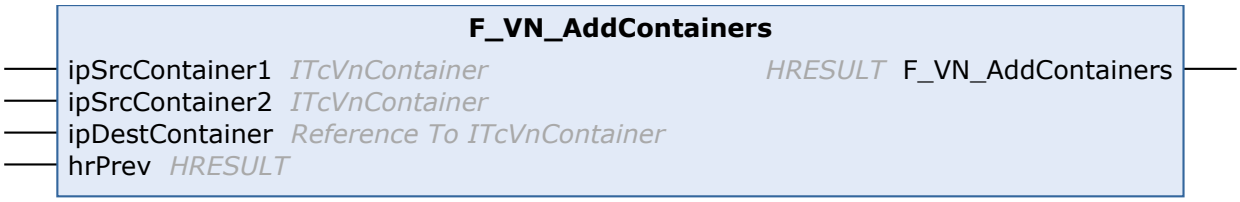
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

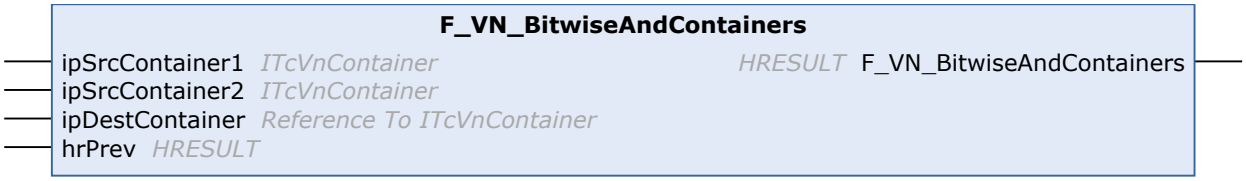
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

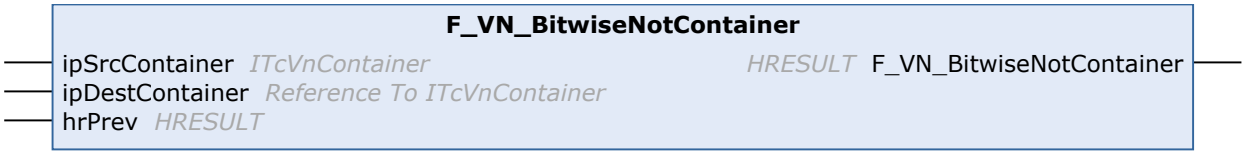
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
ipSrcContainer1	ITcVnContainer [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

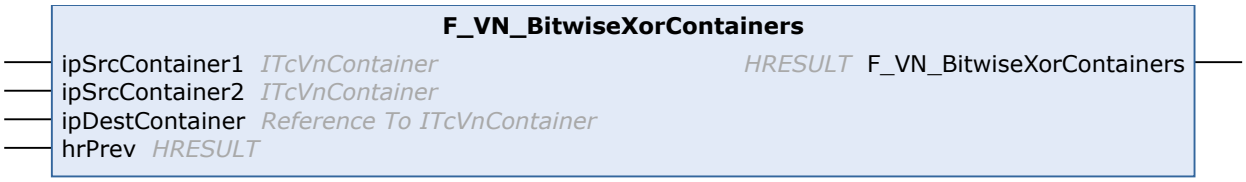
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
ipSrcContainer1	ITcVnContainer [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

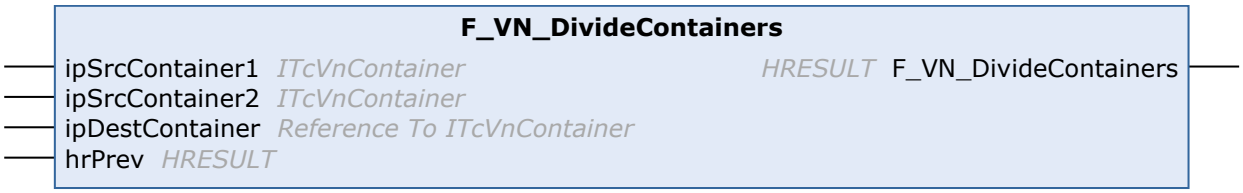
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer 2	ITcVnContainer [▶ 370]	Source container 2
ipDestContaine r	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

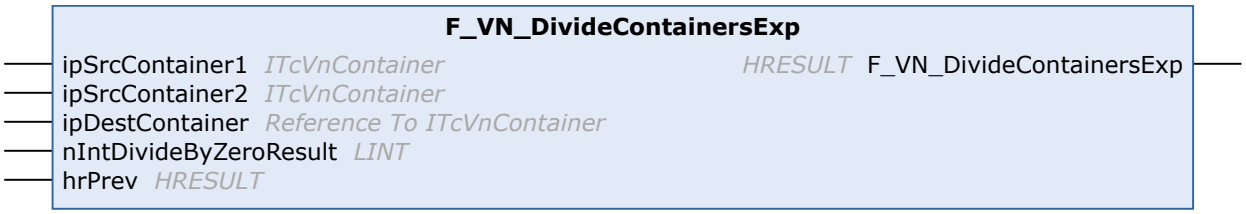
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
nIntDivideByZeroResult	LINT	Value set as result in case of integer division by zero
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

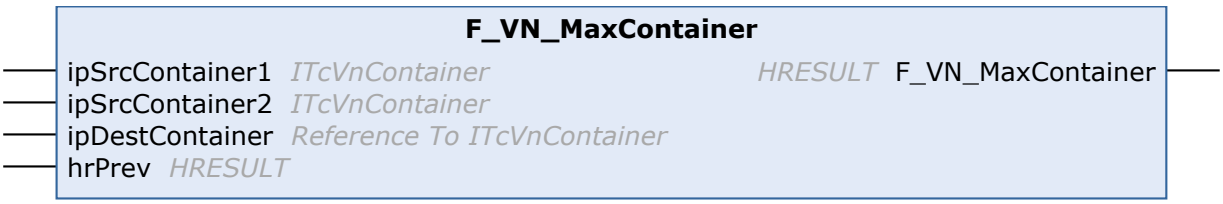
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
ipSrcContainer 1	ITcVnContainer [▶ 370]	Source container 1
ipSrcContainer 2	ITcVnContainer [▶ 370]	Source container 2
ipDestContaine r	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

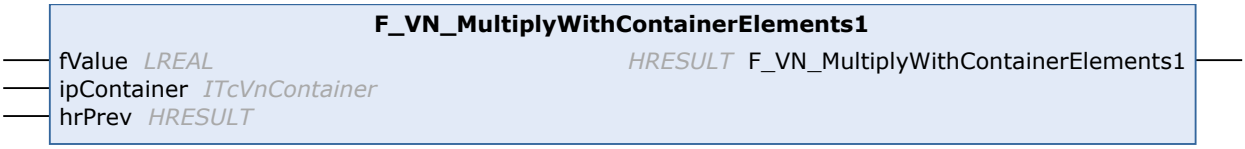
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 1-dimensional elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

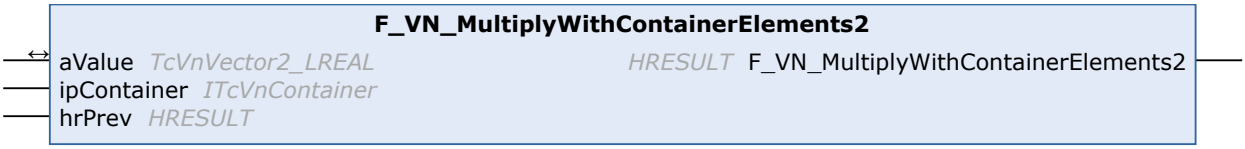
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 2-dimensional elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_LREAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

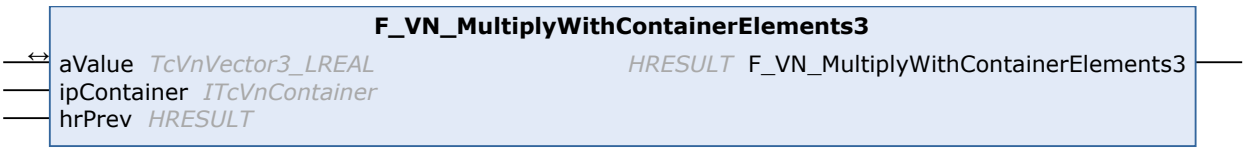
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 3-dimensional elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_LREAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

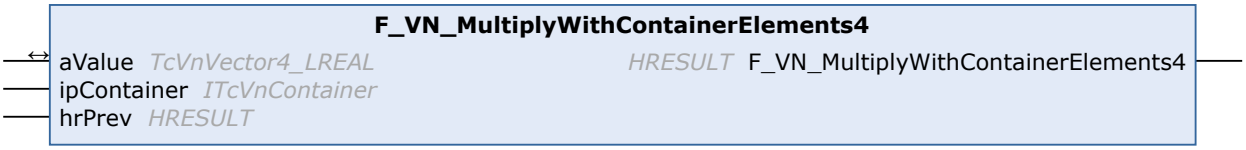
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 4-dimensional elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Value

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container (same type as ipSrcContainer)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

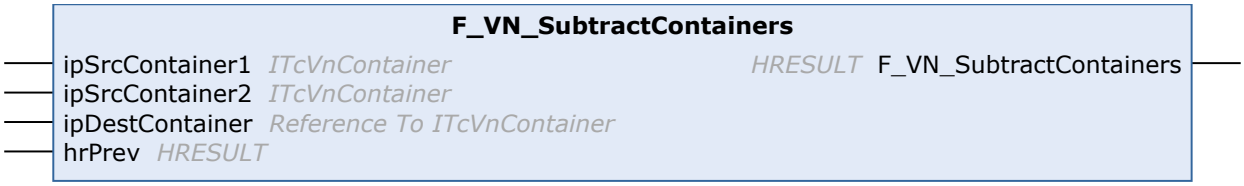
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 370]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the resulting container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3 Algebraic Image Operations

This group contains functions for applying algebraic operations to images [▶ 149] pixel by pixel.

Functions

Addition

- F_VN_AddImages [▶ 507]
- F_VN_AddImagesWeighted(Exp) [▶ 508]
- F_VN_AddScalarToImage [▶ 510]
- F_VN_AddVectorToImage [▶ 511]

Subtraction

- F_VN_SubtractImages [▶ 545]
- F_VN_SubtractImageFromScalar [▶ 543]
- F_VN_SubtractImageFromVector [▶ 544]
- F_VN_SubtractScalarFromImage [▶ 546]
- F_VN_SubtractVectorFromImage [▶ 547]

Multiplication

- F_VN_MultiplyImages [▶ 541]
- F_VN_MultiplyImageWithScalar [▶ 542]
- F_VN_MultiplyImageWithVector [▶ 542]

Division

- F_VN_DivideImages [▶ 526]
- F_VN_DivideImageByScalar [▶ 524]
- F_VN_DivideImageByVector [▶ 525]
- F_VN_DivideScalarByImage [▶ 527]
- F_VN_DivideVectorByImage [▶ 528]

Exponentiation and logarithmization

- F_VN_ElementwiseExp [▶ 529]
- F_VN_ElementwiseLog [▶ 529]

Bitwise AND

- F_VN_BitwiseAndImages(Exp) [▶ 512]
- F_VN_BitwiseAndScalarWithImage [▶ 513]
- F_VN_BitwiseAndVectorWithImage [▶ 514]

Bitwise NOT

- F_VN_BitwiseNotImage(Exp) [▶ 515]

Bitwise OR

- F_VN_BitwiseOrImages(Exp) [▶ 517]
- F_VN_BitwiseOrScalarWithImage [▶ 518]
- F_VN_BitwiseOrVectorWithImage [▶ 519]

Bitwise XOR

- F_VN_BitwiseXorImages(Exp) [▶ 520]
- F_VN_BitwiseXorScalarWithImage [▶ 522]
- F_VN_BitwiseXorVectorWithImage [▶ 522]

Cross-fading

- F_VN_BlendImages [▶ 523]

Maximum

- F_VN_MaxImage(Exp) [▶ 530]
- F_VN_MaxImageWithScalar(Exp) [▶ 532]
- F_VN_MaxImageWithVector(Exp) [▶ 534]

Minimum

- F_VN_MinImage(Exp) [▶ 536]
- F_VN_MinImageWithScalar(Exp) [▶ 537]
- F_VN_MinImageWithVector(Exp) [▶ 539]

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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Samples

- Copy image areas [▶ 2758]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

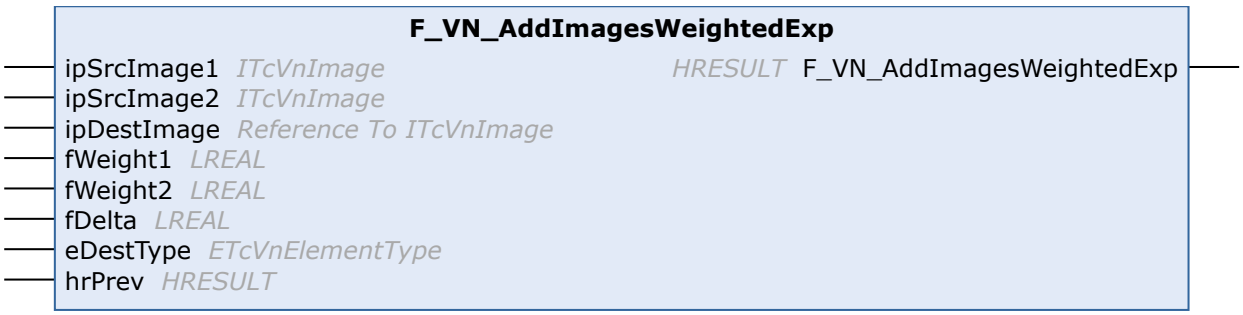
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 199]	Destination image depth
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

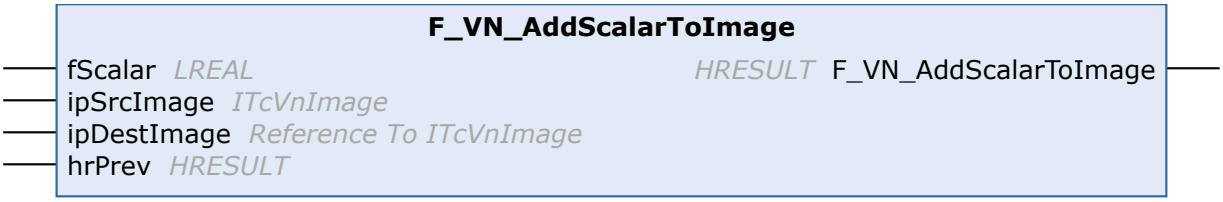
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

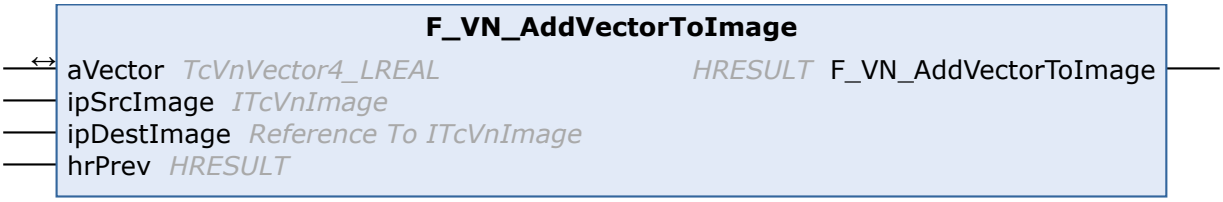
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

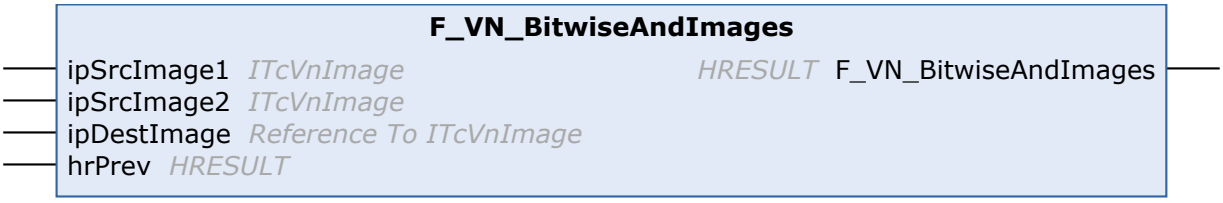
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

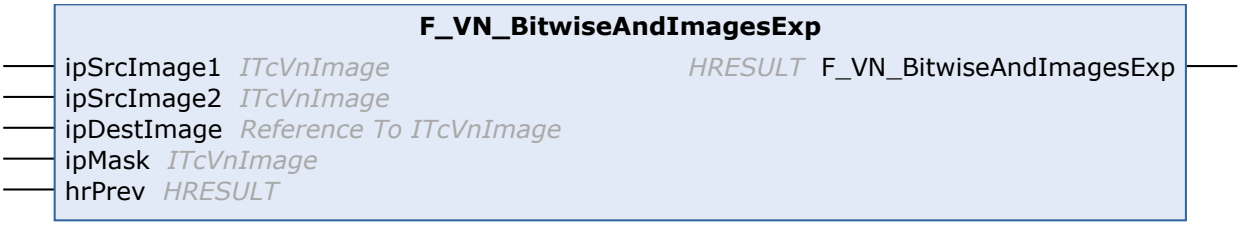
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

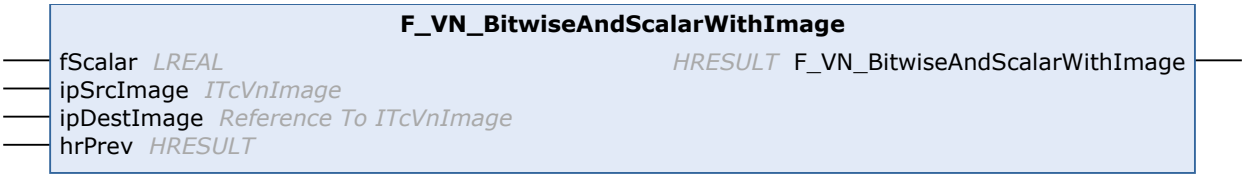
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

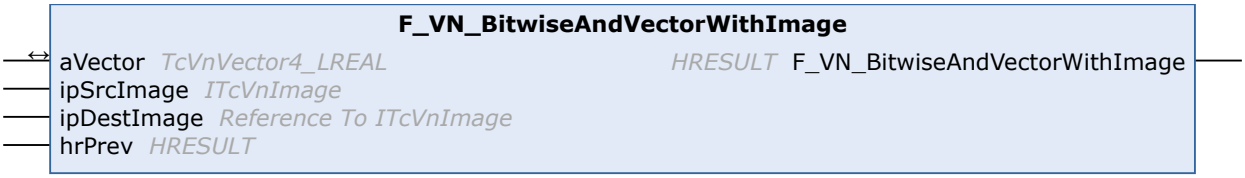
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

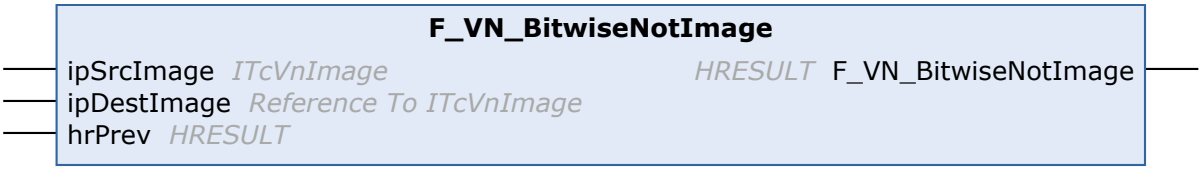
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

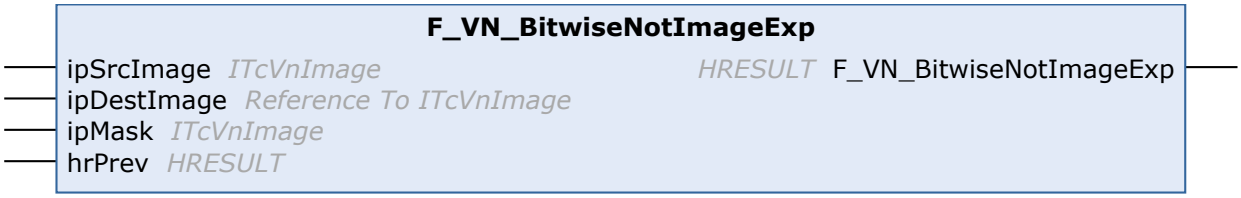
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

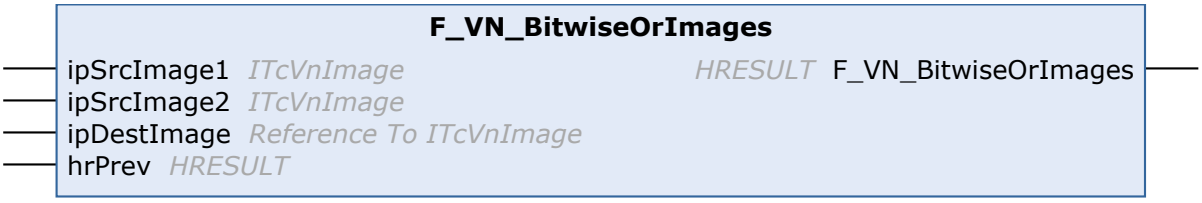
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

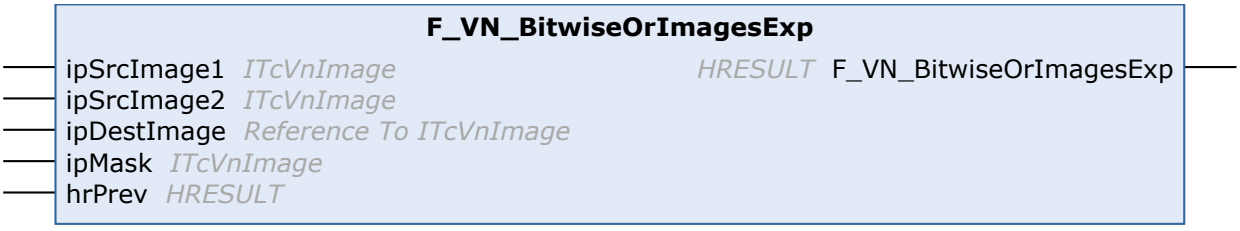
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

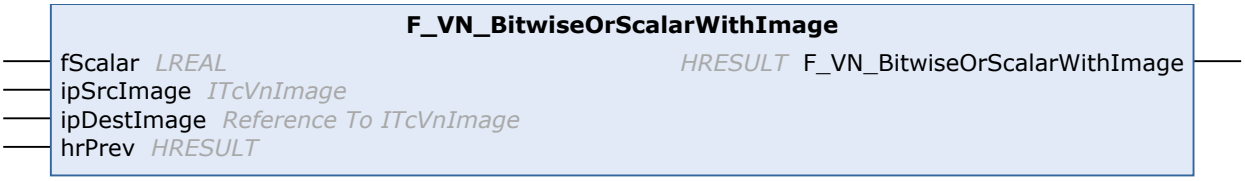
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

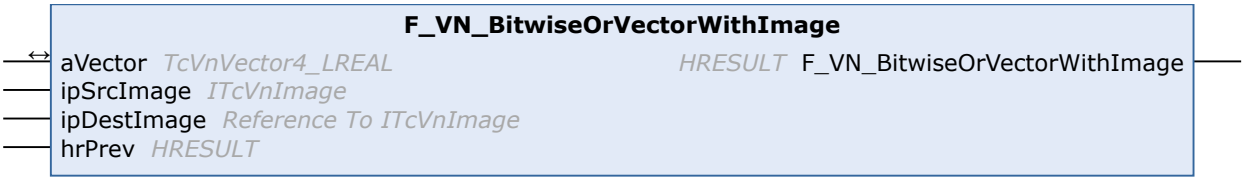
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

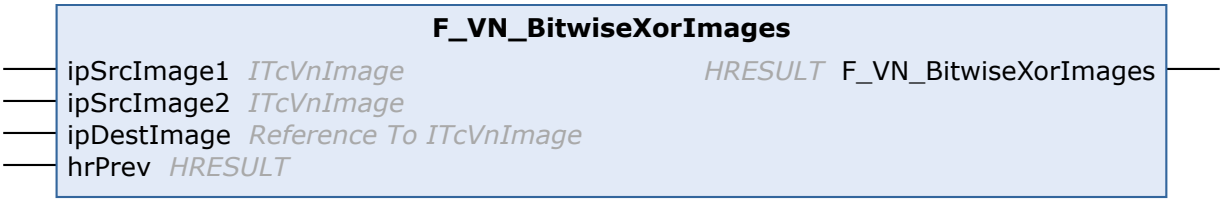
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

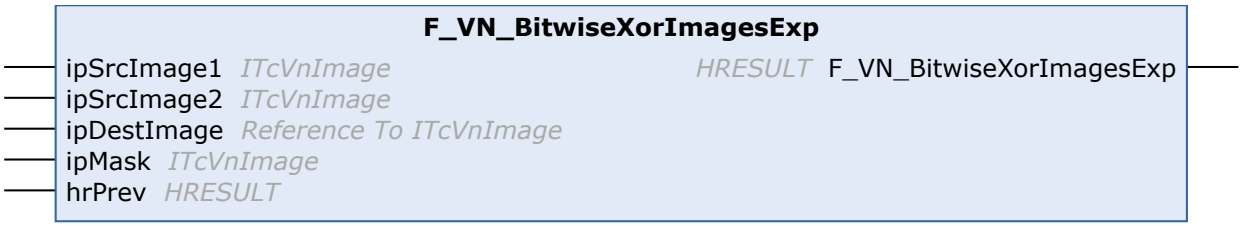
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

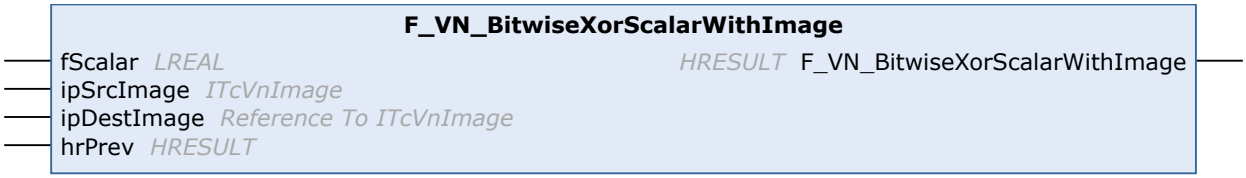
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

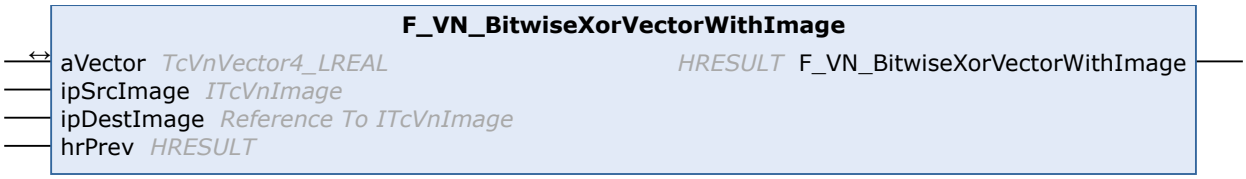
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

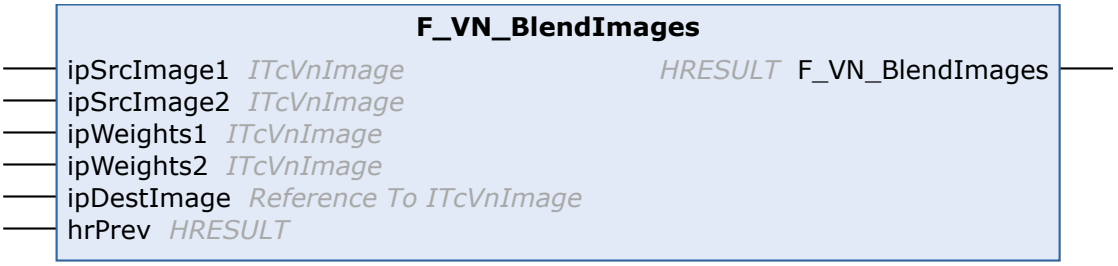
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image (ET_USINT or ET_REAL)
ipSrcImage2	ITcVnImage [▶ 411]	Second source image (same type and size as ipSrcImage1)
ipWeights1	ITcVnImage [▶ 411]	Weights for ipSrcImage1 (1 channel, ET_REAL)
ipWeights2	ITcVnImage [▶ 411]	Weights for ipSrcImage2 (1 channel, ET_REAL)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

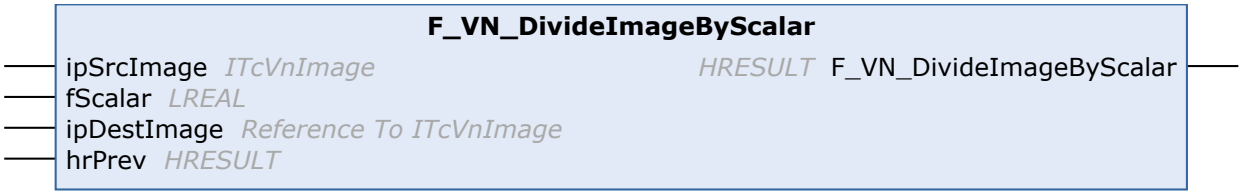
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
fScalar	LREAL	Scalar value
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

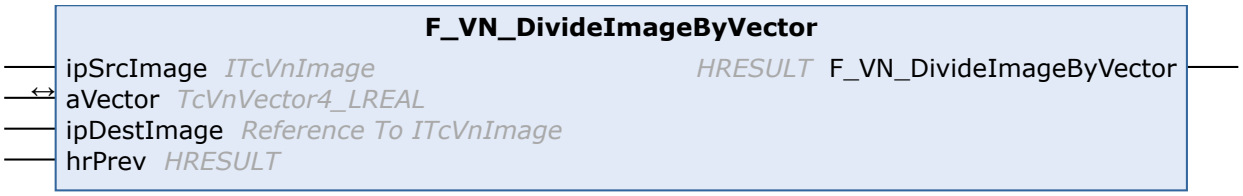
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image (dividend)
ipSrcImage2	ITcVnImage [▶ 411]	Second source image (divisor)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_DivideImages` returns zero for the respective elements in the case of division by zero. However, this only works if the option **Floating point exceptions** of the executing PLC task [▶ 61] is deactivated. Otherwise a division-by-zero error occurs.

NOTICE

Floating point exceptions

This function can cause errors if the option **Floating point exceptions** of the executing PLC task [▶ 61] is active. Therefore, deactivate this option.

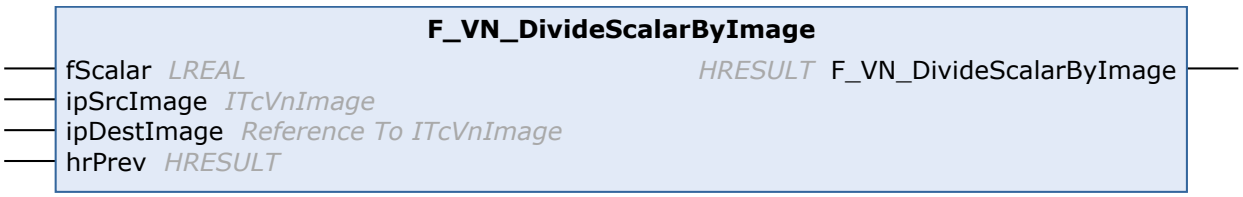
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

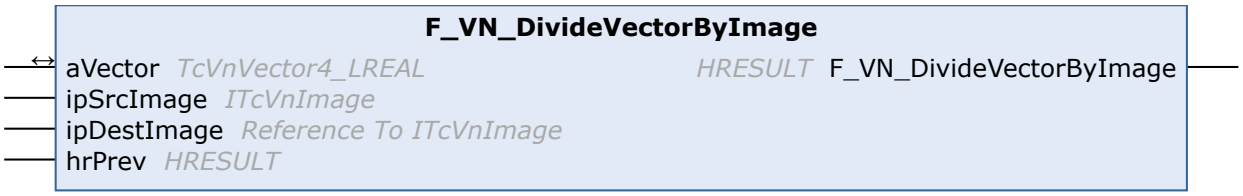
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

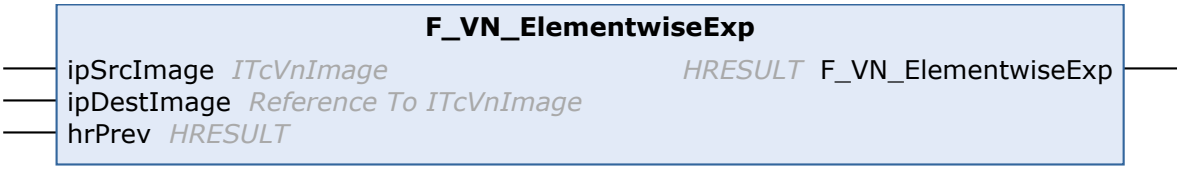
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (REAL or LREAL)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (REAL or LREAL)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

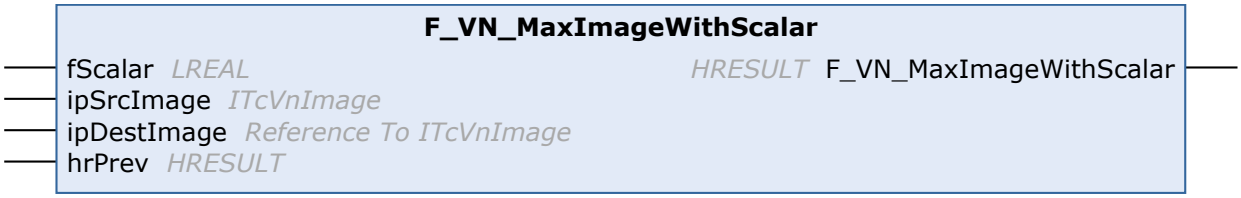
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

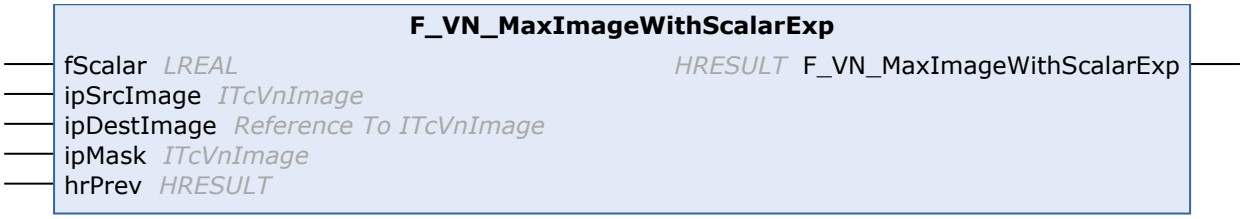
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

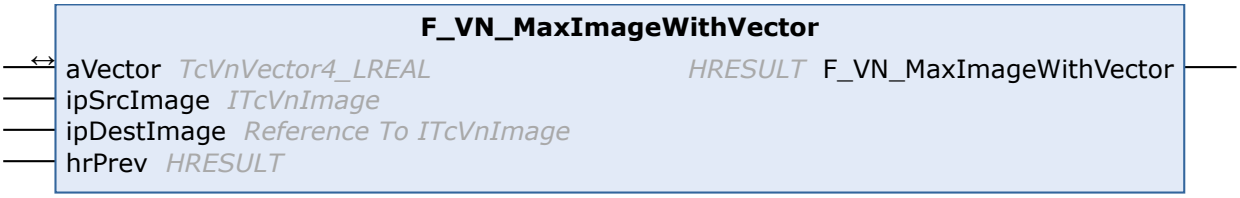
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

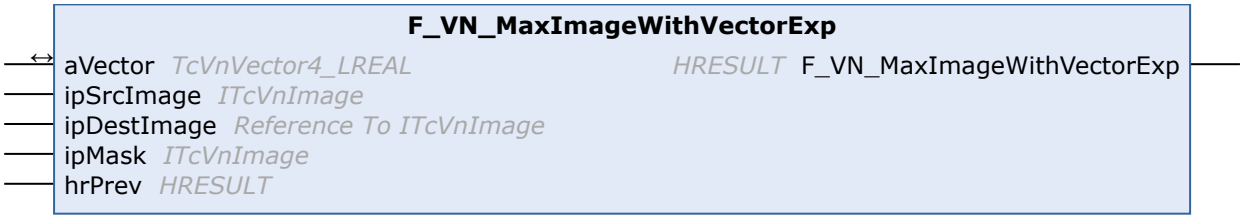
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

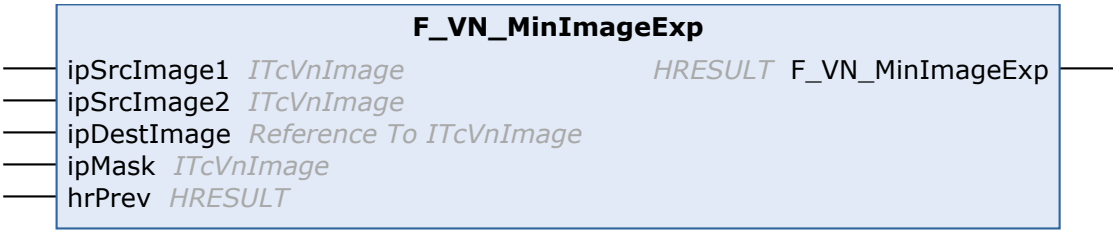
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

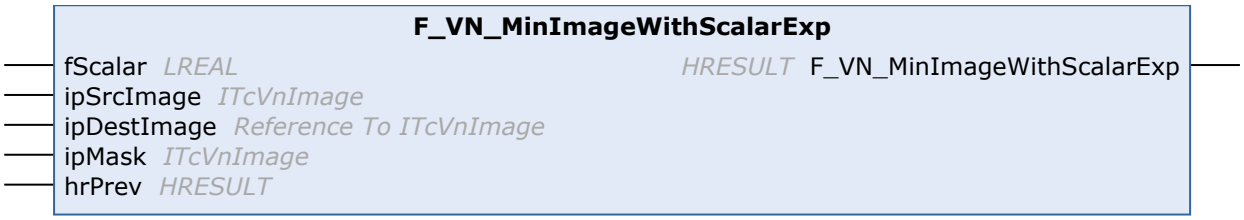
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

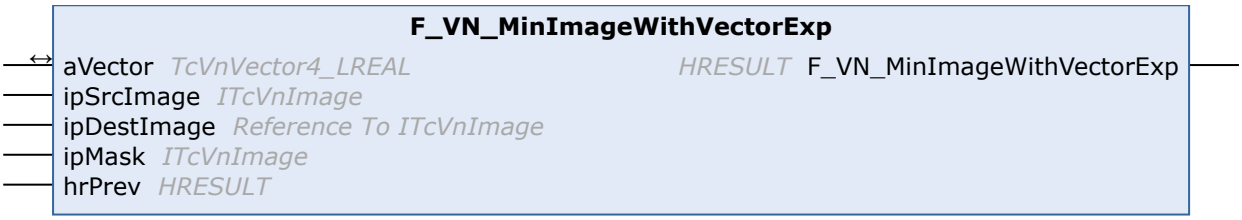
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image
ipSrcImage2	ITcVnImage [▶ 411]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

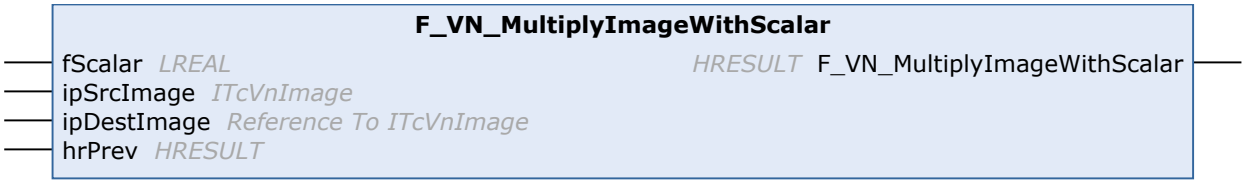
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

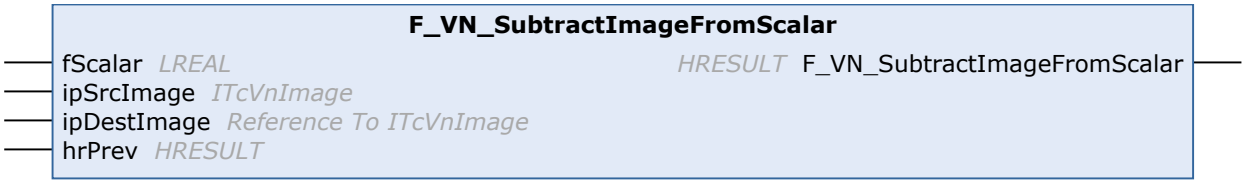
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

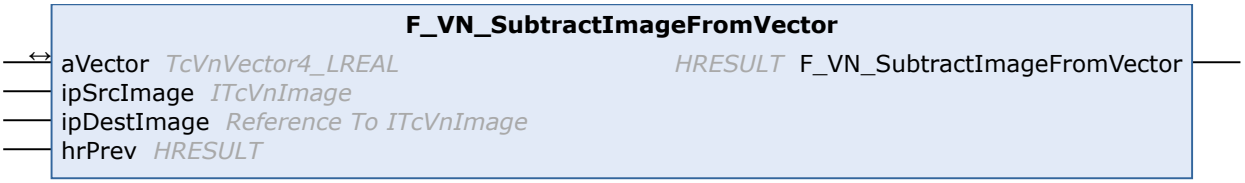
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

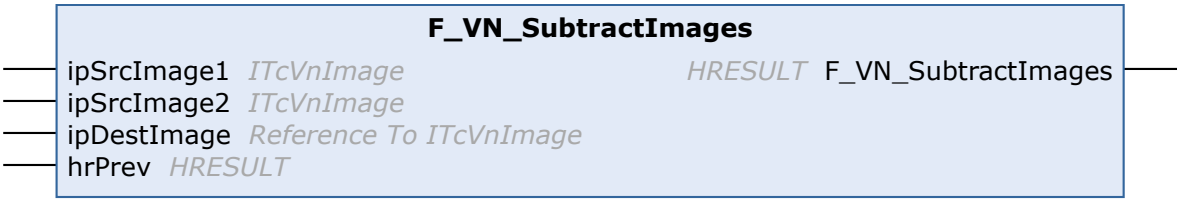
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First source image (minuend)
ipSrcImage2	ITcVnImage [▶ 411]	Second source image (subtrahend)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

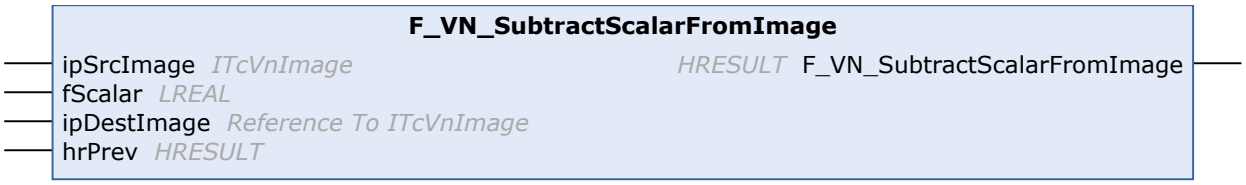
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
fScalar	LREAL	Scalar value
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

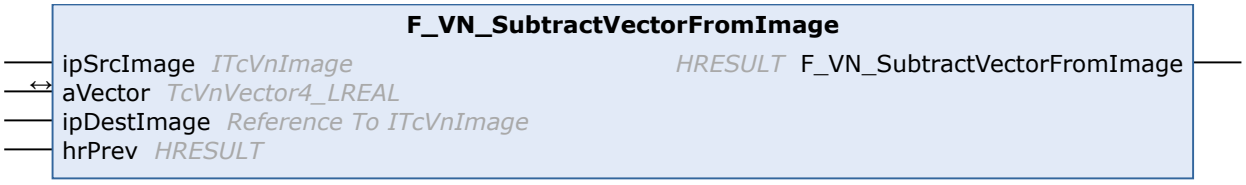
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 162]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4 Basic Container Operations

This group contains functions for handling containers [▶ 152].

Functions

Creation of containers

- F_VN_CopyContainer [▶ 750]
- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751]
- F_VN_CreateContainer [▶ 753]
- F_VN_CreateContainerFromArray [▶ 754]
- F_VN_ExtractContainerRange [▶ 763]
- F_VN_ReserveContainerMemory [▶ 770]

Handling of container elements

- F_VN_AppendToContainer [▶ 549]
- F_VN_CheckIfEmpty [▶ 746]
- F_VN_ConvertContainerType [▶ 748]
- F_VN_EraseFromContainer [▶ 755]
- F_VN_FillContainer(Exp) [▶ 608]
- F_VN_GetAt [▶ 642]
- F_VN_GetContainer(Exp) [▶ 764]
- F_VN_GetNumberOfElements [▶ 766]
- F_VN_InsertIntoContainer [▶ 679]
- F_VN_ReverseContainer [▶ 771]
- F_VN_SetAt [▶ 711]
- F_VN_SetContainer [▶ 772]

Exporting container data to the memory area

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Handling of iterators

- F_VN_AdvanceIterator [▶ 745]
- F_VN_CheckIfIteratorIsAtEnd [▶ 747]
- F_VN_GetForwardIterator [▶ 766]
- F_VN_GetRandomAccessIterator [▶ 767]
- F_VN_IncrementIterator [▶ 768]
- F_VN_IteratorDistance [▶ 769]
- F_VN_SetIteratorToBegin [▶ 772]

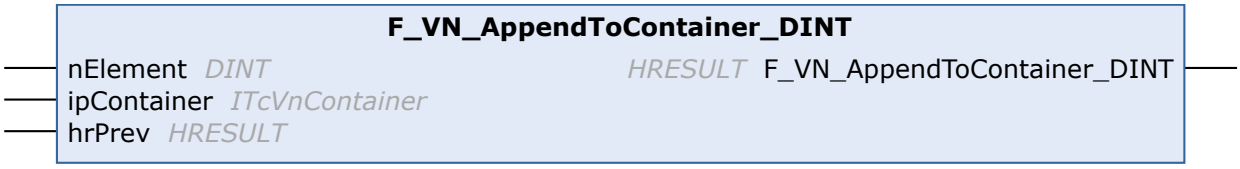
Samples

Basic Container Operations [▶ 2755]

6.1.4.4.1 F_VN_AppendToContainer

This chapter contains functions to add an element to a container according to the data type.

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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

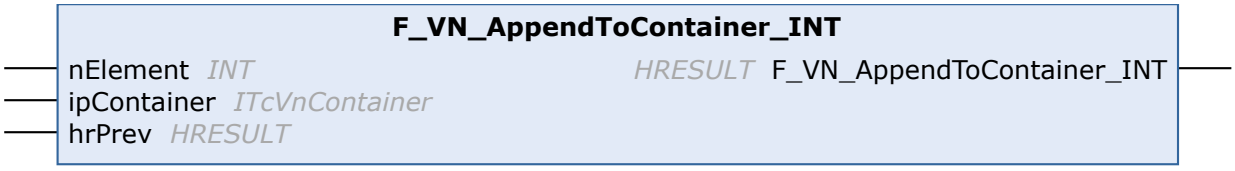
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

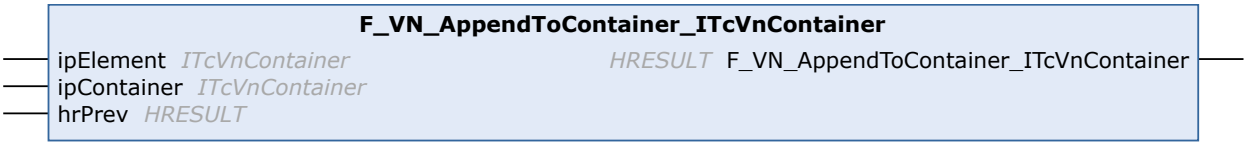
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single element to append to ipContainer or container with several elements to be concatenated
ipContainer	ITcVnContainer [▶ 370]	Container to which the element(s) will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

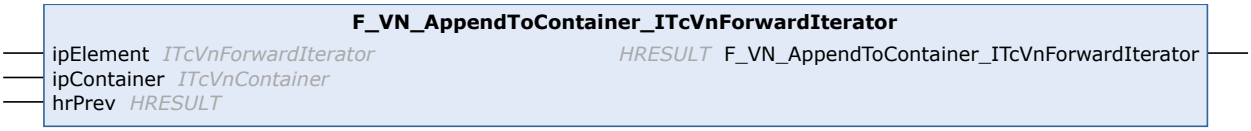
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 370]	Container to which the element(s) will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

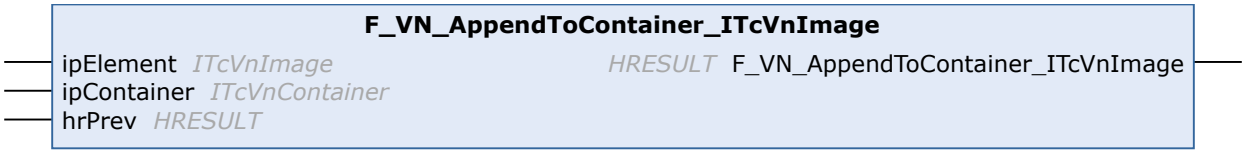
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

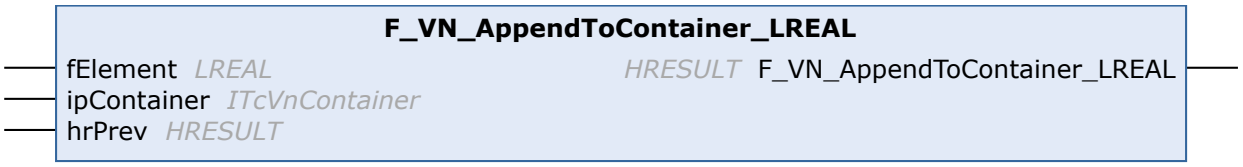
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

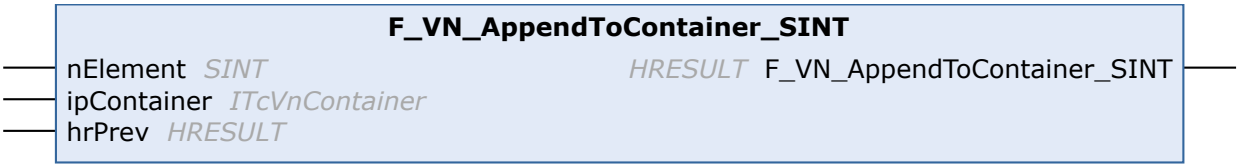
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

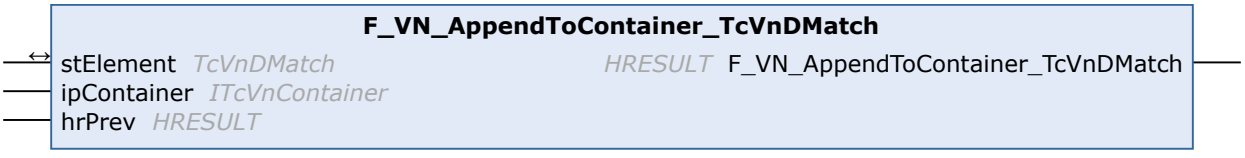
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 230]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

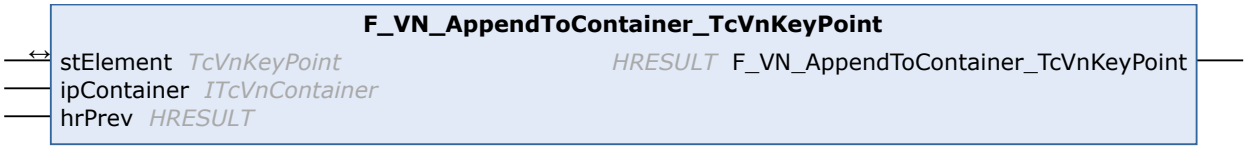
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 231]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

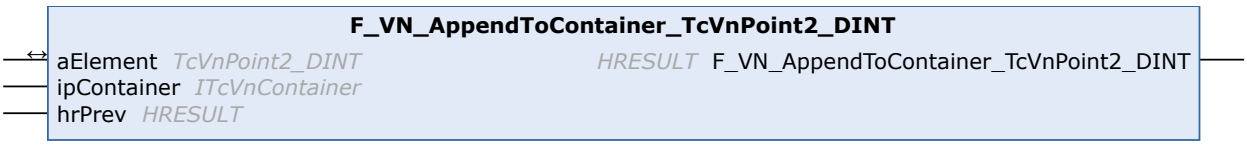
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 160]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

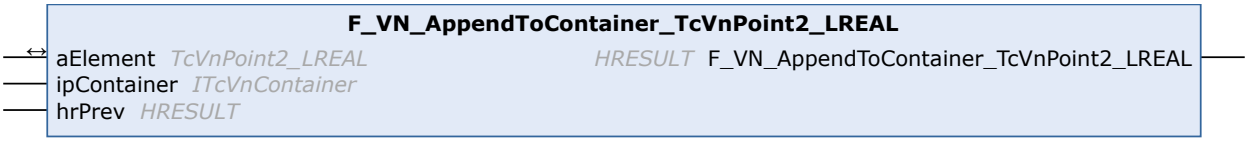
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 160]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

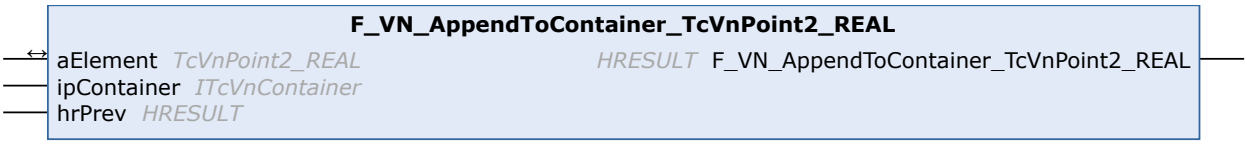
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 160]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

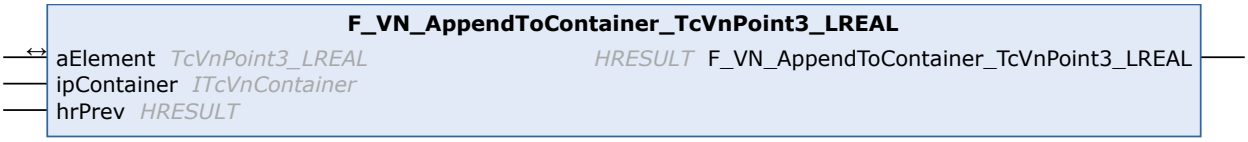
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 160]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

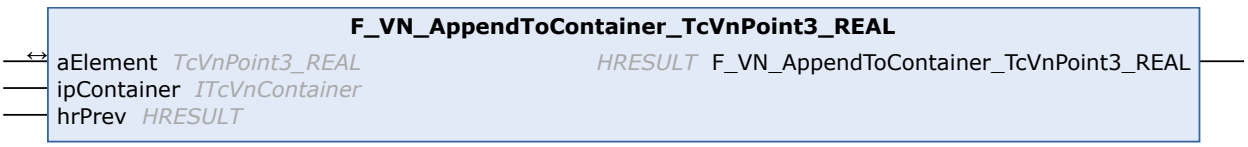
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 160]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

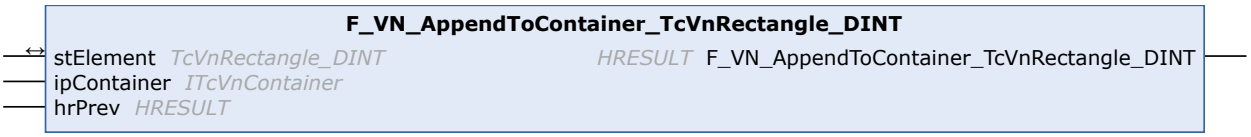
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 244]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

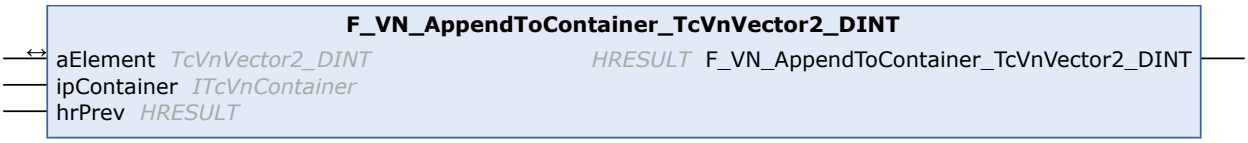
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.17 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

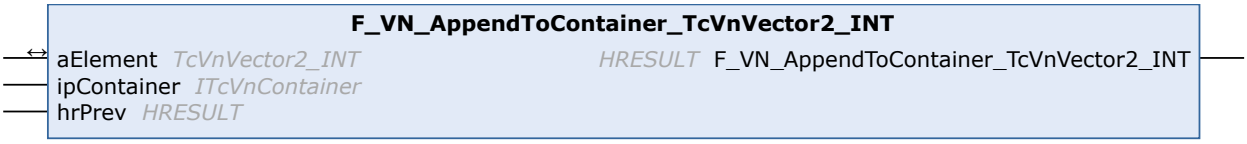
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.18 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

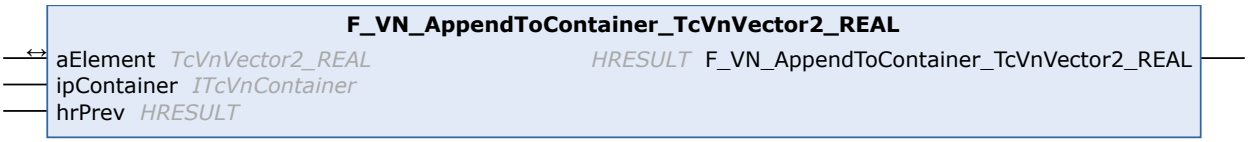
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.19 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

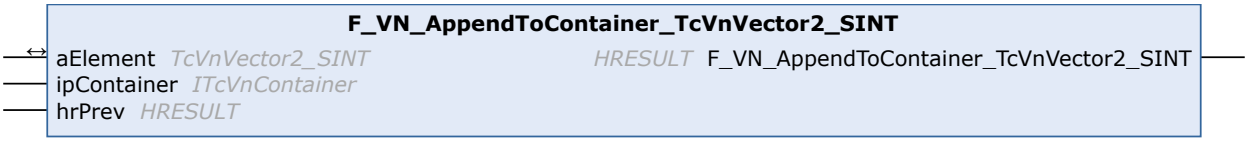
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.20 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

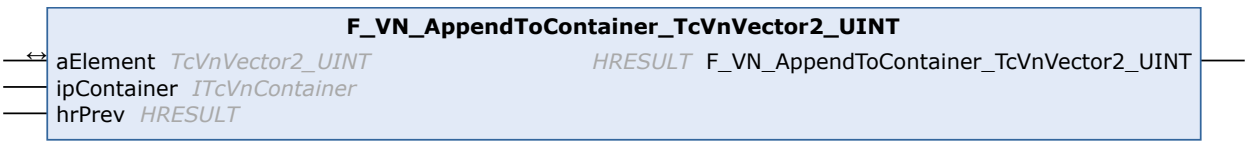
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.21 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

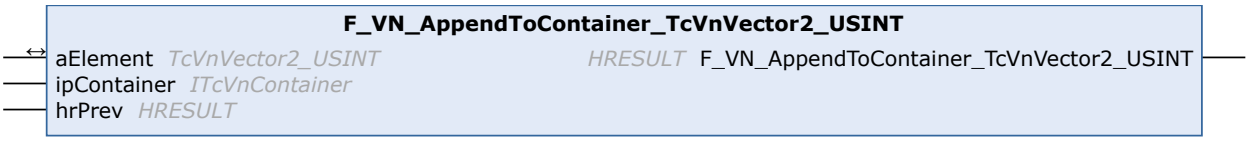
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.22 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

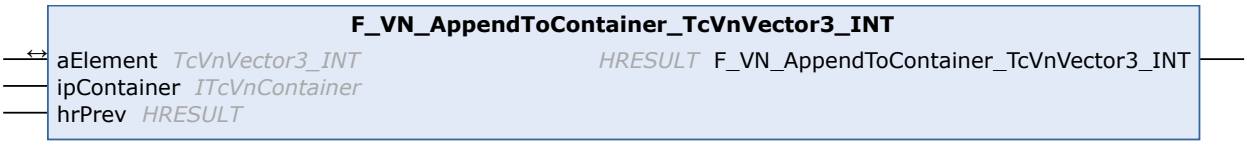
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.23 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

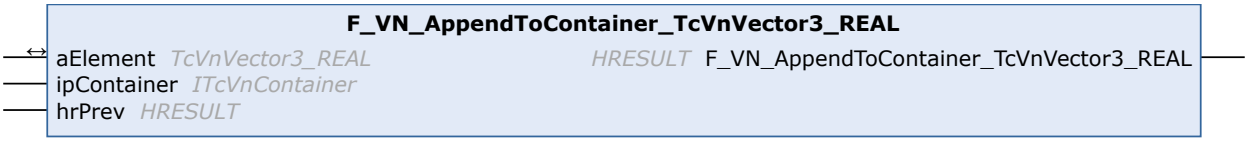
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.24 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

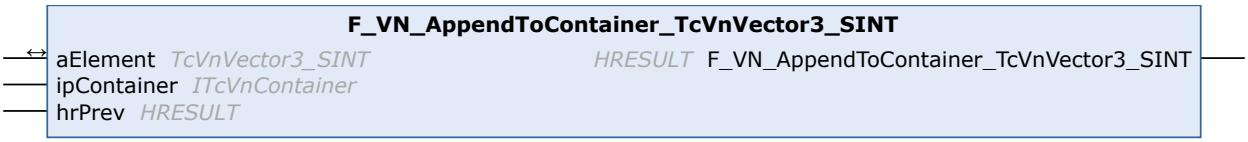
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.25 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

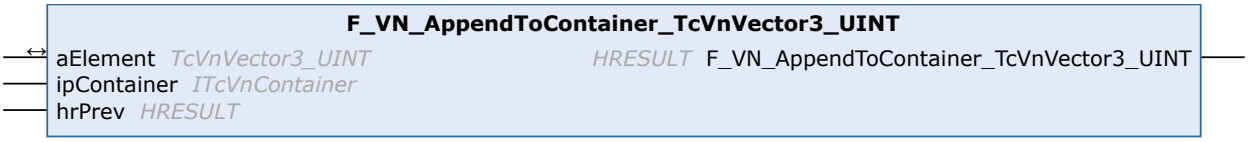
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.26 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

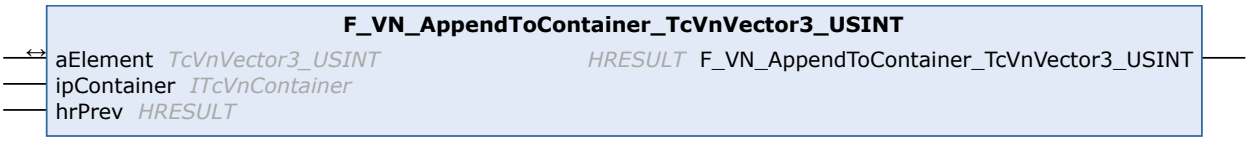
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.27 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

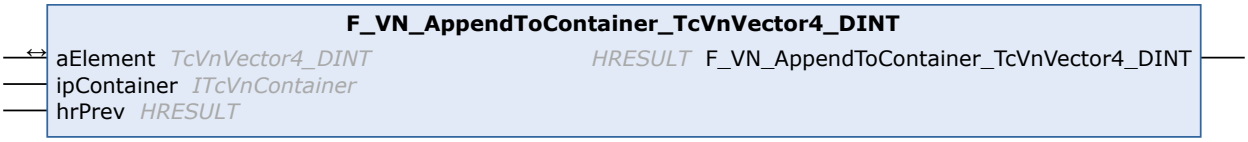
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.28 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

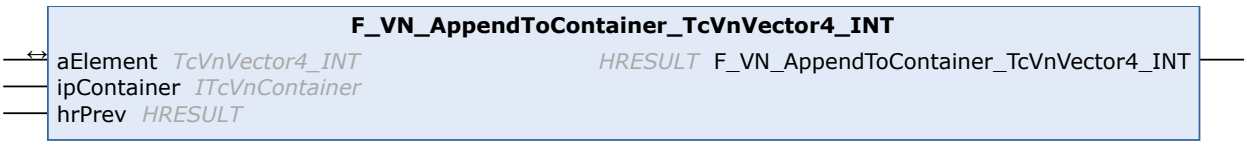
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.29 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.30 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

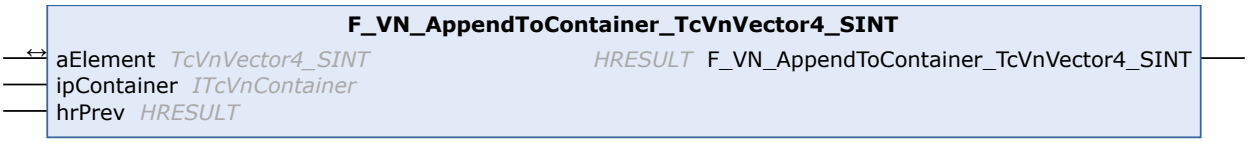
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.31 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

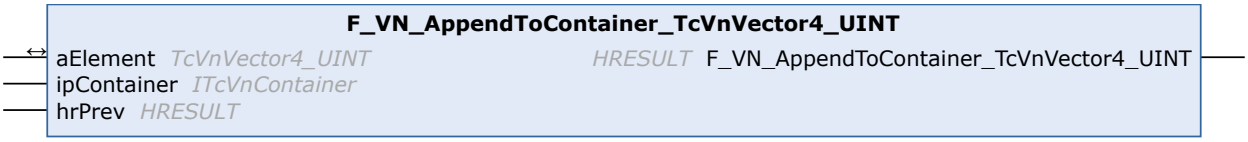
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.32 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

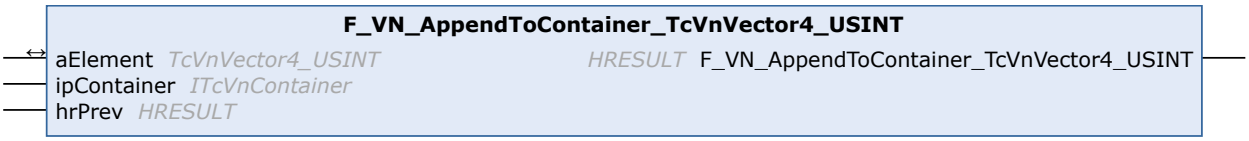
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.33 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

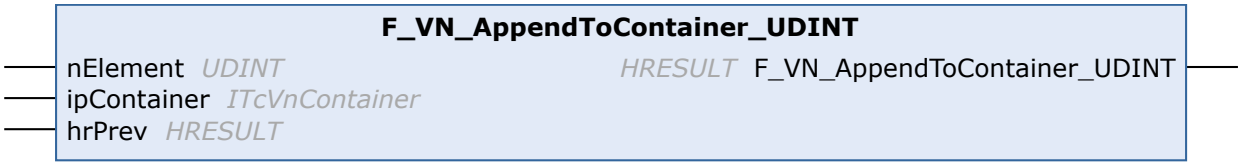
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.34 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

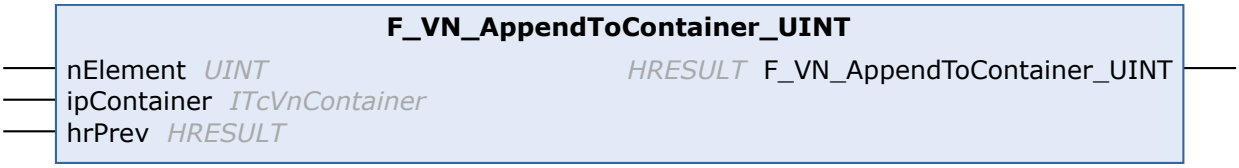
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.35 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

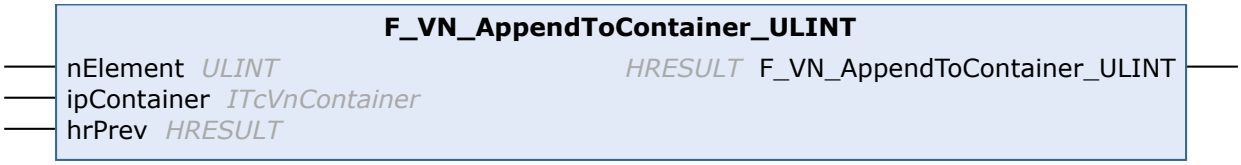
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.36 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

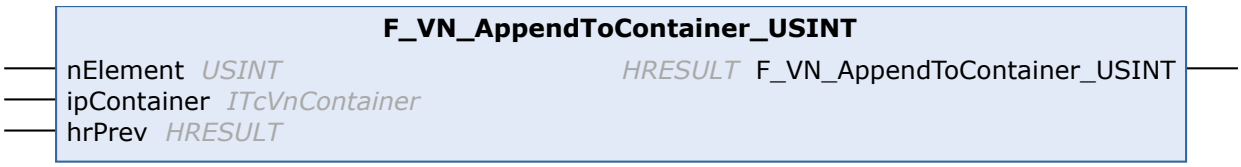
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.37 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

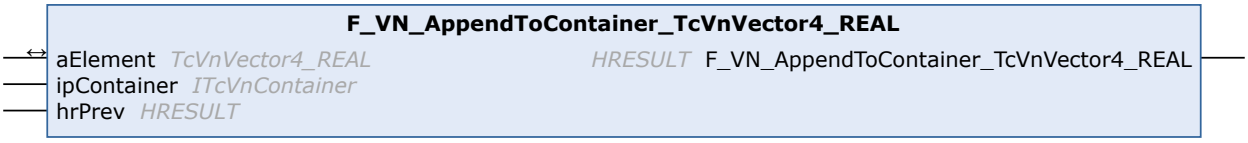
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.38 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 [▶ 370]	Container to which the element will be appended
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 162]	Single element to append to ipContainer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2 F_VN_FillContainer

This chapter contains functions to fill a container with a special value according to the data type. This function is available for containers with basic elements.

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 [▶ 370]	Container with basic elements
nValue	DINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

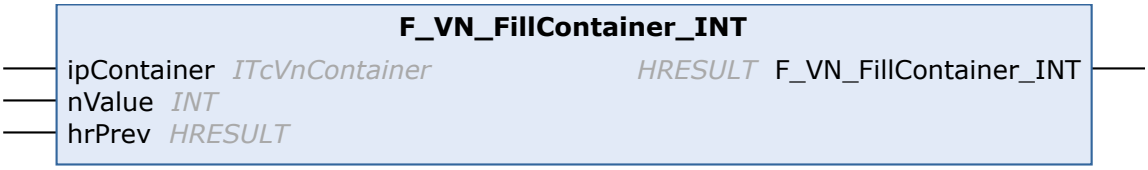
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nValue	INT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

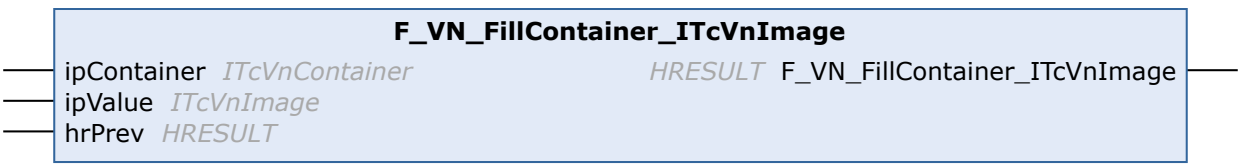
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
ipValue	ITcVnImage [▶ 411]	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

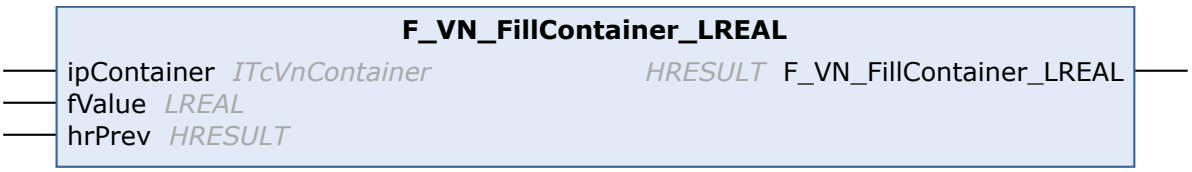
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
fValue	LREAL	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
fValue	REAL	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

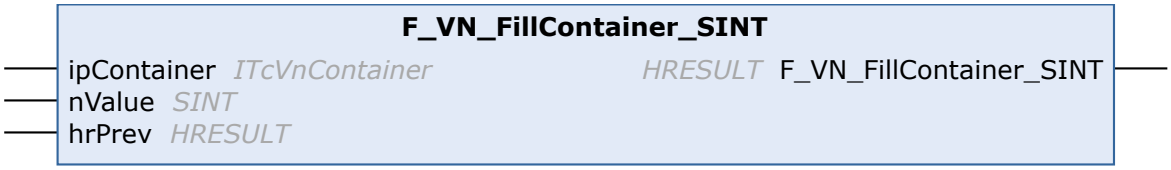
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nValue	SINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

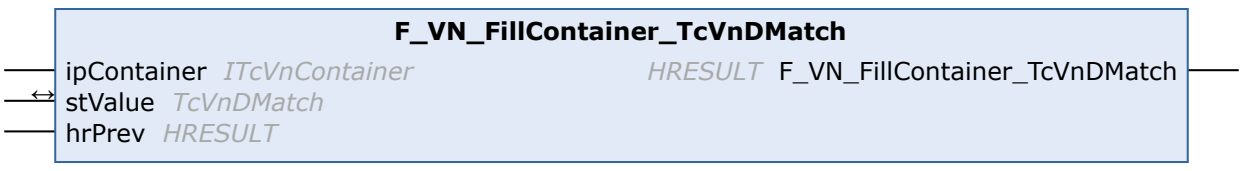
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnDMatch [▶ 230]	Value to set the container elements

Return value

HRESULT [▶ 141]

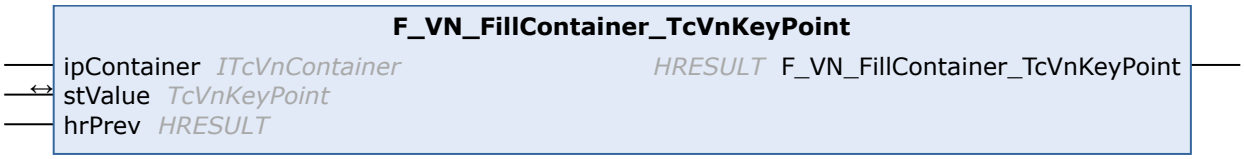
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 231]	Value to set the container elements

Return value

HRESULT [▶ 141]

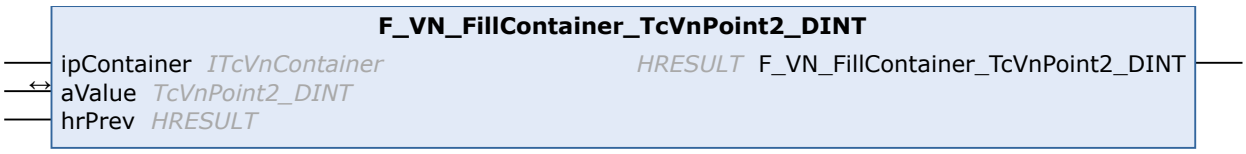
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

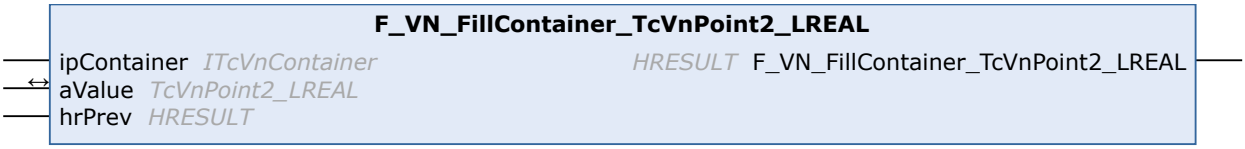
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

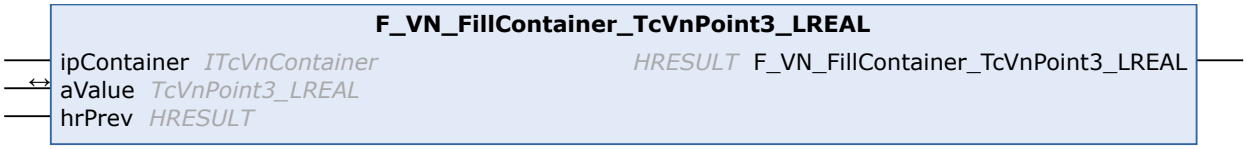
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

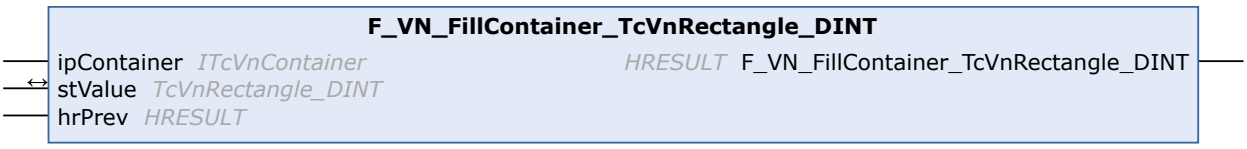
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 141]

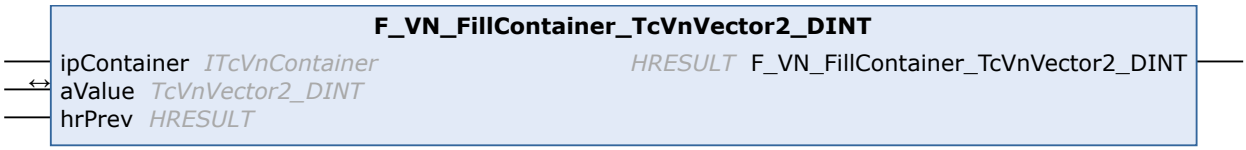
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.15 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.16 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

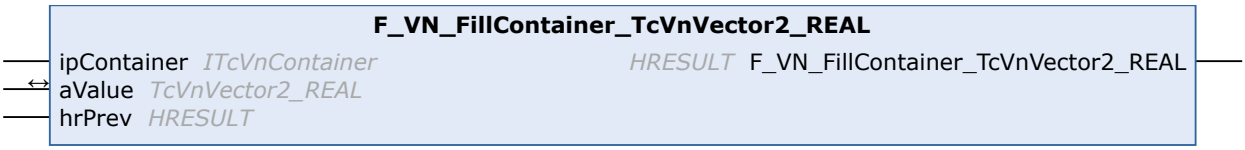
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.17 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

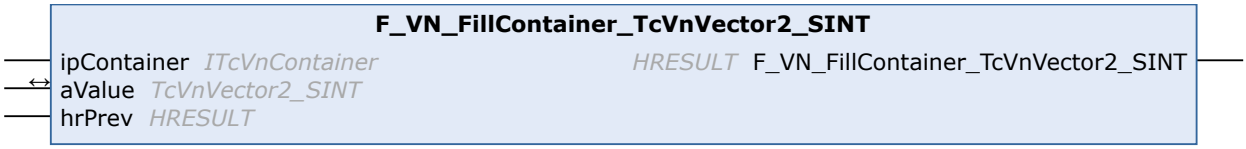
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.18 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

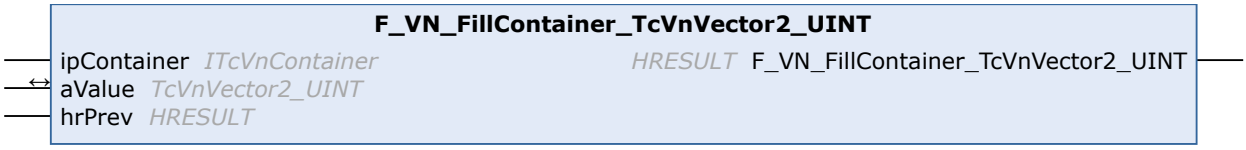
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.19 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

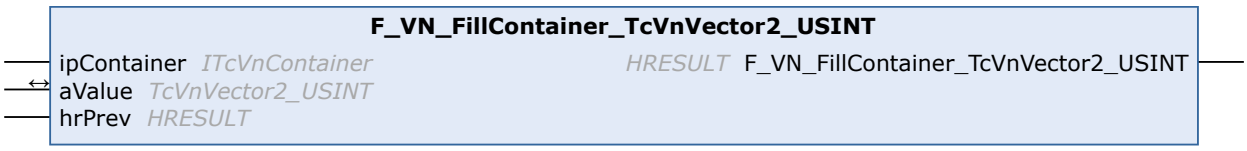
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.20 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

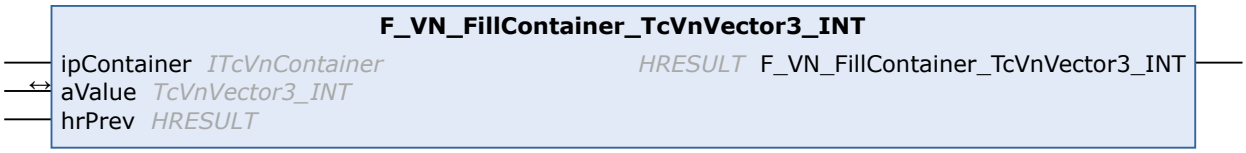
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.21 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

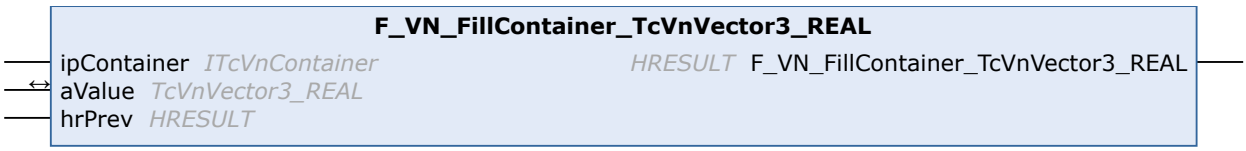
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.22 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

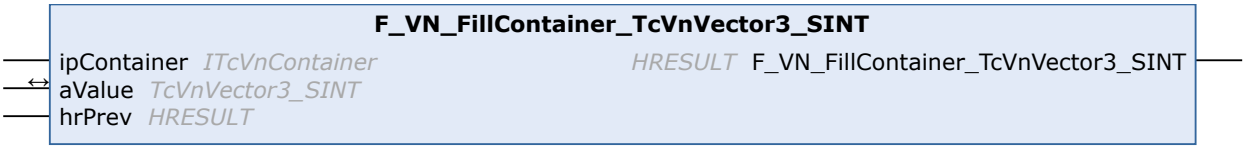
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.23 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

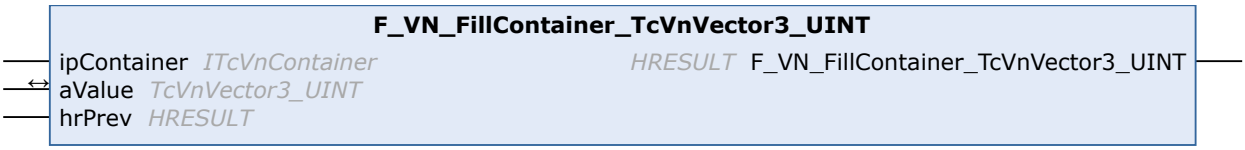
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.24 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

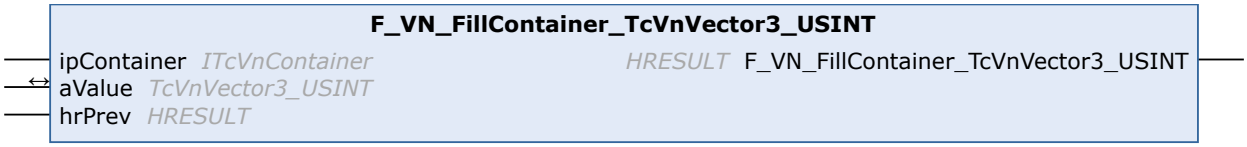
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.25 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

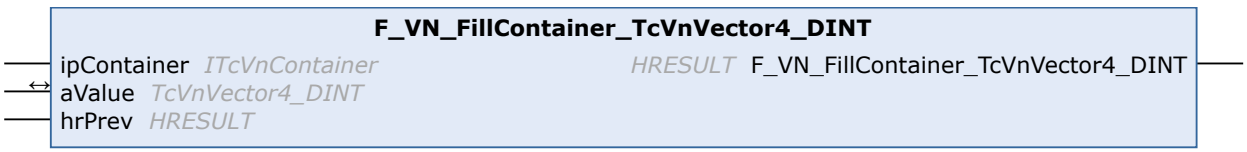
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.26 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.27 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

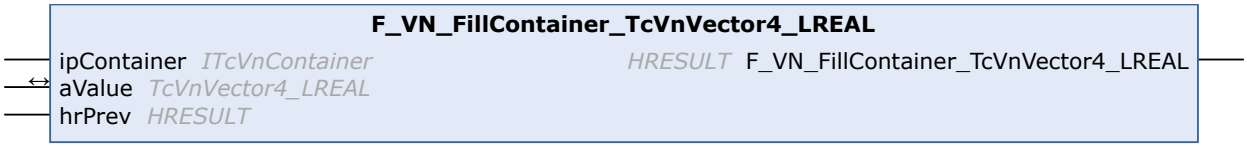
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.28 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

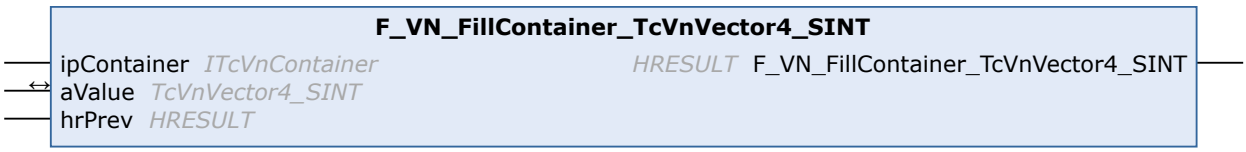
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.29 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

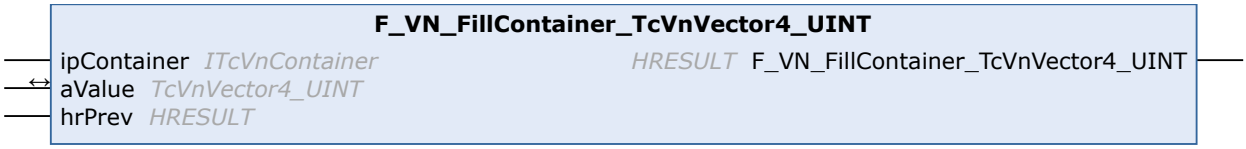
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.30 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

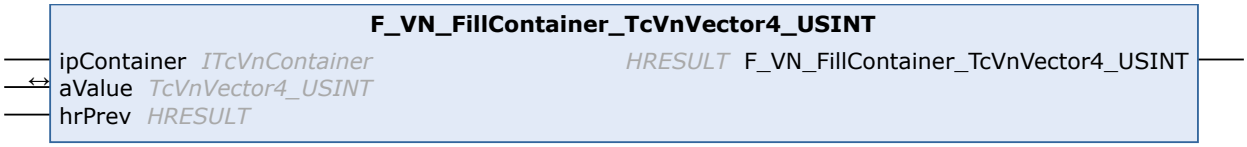
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.31 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

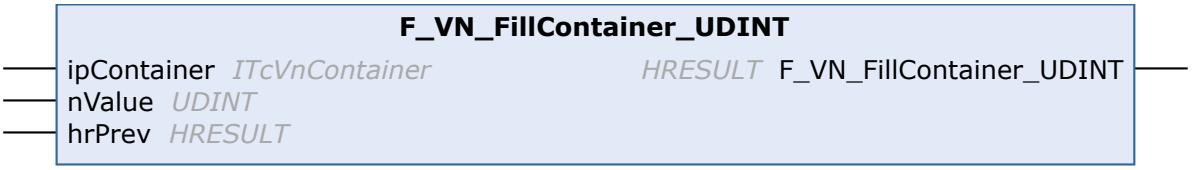
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.32 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 [▶ 370]	Container with basic elements
nValue	UDINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.33 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 [▶ 370]	Container with basic elements
nValue	UINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

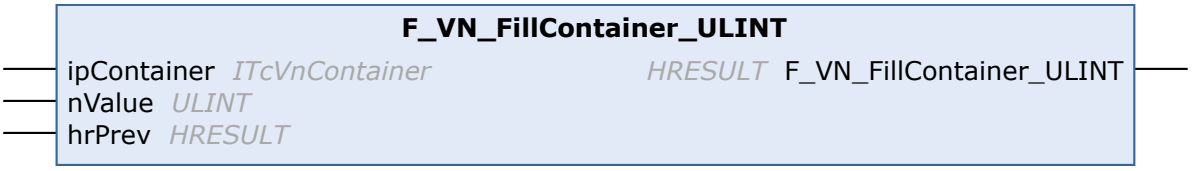
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.34 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 [▶ 370]	Container with basic elements
nValue	ULINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

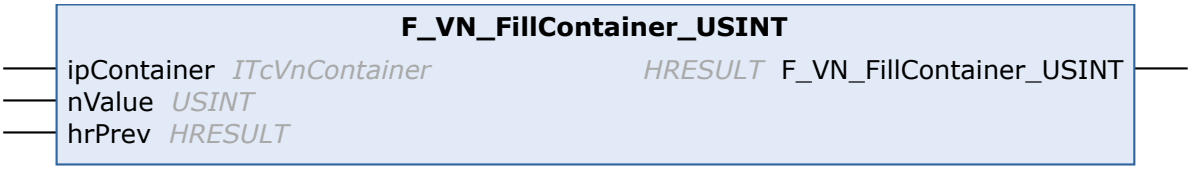
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.35 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 [▶ 370]	Container with basic elements
nValue	USINT	Value to set the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.36 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3 F_VN_FillContainerExp

This chapter contains expert functions that can be used to fill a container with a specific value, even only partially, according to the data type. This function is available for containers with basic elements.

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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
ipValue	ITcVnImage [▶ 411]	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnDMatch [▶ 230]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 231]	Value to set the container elements

Return value

HRESULT [▶ 141]

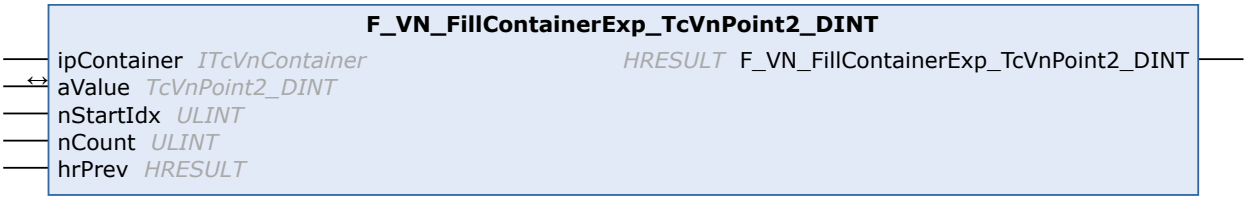
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

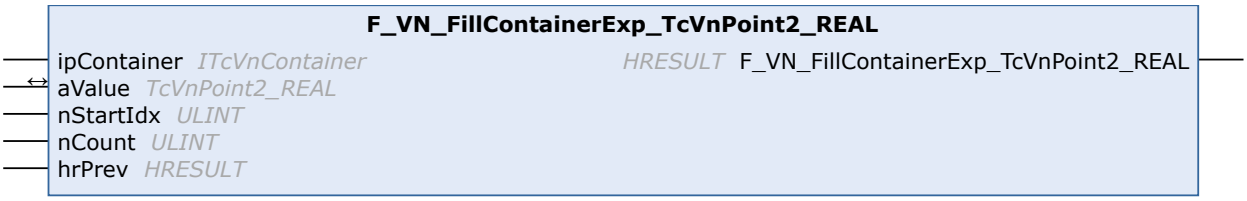
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

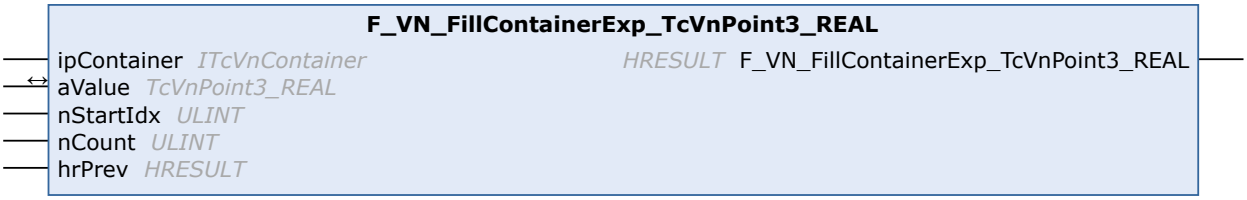
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 160]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.15 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

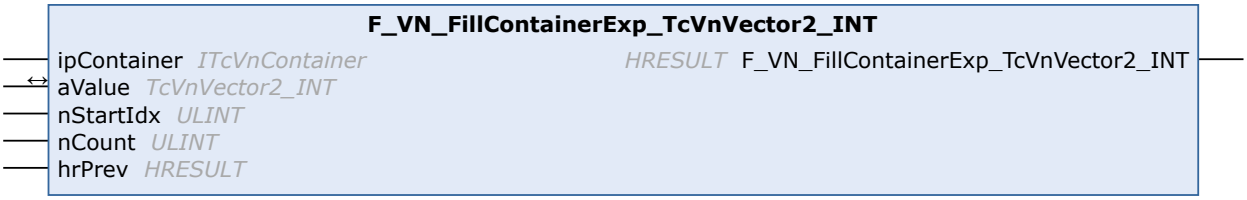
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.16 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.17 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.18 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.19 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.20 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

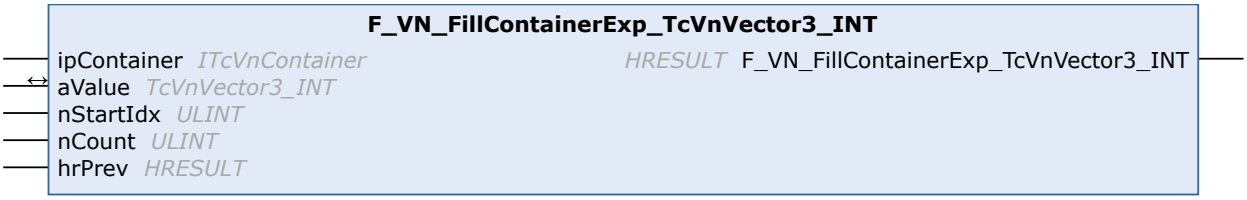
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.21 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.22 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.23 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.24 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.25 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.26 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

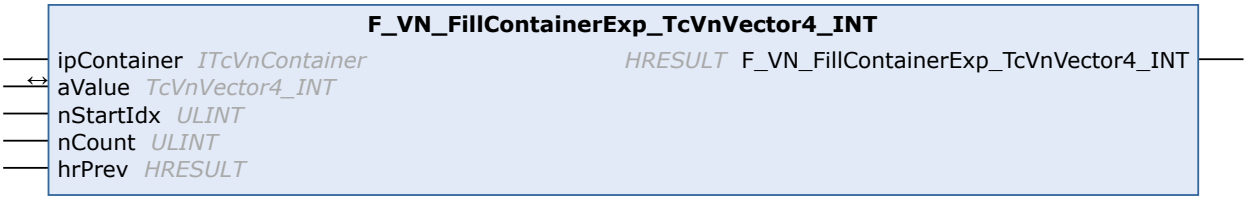
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.27 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

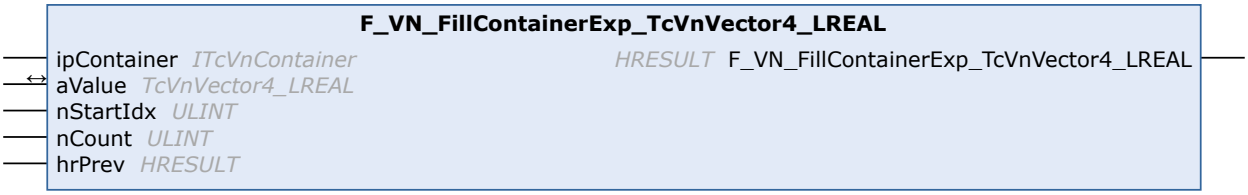
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.28 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.29 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.30 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.31 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.32 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.33 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.34 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.35 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.36 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 [▶ 370]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 162]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

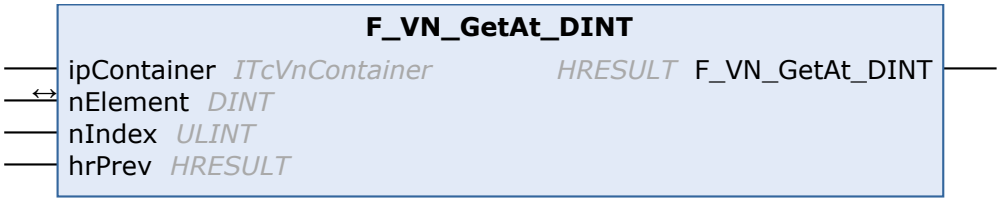
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4 F_VN_GetAt

The `F_VN_GetAt` functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [▶ 766].

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 [▶ 370]	Container with DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

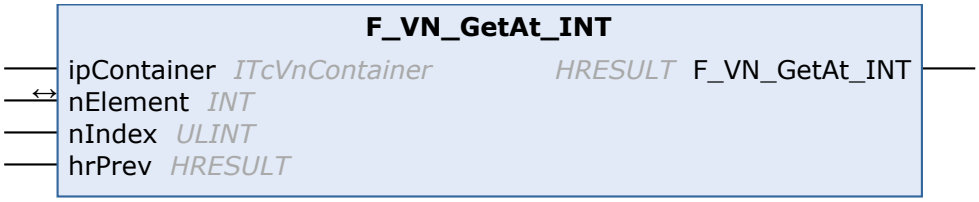
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with ITcVnContainer elements
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the container at the specified index
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with ITcVnImage elements
ipElement	Reference To ITcVnImage [▶ 411]	Returns the element at the specified index
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

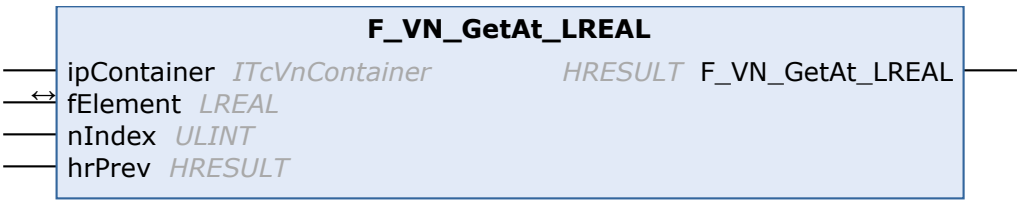
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

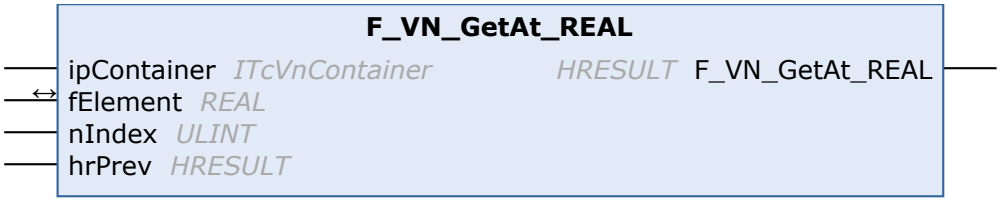
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

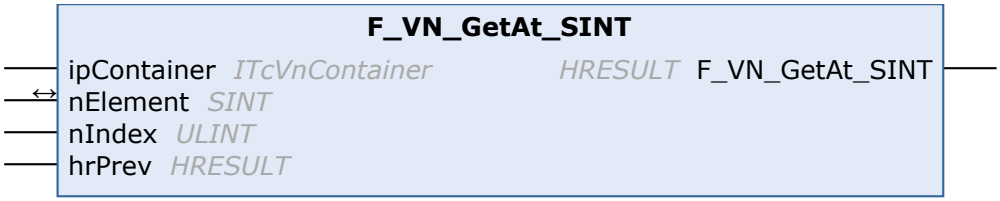
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

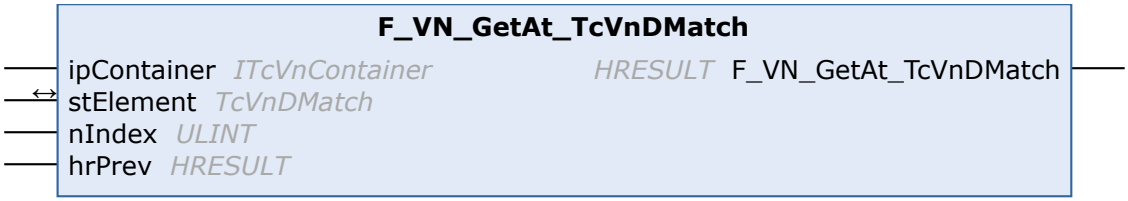
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnDMatch elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 230]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnKeyPoint elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 231]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 160]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

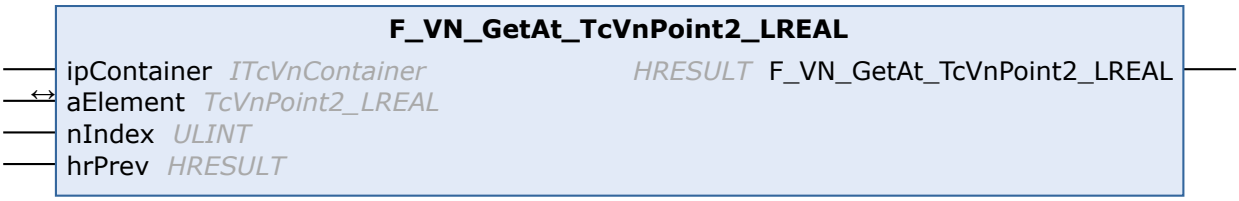
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 160]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 160]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

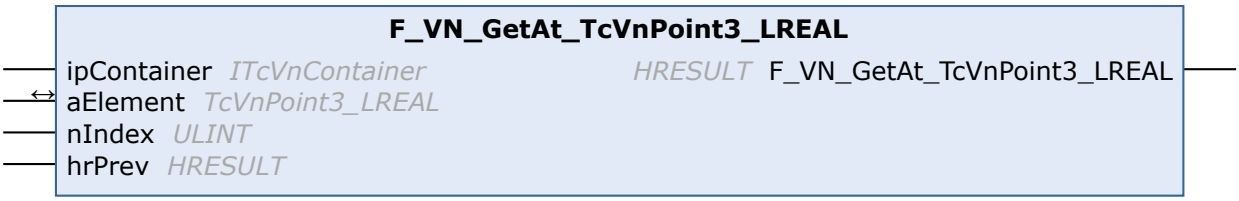
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 160]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 160]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

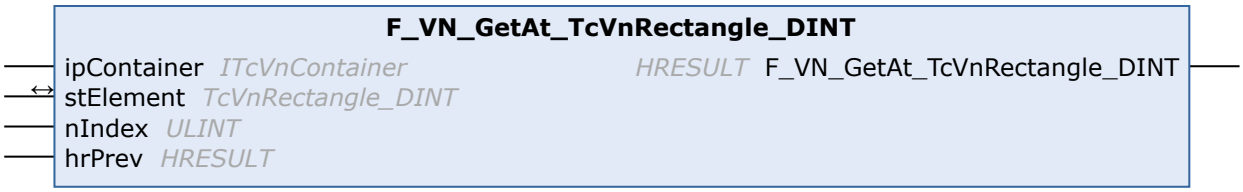
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnRectangle_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 244]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.16 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 [▶ 370]	Container with TcVnVector2_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.17 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 [▶ 370]	Container with TcVnVector2_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

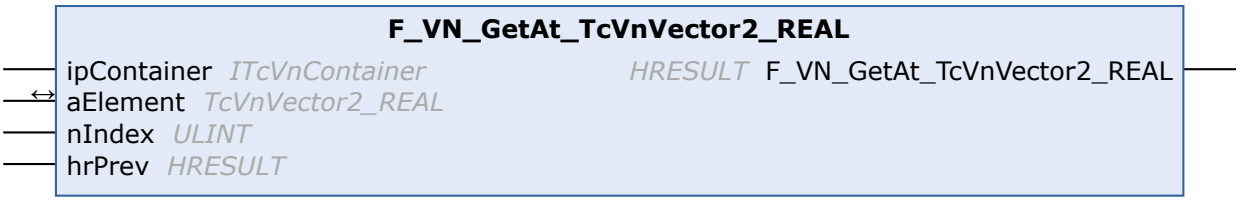
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.18 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 [▶ 370]	Container with TcVnVector2_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

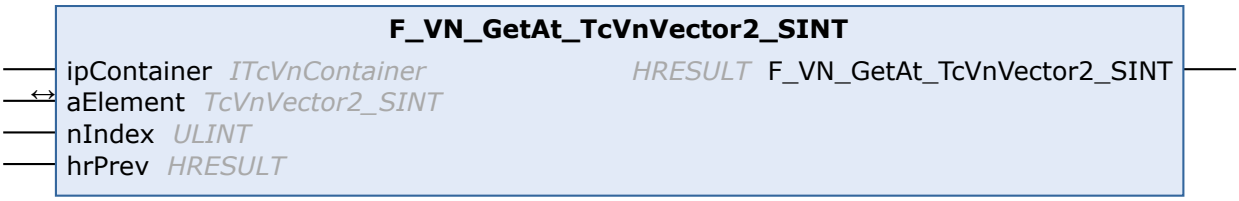
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.19 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 [▶ 370]	Container with TcVnVector2_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

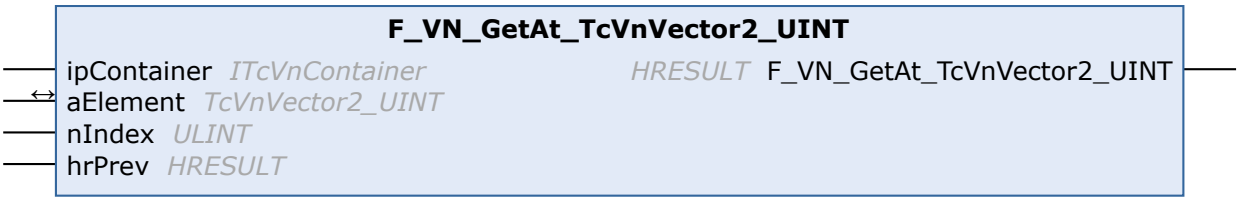
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.20 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 [▶ 370]	Container with TcVnVector2_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

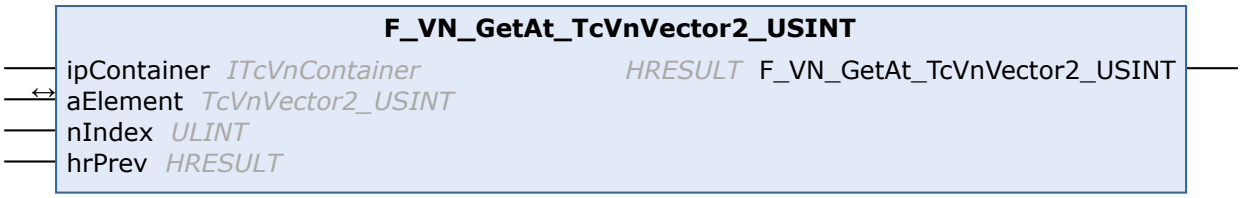
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.21 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 [▶ 370]	Container with TcVnVector2_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.22 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 [▶ 370]	Container with TcVnVector3_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

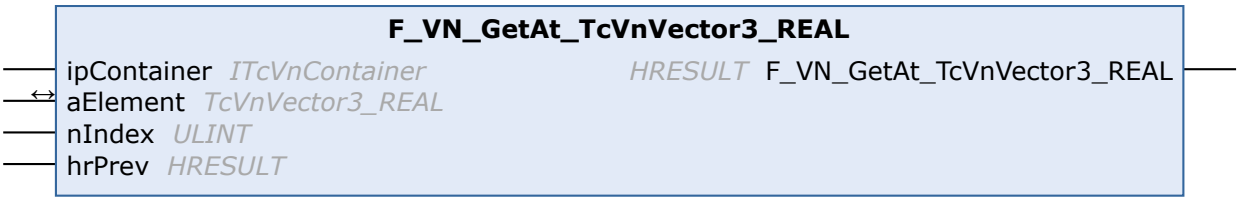
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.23 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 [▶ 370]	Container with TcVnVector3_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

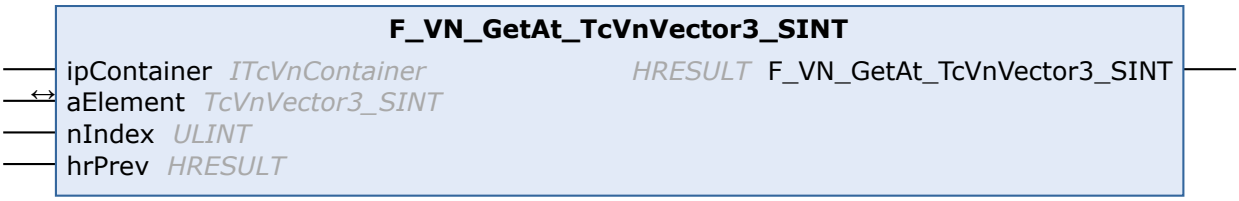
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.24 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 [▶ 370]	Container with TcVnVector3_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

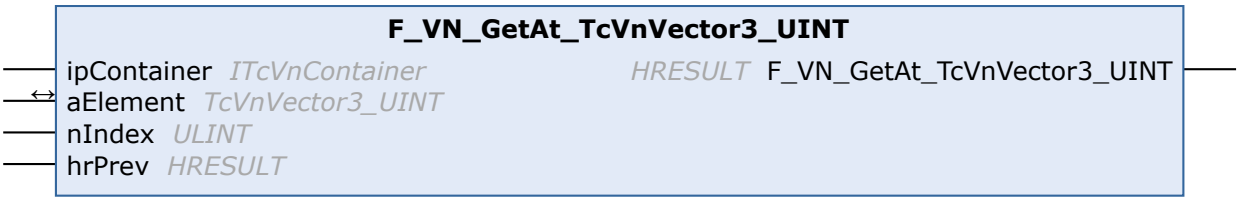
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.25 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 [▶ 370]	Container with TcVnVector3_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

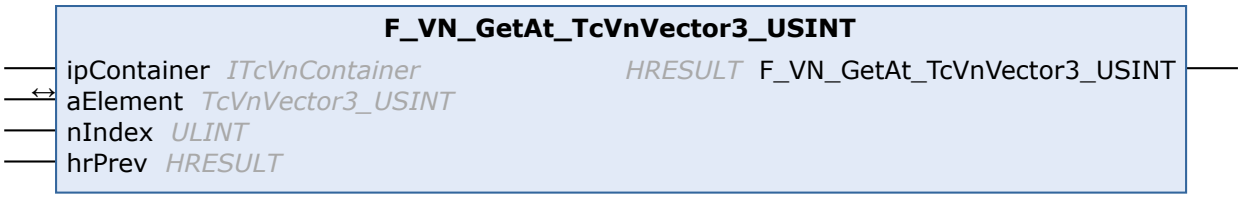
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.26 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 [▶ 370]	Container with TcVnVector3_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.27 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 [▶ 370]	Container with TcVnVector4_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.28 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 [▶ 370]	Container with TcVnVector4_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

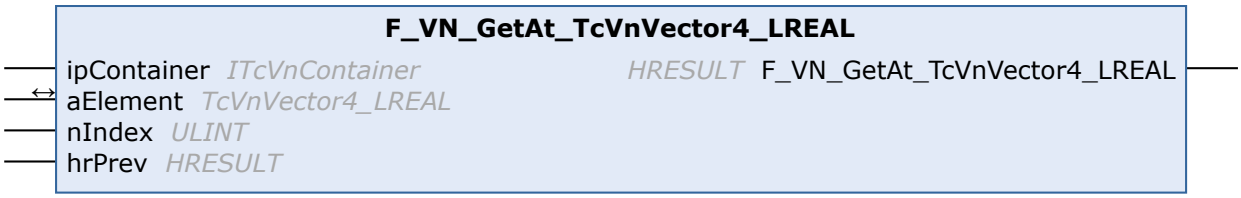
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.29 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 [▶ 370]	Container with TcVnVector4_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

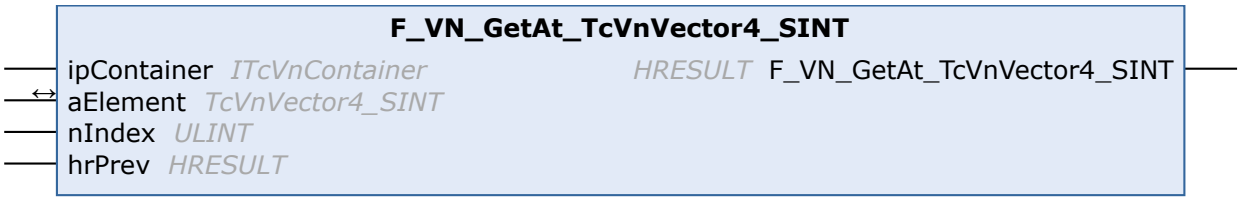
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.30 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 [▶ 370]	Container with TcVnVector4_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

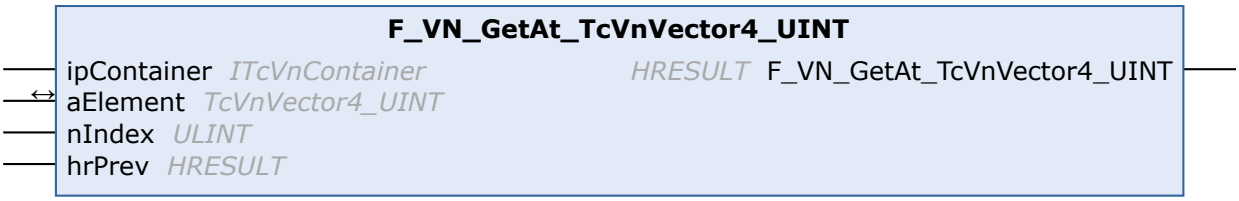
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.31 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 [▶ 370]	Container with TcVnVector4_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

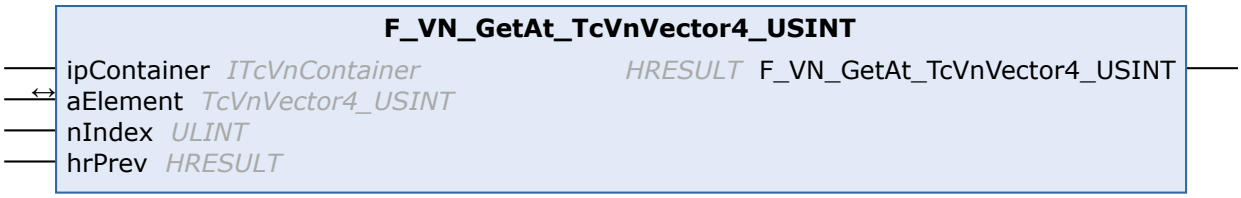
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.32 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 [▶ 370]	Container with TcVnVector4_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.33 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 [▶ 370]	Container with UDINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

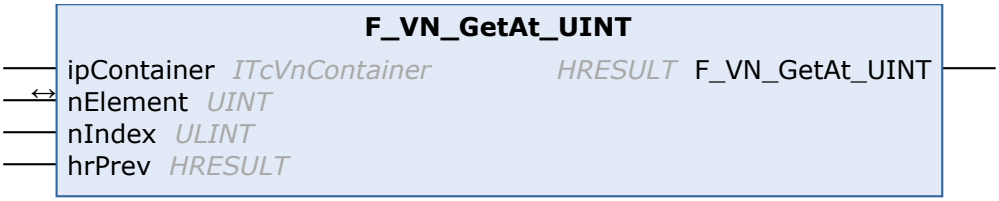
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.34 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 [▶ 370]	Container with UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

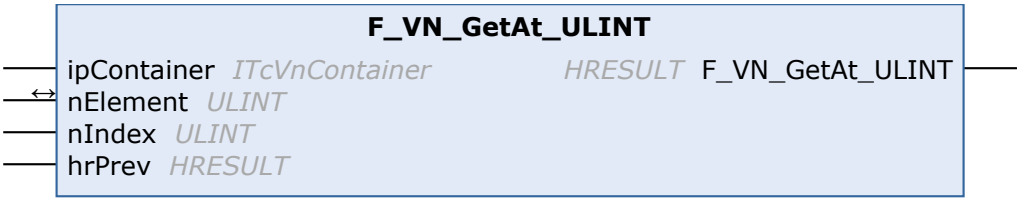
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.35 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 [▶ 370]	Container with ULINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

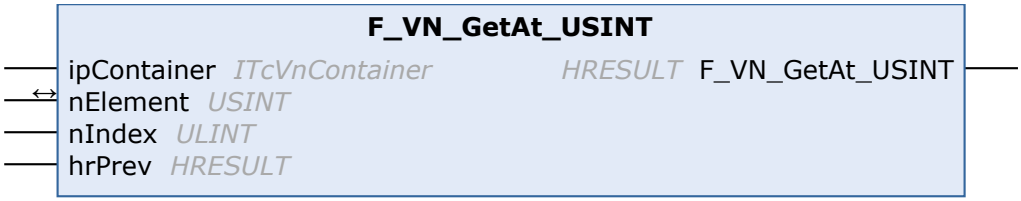
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.36 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 [▶ 370]	Container with USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

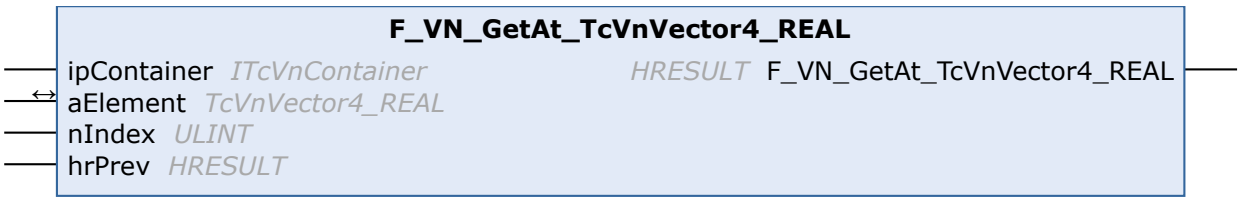
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.37 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 [▶ 370]	Container with TcVnVector4_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 162]	Returns the element at the specified index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

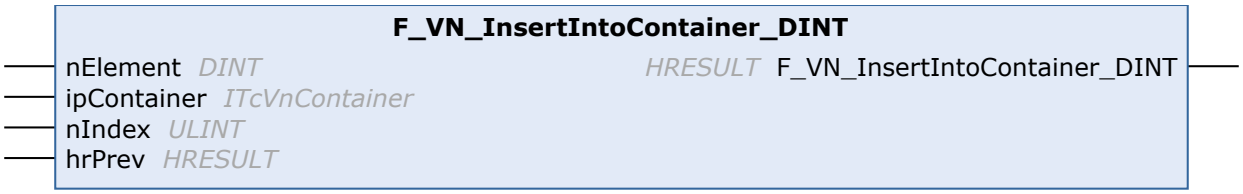
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5 F_VN_InsertIntoContainer

This chapter contains functions to insert an element at a specific container position according to the data type.

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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single element to insert into ipContainer or container with several elements
ipContainer	ITcVnContainer [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

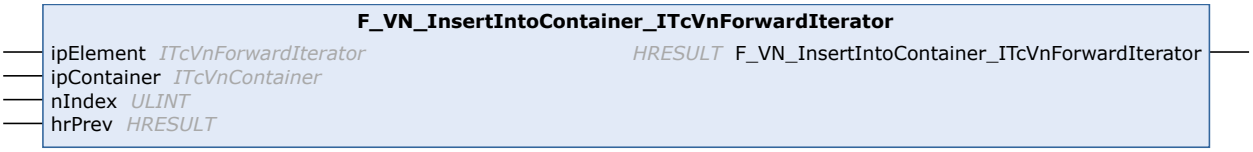
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

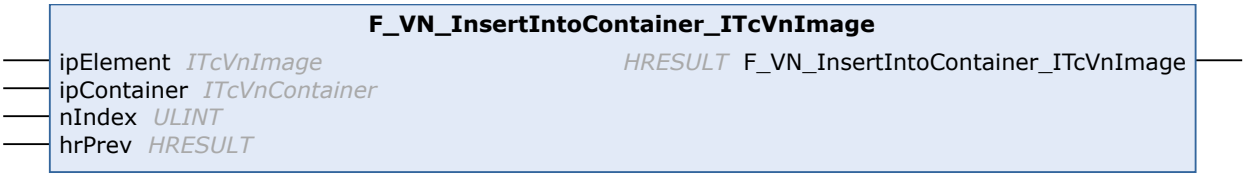
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

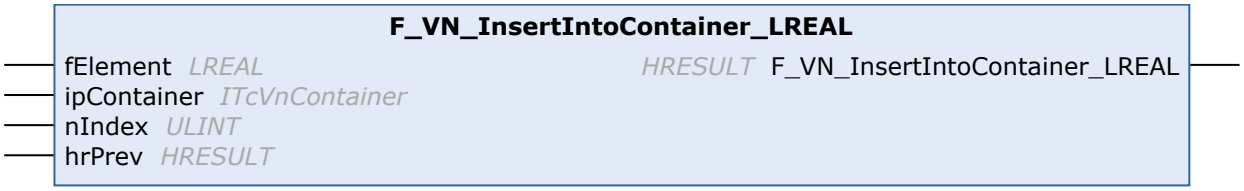
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

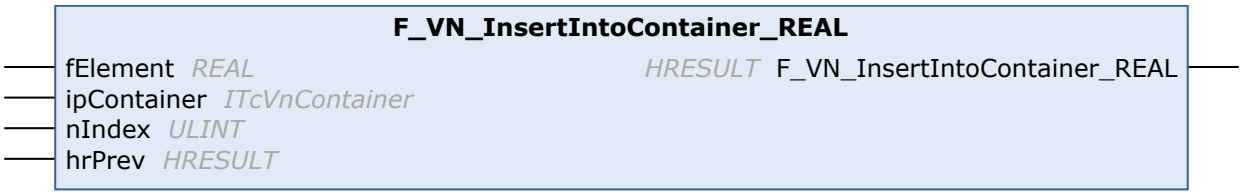
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

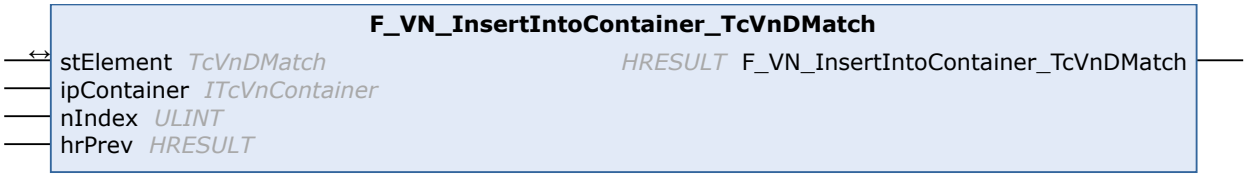
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 230]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

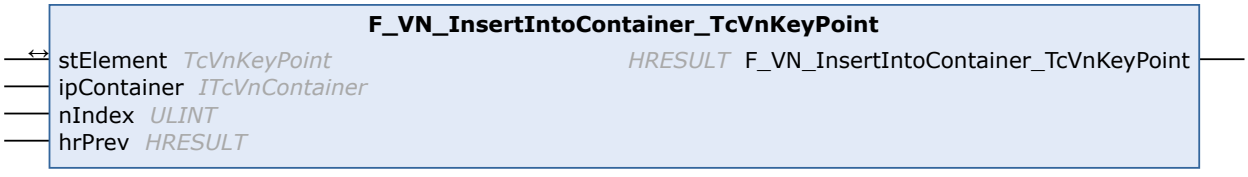
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 231]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

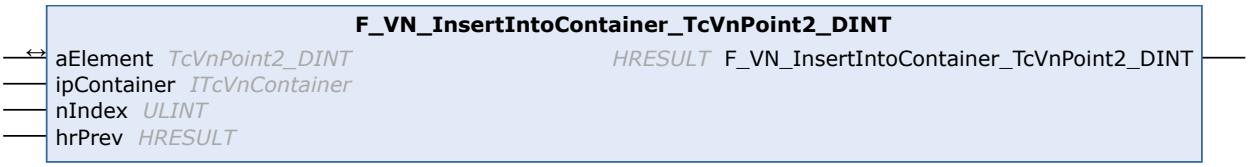
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 160]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

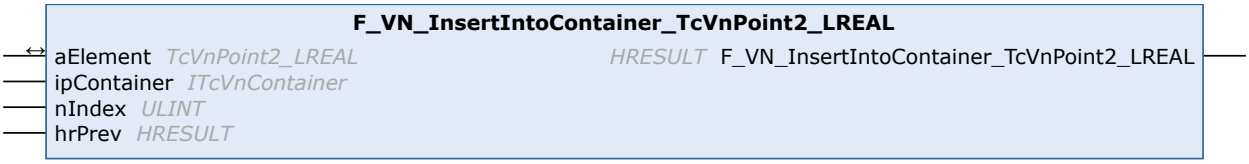
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 160]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

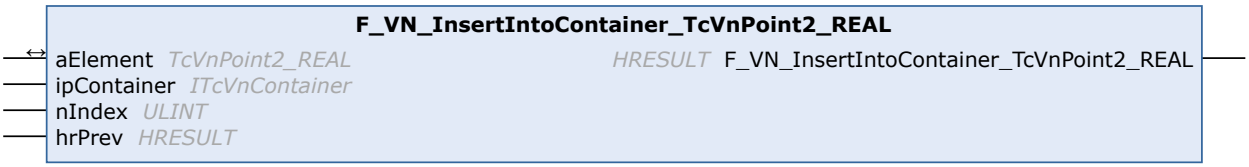
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 160]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

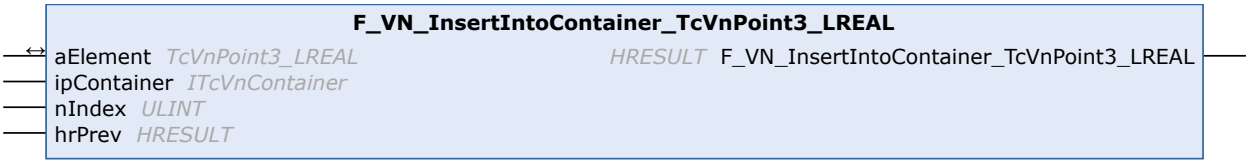
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 160]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

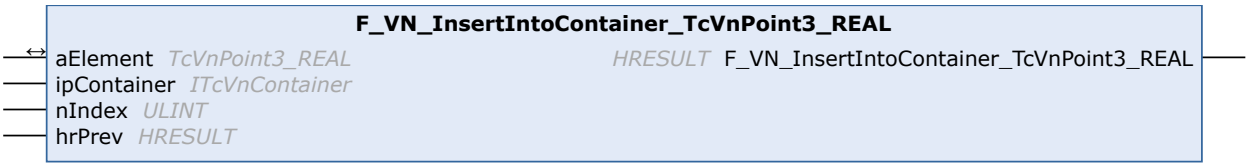
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 160]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

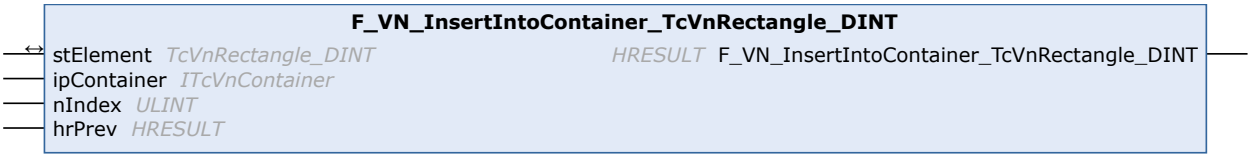
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 244]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

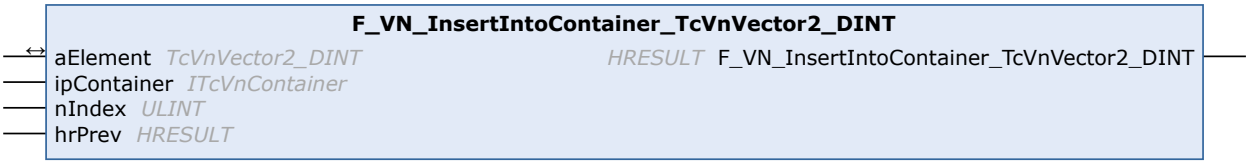
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

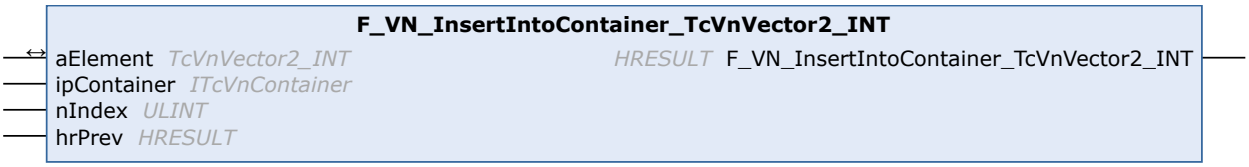
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

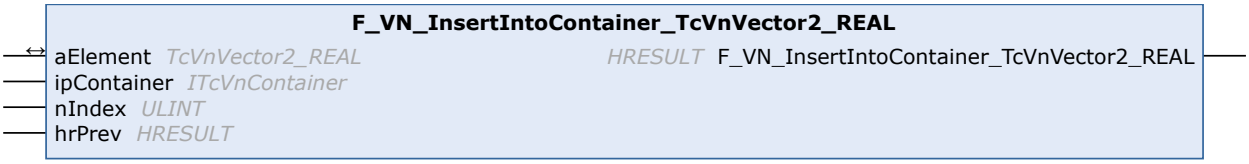
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

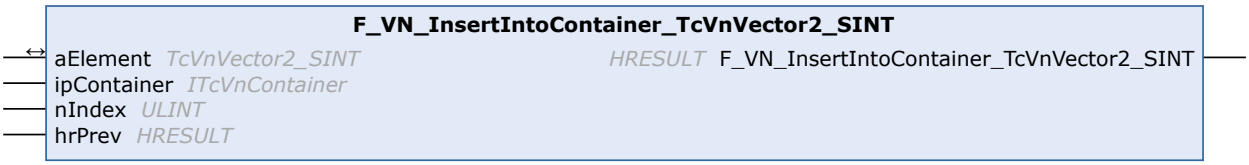
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

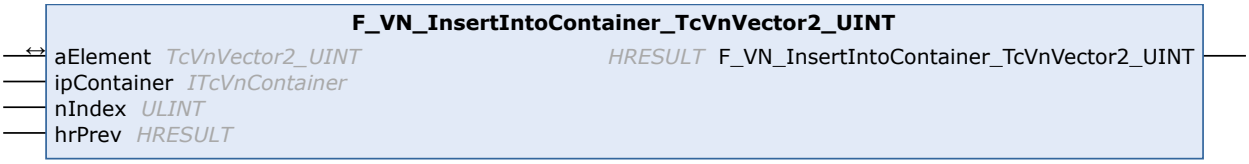
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

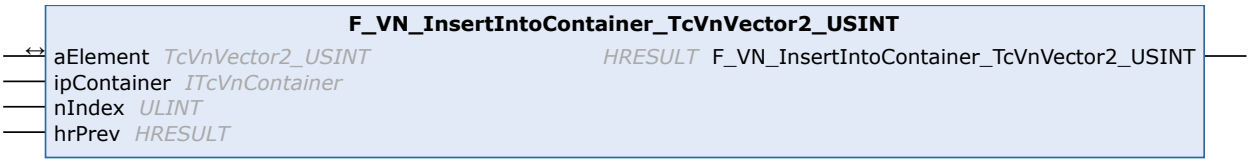
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

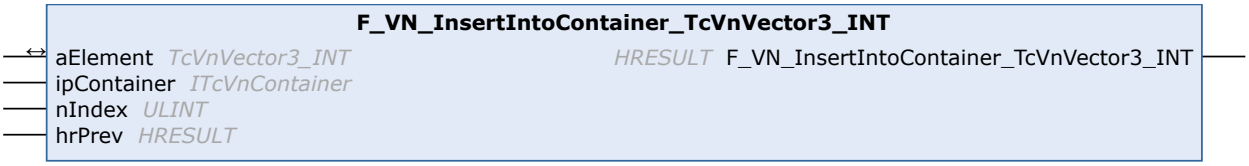
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

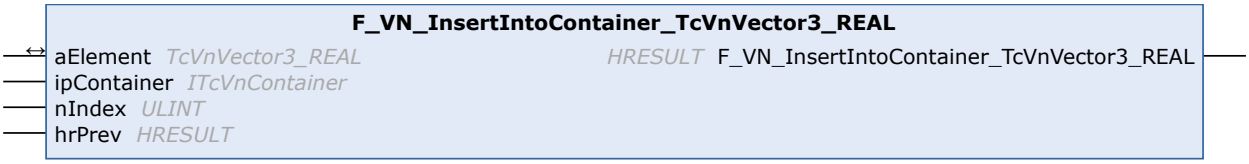
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

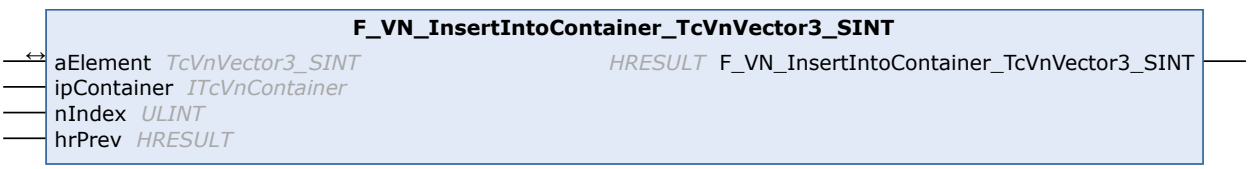
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

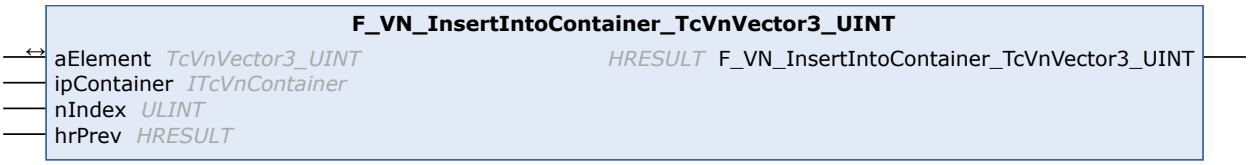
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

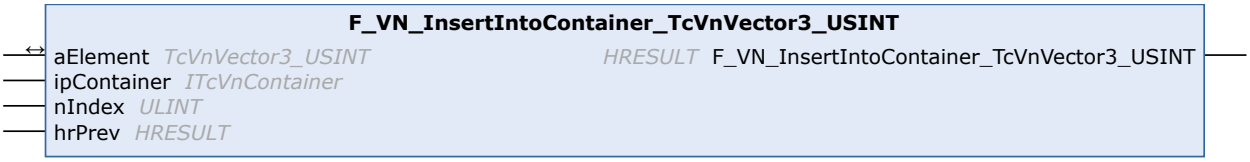
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

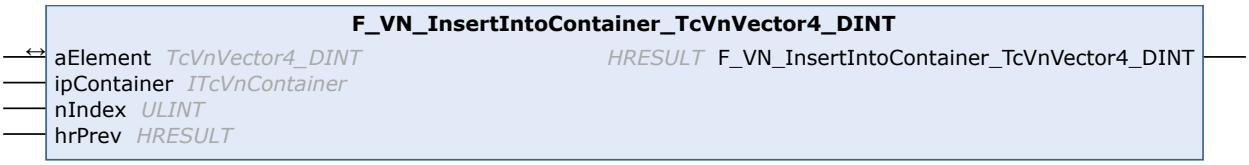
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

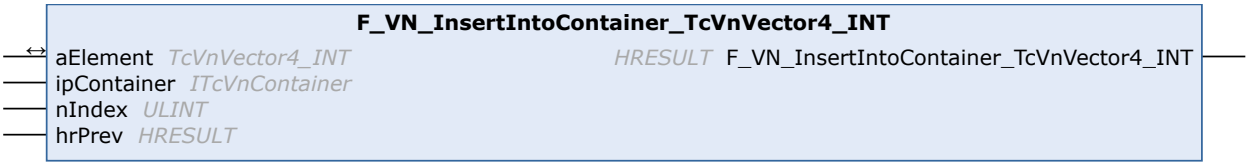
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

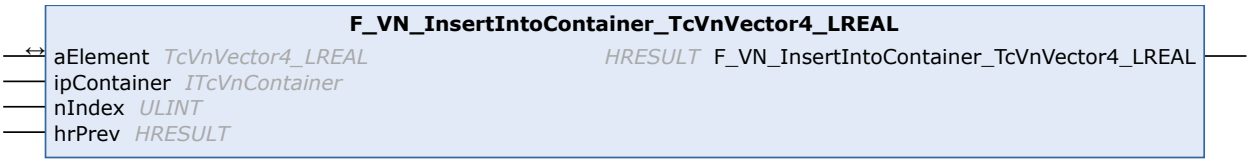
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

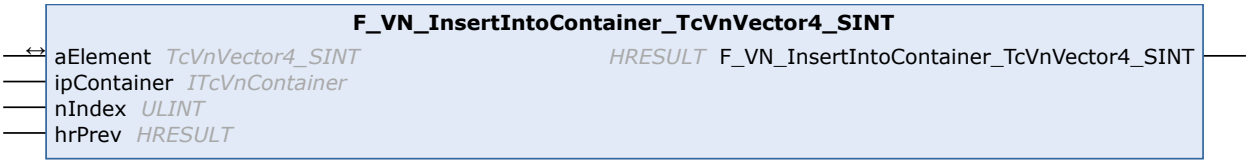
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

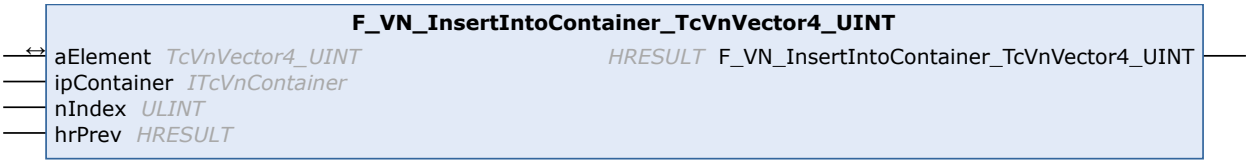
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

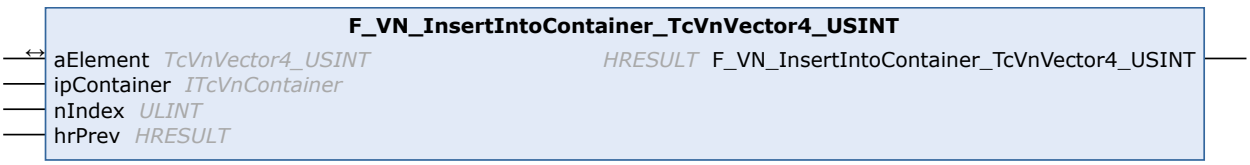
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

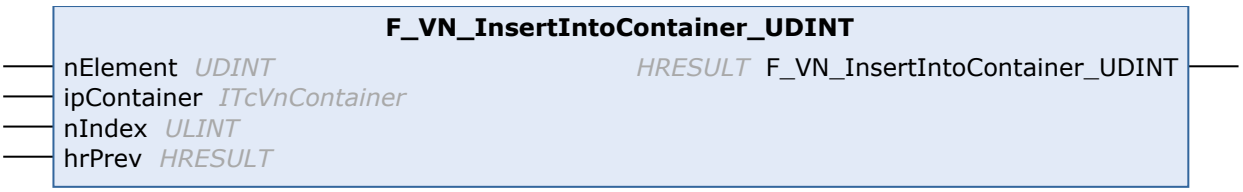
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

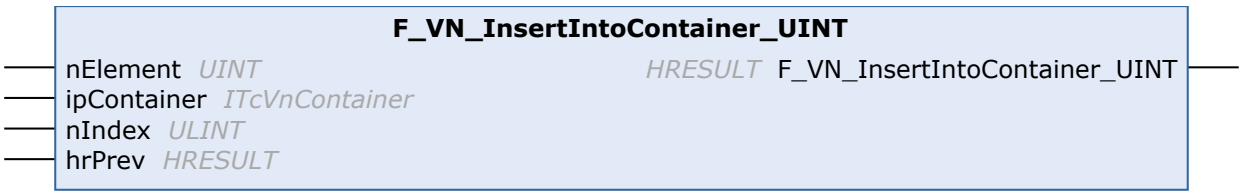
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

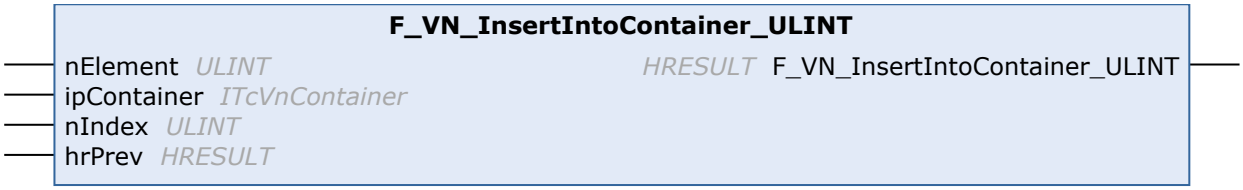
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

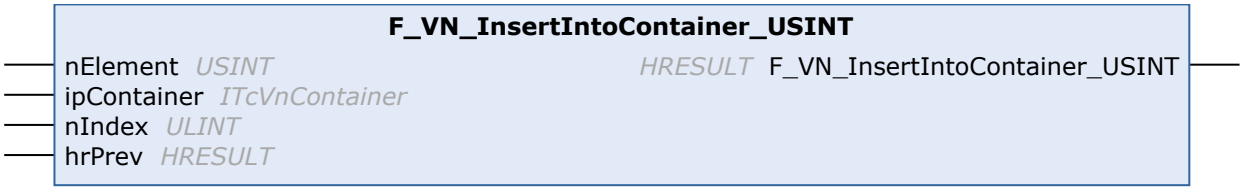
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

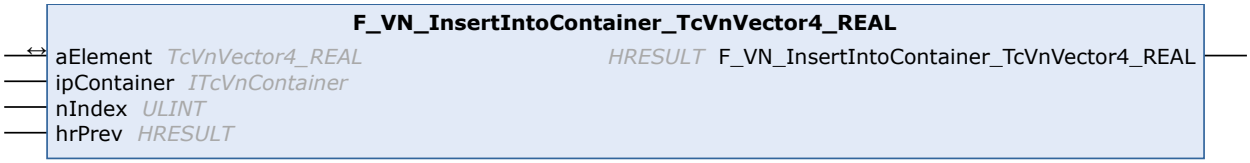
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 162]	Single element to insert into ipContainer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

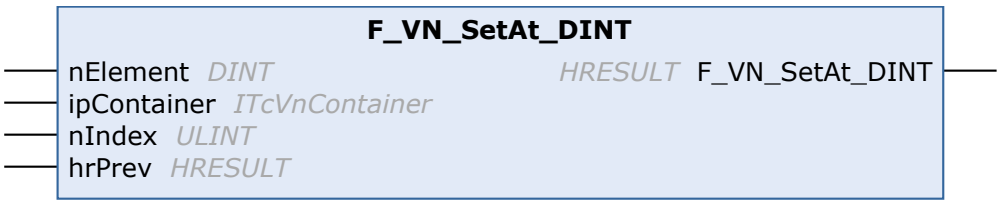
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6 F_VN_SetAt

The `F_VN_SetAt` functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [▶ 766].

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 [▶ 370]	Container with DINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

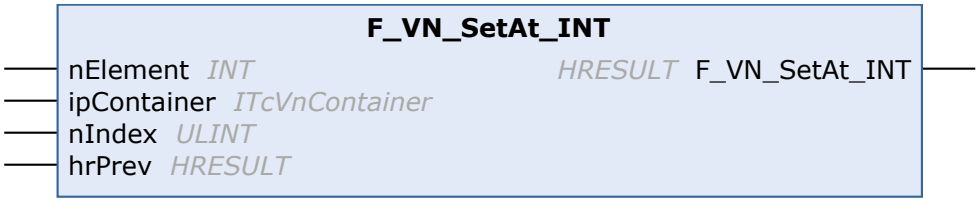
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with INT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

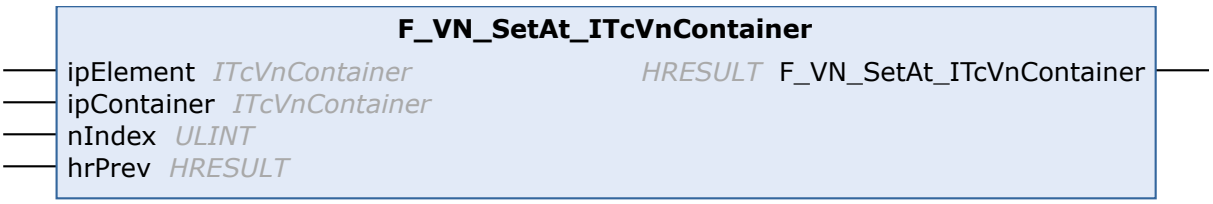
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 370]	Container with ITcVnContainer elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

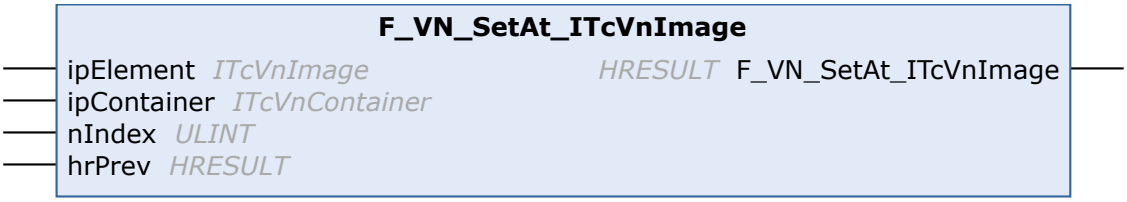
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 370]	Container with ITcVnImage elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

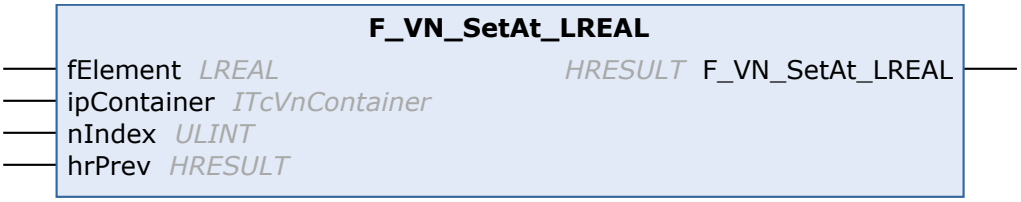
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with LREAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

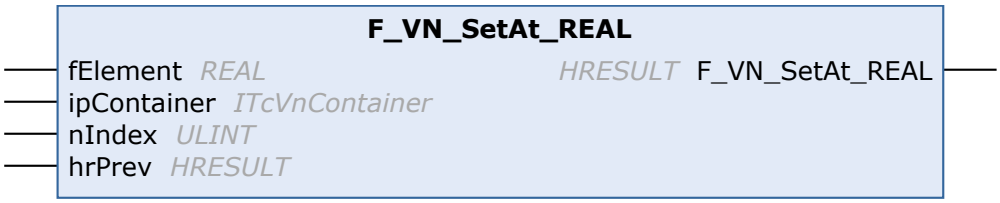
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with REAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

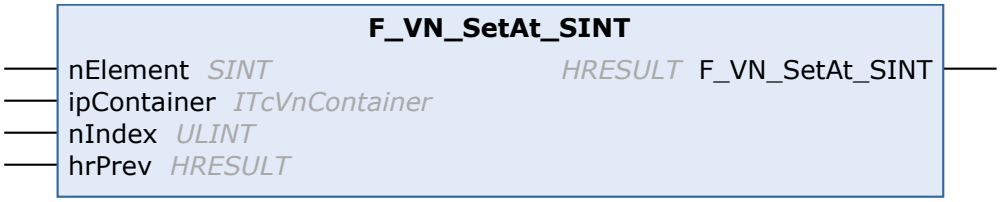
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with SINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

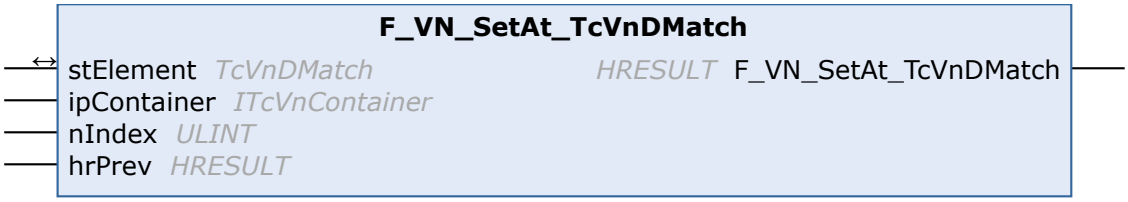
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnDMatch elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 230]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnKeyPoint elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 231]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 160]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

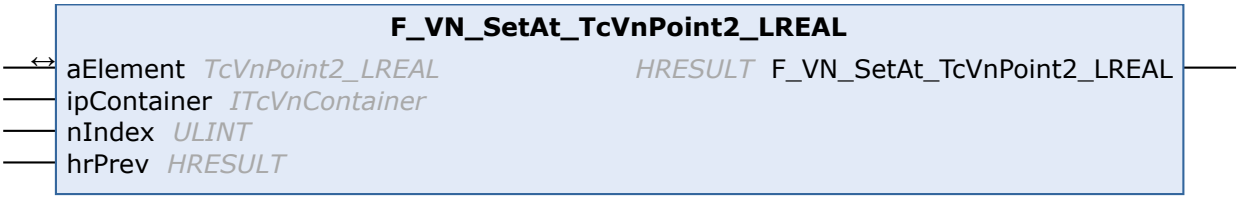
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 160]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 160]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

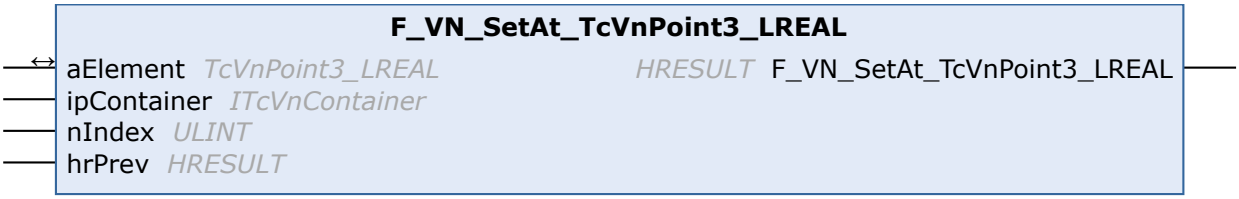
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 160]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 160]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

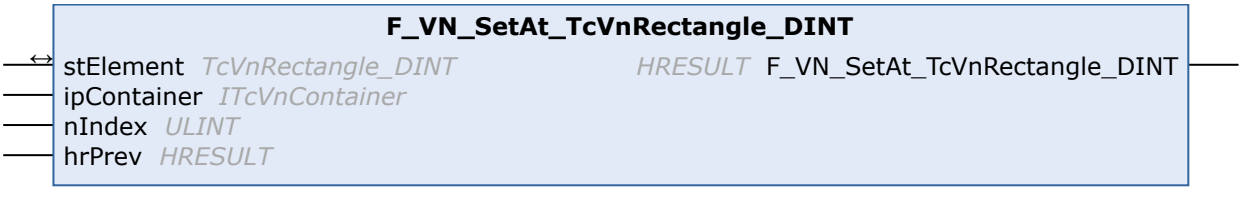
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnRectangle_DINT elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 244]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

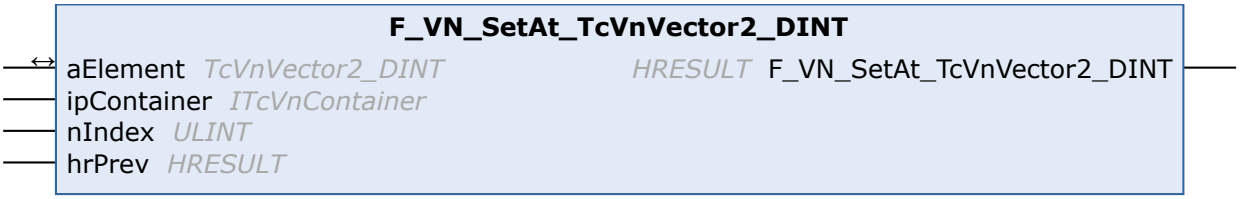
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.16 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 [▶ 370]	Container with TcVnVector2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

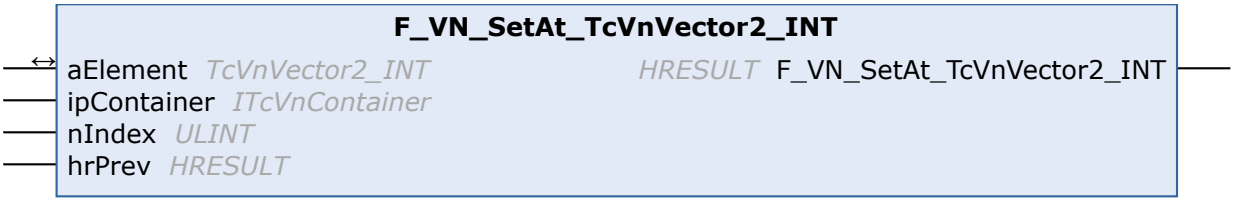
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.17 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 [▶ 370]	Container with TcVnVector2_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

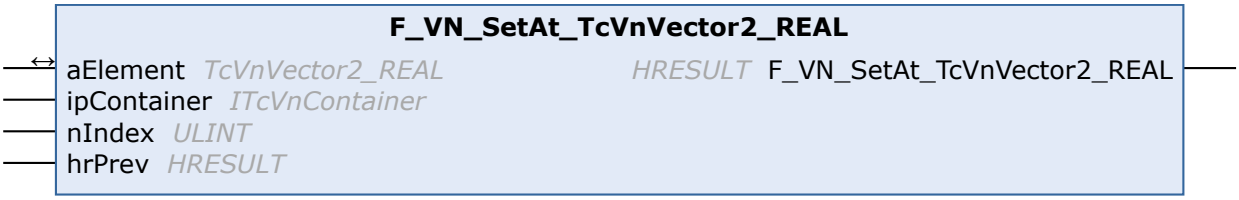
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.18 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 [▶ 370]	Container with TcVnVector2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

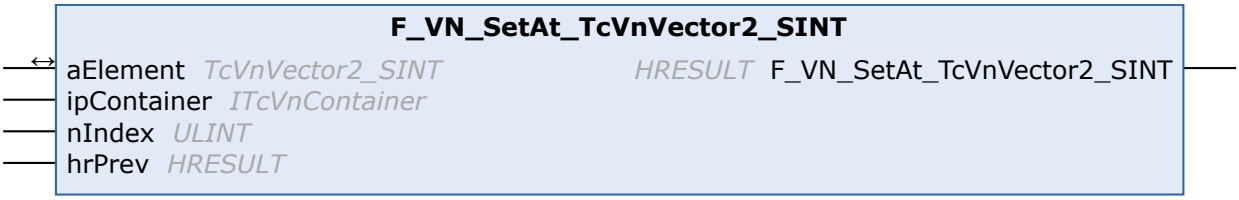
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.19 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 [▶ 370]	Container with TcVnVector2_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

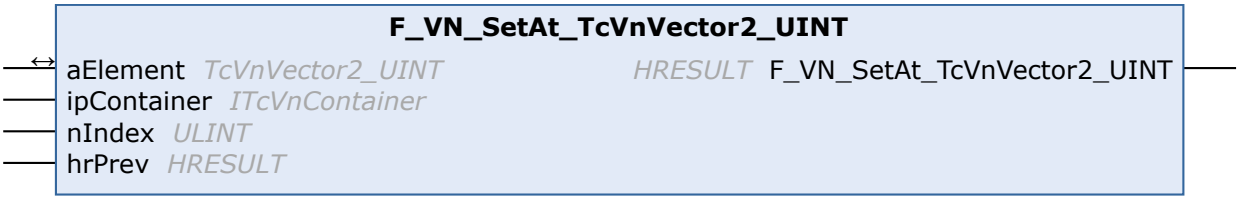
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.20 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 [▶ 370]	Container with TcVnVector2_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

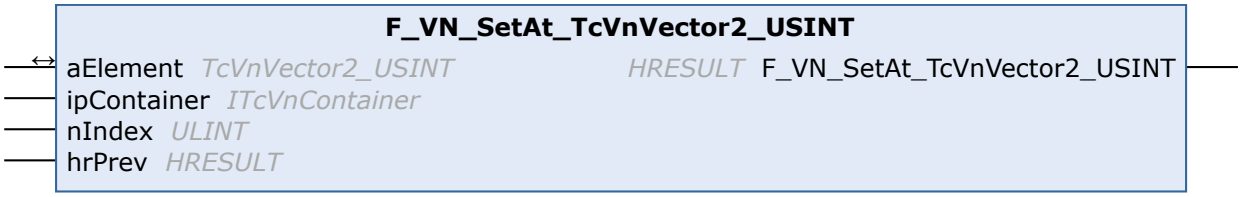
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.21 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 [▶ 370]	Container with TcVnVector2_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

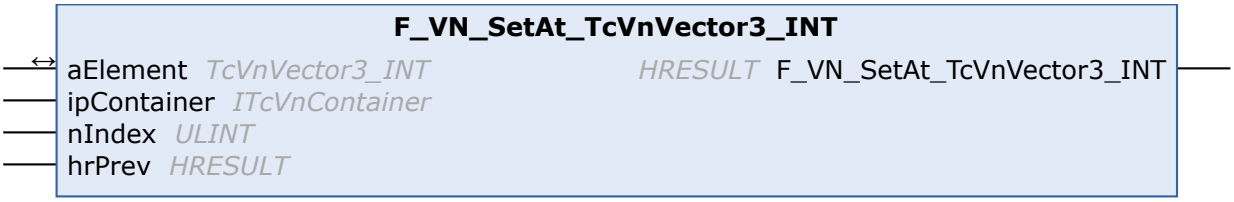
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.22 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 [▶ 370]	Container with TcVnVector3_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

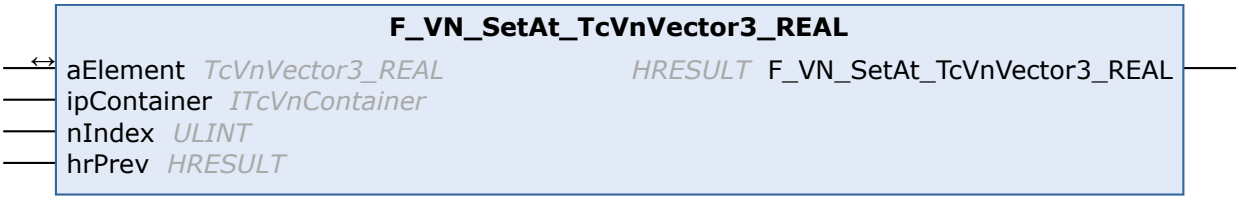
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.23 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 [▶ 370]	Container with TcVnVector3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

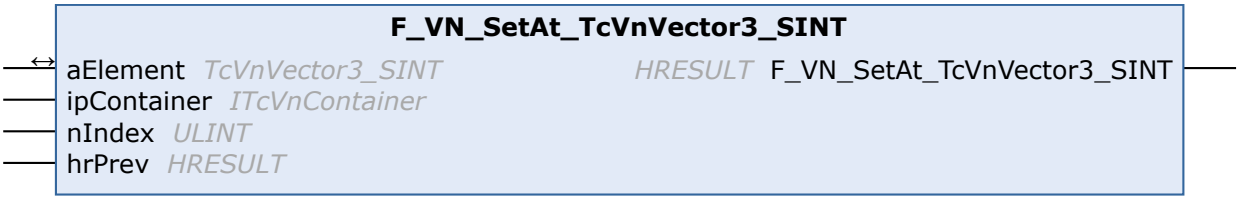
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.24 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 [▶ 370]	Container with TcVnVector3_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

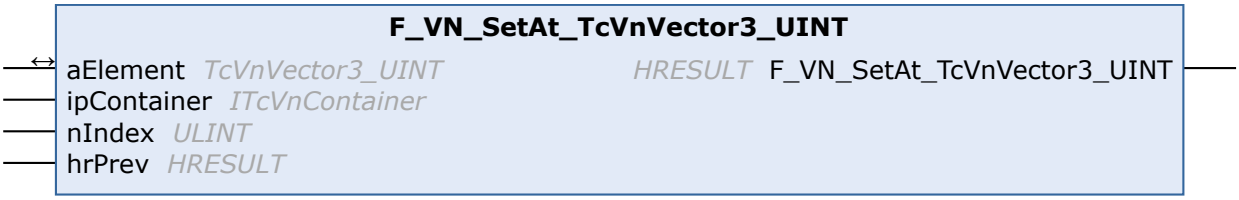
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.25 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 [▶ 370]	Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

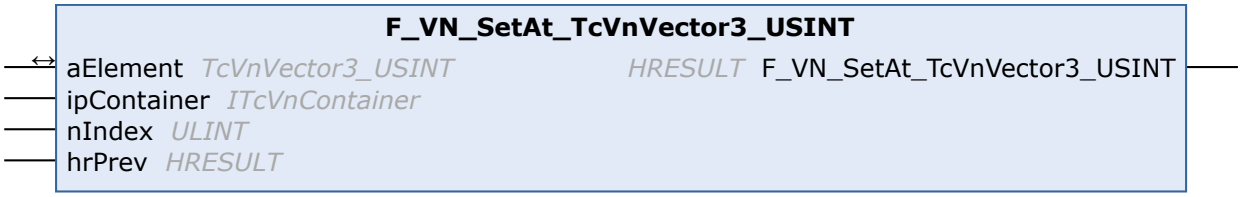
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.26 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 [▶ 370]	Container with TcVnVector3_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

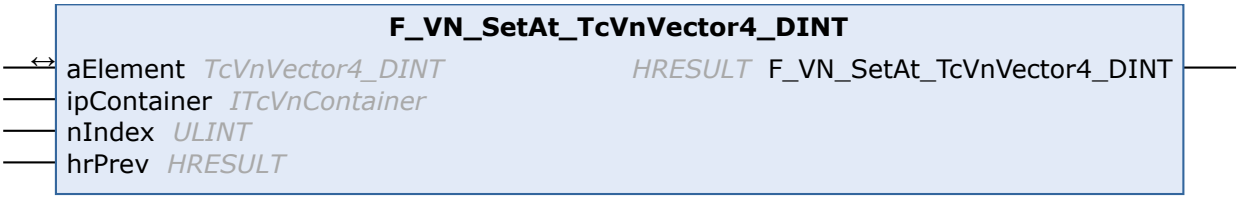
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.27 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 [▶ 370]	Container with TcVnVector4_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

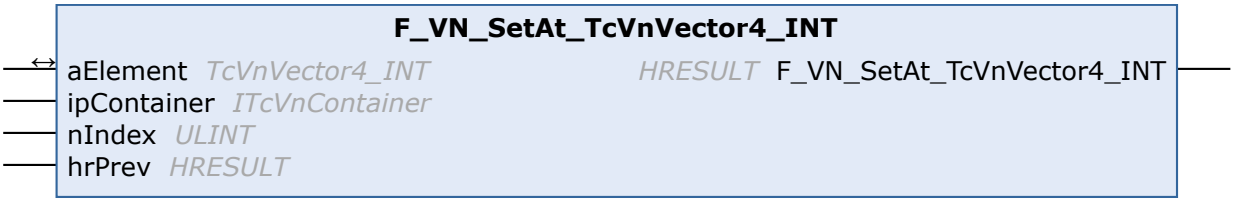
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.28 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 [▶ 370]	Container with TcVnVector4_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

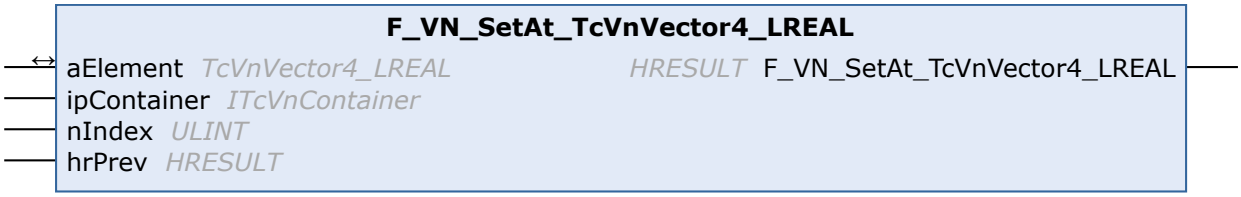
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.29 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 [▶ 370]	Container with TcVnVector4_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

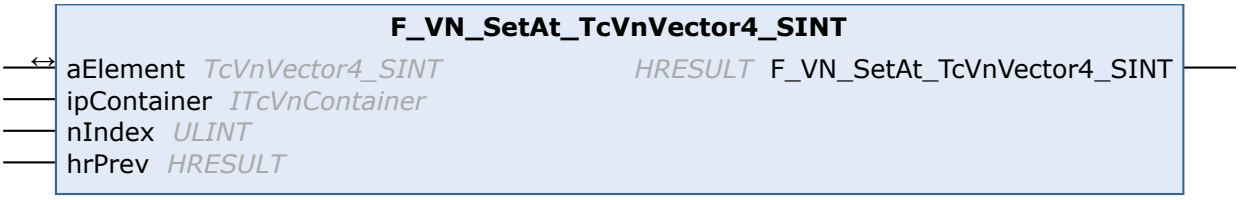
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.30 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 [▶ 370]	Container with TcVnVector4_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

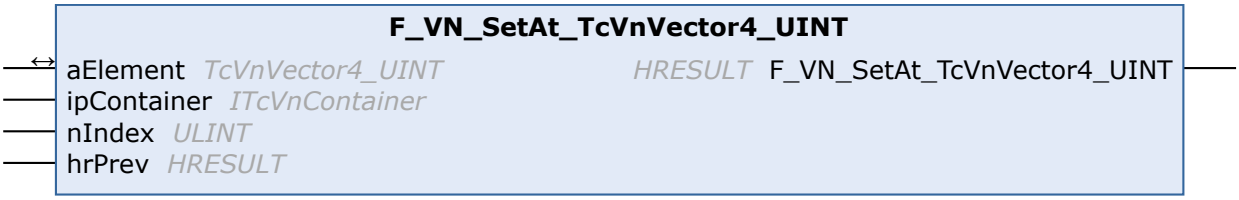
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.31 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 [▶ 370]	Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

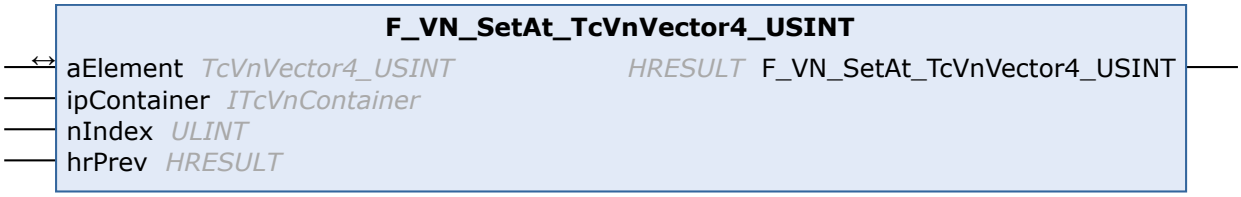
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.32 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 [▶ 370]	Container with TcVnVector4_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

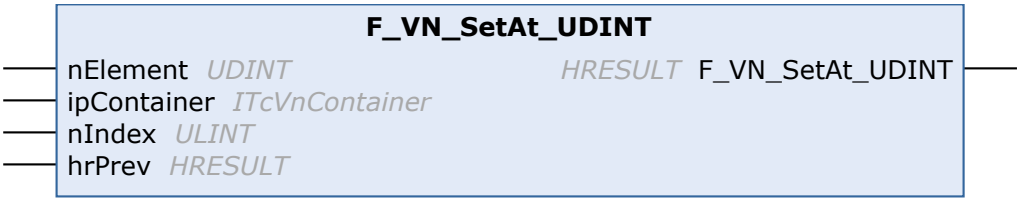
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.33 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 [▶ 370]	Container with UDINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

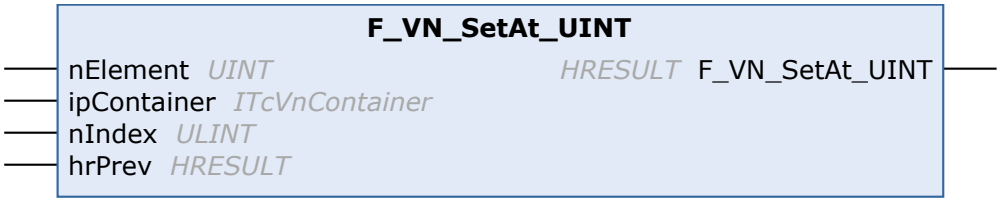
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.34 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 [▶ 370]	Container with UINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.35 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 [▶ 370]	Container with ULINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

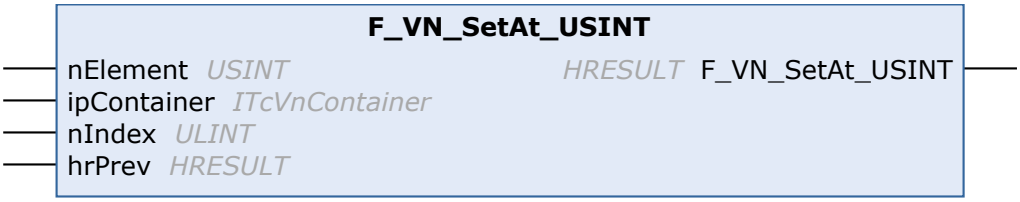
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.36 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 [▶ 370]	Container with USINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

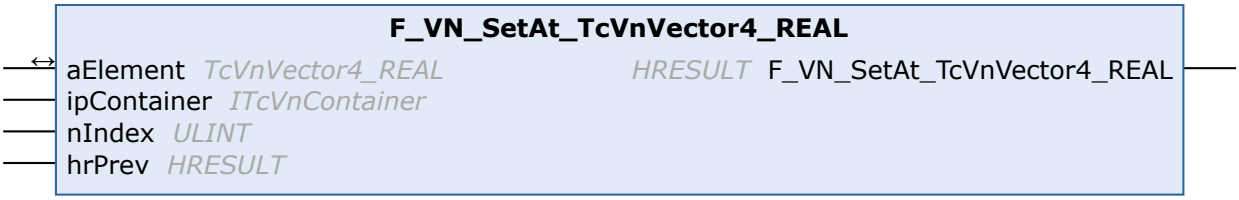
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.37 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 [▶ 370]	Container with TcVnVector4_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 162]	Element to set at the specified container position

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator to be advanced
nOffset	LINT	Offset, by which ipIterator is advanced
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_AdvanceIterator` increments or decrements an iterator [▶ 359] by a certain offset `nOffset`.

If the offset `nOffset` is always to be `1`, the simpler function F_VN_IncrementIterator [▶ 768] can also be used.

Parameter

Iterator

The parameter `ipIterator` is of the type `ITcVnForwardIterator`. Since `ITcVnBidirectionalIterator` and `ITcVnRandomAccessIterator` inherit from it, these can also be transferred to the function. If a forward iterator is to be decremented, an internal attempt is made to transform it into a bidirectional iterator and to decrement it.

Offset

The offset `nOffset` defines by how much the iterator `ipIterator` is to be incremented or decremented. If `nOffset > 0` it is incremented; if `nOffset < 0` it is decremented.

Related functions

- F_VN_IncrementIterator [▶ 768]
- F_VN_AdvanceIterator [▶ 745]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns the converted container
nDestTypeGuid	GUID	Specifies the destination container type for the conversion
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ConvertContainerType` converts a container into a container type [▶ 160]that is to be specified.

The global ContainerType [▶ 160] constants can be used to determine the container type via the parameter `nTypeGuid`.

HRESULT

70E	INCOMPATIBLE	The same interface pointer is used as the input container and result container, or a container that has specified a different type to that in `nDestTypeGuid` is used as an interface pointer.

Application

A typical use case, for example, is the conversion of a container so that it can be drawn in an image. A contour of the type `ContainerType_Vector_TcVnPoint2_REAL`, as is returned among others by the function F_VN_LocateEllipseExp [▶ 1510], can be converted into the type `ContainerType_Vector_TcVnPoint2_DINT` in order to be able to draw it in an image using the function F_VN_DrawContours [▶ 1009]:

```
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 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Destination container (same type ID as ipSrcContainer; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyContainer` not only copies the interface pointer [▶ 137] of the container [▶ 152], it also makes a deep copy of the container.

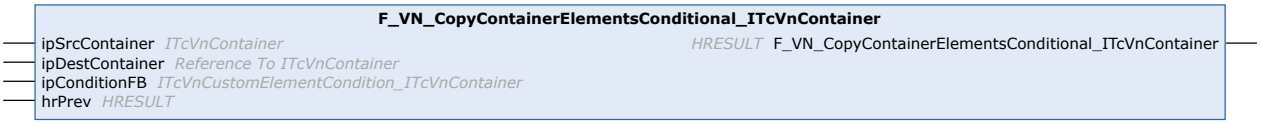
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnContainer [▶ 248]	Custom condition
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyContainerElementsConditional` filters elements of multi-dimensional containers [▶ 152]. The filter condition is to be specified by a function block with the interface ITcVnCustomElementCondition_ITcVnContainer [▶ 248].

Parameter

Input container

The input container `ipSrcContainer` must be a multi-dimensional container; i.e. the container must contain sub-containers as elements. Thus all containers of the type `ContainerType_Vector_Vector_<...>` can be used.

Result container

The result container `ipDestContainer` contains all sub-containers that satisfy the condition in `ipConditionFB`.

Filter condition

The filter condition must be defined as the method `Condition` of a function block. So that this function block can be transferred to the function as the parameter `ipConditionFB`, it must implement the interface ITcVnCustomElementCondition_ITcVnContainer [▶ 248].

Samples

- Selections of container elements [▶ 2755]

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751]

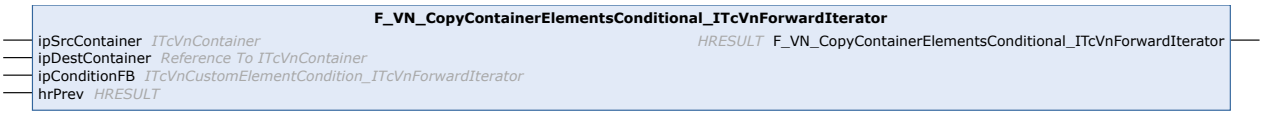
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 249]	Custom condition
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyContainerElementsConditional_ITcVnForwardIterator` filters elements of multi-dimensional containers [▶ 152]. The filter condition is to be specified by a function block with the interface ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 249].

Unlike F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750], sub-containers are not transferred to the conditional method `Condition`, but iterators, which each point to the first element of the sub-container.

Parameter

Input container

The input container `ipSrcContainer` must be a multi-dimensional container; i.e. the container must contain sub-containers as elements. Thus all containers of the type `ContainerType_Vector_Vector_<...>` can be used.

Result container

The result container `ipDestContainer` contains all sub-containers that satisfy the condition in `ipConditionFB`.

Filter condition

The filter condition must be defined as the method `Condition` of a function block. So that this function block can be transferred to the function as the parameter `ipConditionFB`, it must implement the interface ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 249].

Samples

- Selections of container elements [▶ 2755]

Related functions

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751]

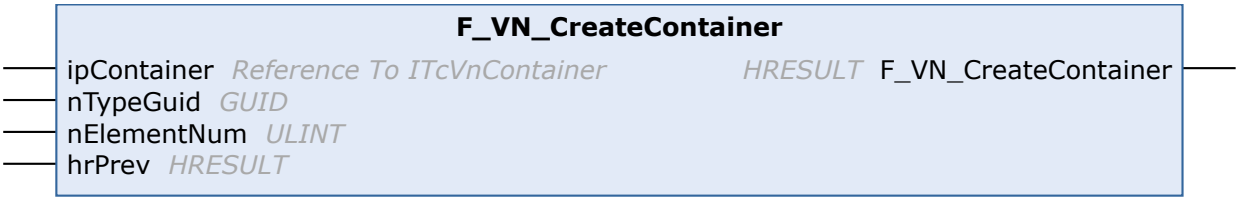
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULINT	Number of elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The global ContainerType [▶ 160] constants can be used to determine the container type via the parameter `nTypeGuid`.

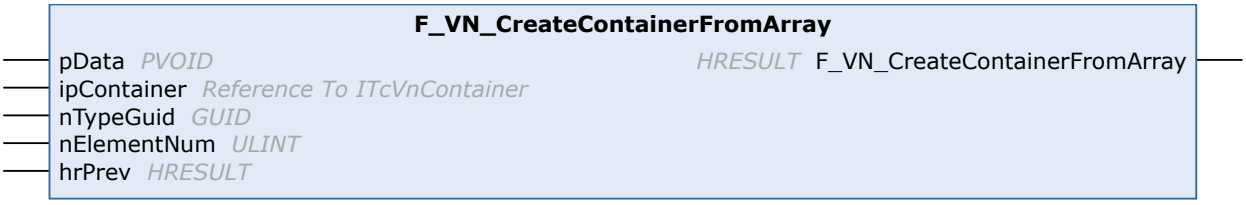
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The global ContainerType [▶ 160] constants can be used to determine the container type via the parameter `nTypeGuid`.

Since the array data is only specified via a pointer, a type check within the function is not possible. Therefore, please note the following warning:

NOTICE

Memory access

It must be ensured that the type and length of the container correspond to the size of the memory area at the pointer. Otherwise, this can lead to a system crash or invalid data.

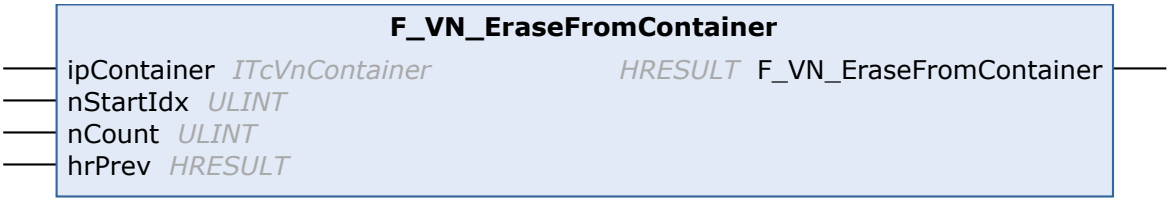
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container from which to erase elements
nStartIdx	ULINT	Index of the first element to erase
nCount	ULINT	Number of elements to erase
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ExportContainer` exports a container to a data buffer. Make sure that sufficient memory is allocated at the place of the pointer in order to write the container data to it! The required size in bytes can be determined with the function F_VN_ExportContainerSize [▶ 758]. You can allocate the memory, for example, by creating an array of the appropriate size.

Application

The export of a container of the type `Vector_REAL` with `10` elements looks like this, for example:

```
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
```

NOTICE

Memory access

It must be ensured that the type and length of the container correspond to the size of the memory area at the pointer. Otherwise, this can lead to a system crash or invalid data.

Related functions

- F_VN_ExportContainer [▶ 755]

- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Related functions

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Related functions

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Related functions

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Application

The function `F_VN_ExportSubContainer` is used, for example, to obtain the result text in the case of Code Reading [▶ 837]:

```
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
```

Related functions

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

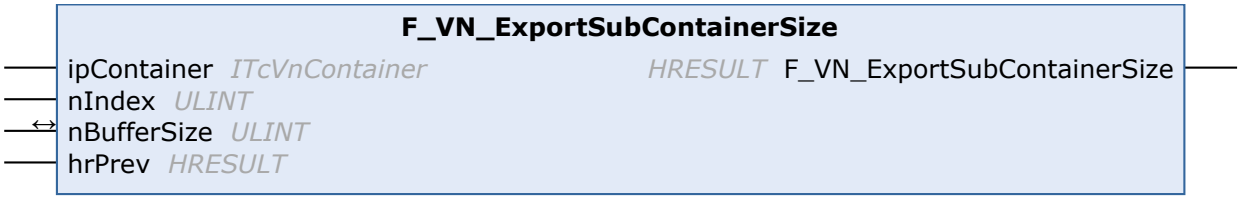
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with basic elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Related functions

- F_VN_ExportContainer [▶ 755]
- F_VN_ExportSubContainer [▶ 759]
- F_VN_ExportContainer_String [▶ 757]
- F_VN_ExportSubContainer_String [▶ 760]
- F_VN_ExportContainerSize [▶ 758]
- F_VN_ExportSubContainerSize [▶ 762]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 370]	Destination container
nFirstIdx	ULINT	Index of the first element to copy
nLastIdx	ULINT	Index of the last element to copy
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

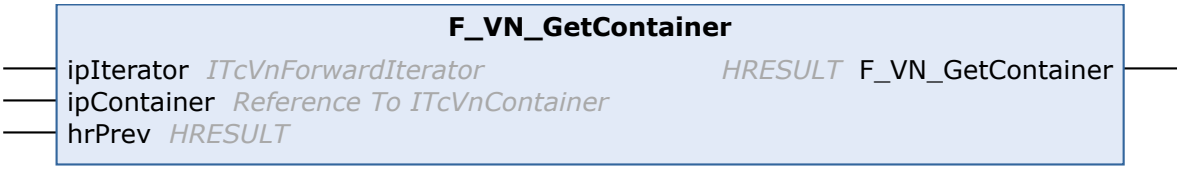
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
ipContainer	Reference To ITcVnContainer [▶ 370]	Returns the container interface
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

Deep copy

This function creates a deep copy of the container element. This means that if you wish to change the element in the container, you must write the changes back to the container separately.

If you wish to increment the iterator at the same time as accessing the container element, the expert version F_VN_GetContainerExp [▶ 765] is available for this.

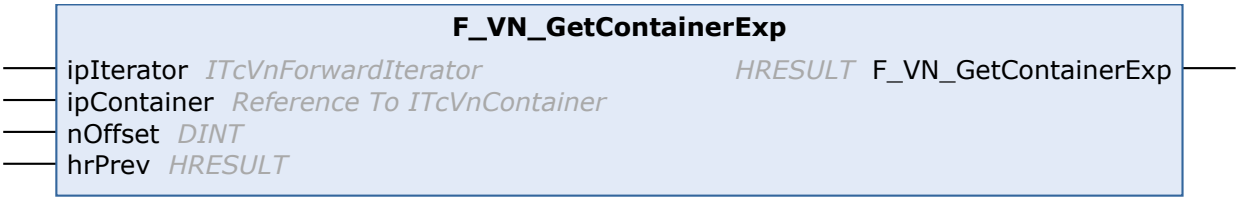
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
ipContainer	Reference To ITcVnContainer [▶ 370]	Returns the container interface
nOffset	DINT	Offset to increment (>0) or decrement (<0) the iterator afterwards
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetContainerExp` is the expert version of F_VN_GetContainer [▶ 764]. It helps you if, at the same time as accessing a container element by means of an iterator, you also wish to increment this iterator. Set the parameter `nOffset` to the value `1` for a normal, sequential incrementation:

```
hr := F_VN_GetContainerExp(ipIterator, ipElement, 1, hr);
```

With the standard function you would need two function calls for this:

```
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 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 [▶ 370]	Container
ipIterator	Reference To ITcVnForwardIterator [▶ 360]	Returns the iterator interface
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

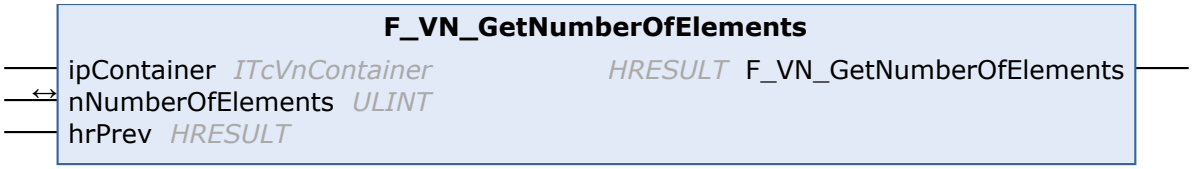
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

This function should not be used to iterate over the elements of a container. Use iterators to access the elements [▶ 153] instead. In particular, the following code construct is not recommended:

```
hr := F_VN_GetNumberOfElements(ipContainer, ipNumber, hr);
FOR i:=0 TO (ipNumber-1) DO
    // access container elements
END_FOR
```

Since `ipNumber` is a `UDINT` (natural integer), a return value of `ipNumber = 0` would cause an underflow.

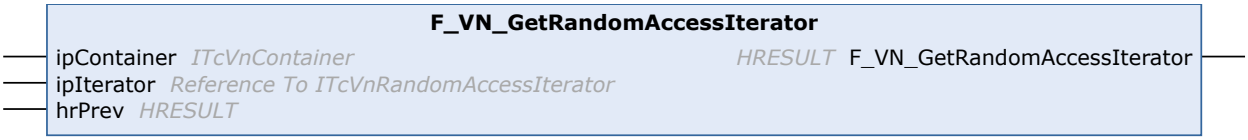
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container
ipIterator	Reference To ITcVnRandomAccessIterator [▶ 367]	Returns the iterator interface
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

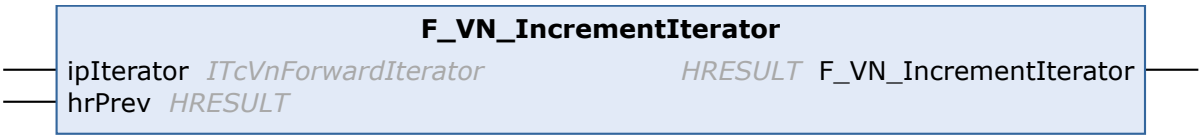
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_IncrementIterator` increments an iterator [▶ 359] by one element. If instead of that you wish to decrement the iterator or to increment it by several positions, use the function F_VN_AdvanceIterator [▶ 745].

Parameter

Iterator

The parameter `ipIterator` is of the type `ITcVnForwardIterator`. Since `ITcVnBidirectionalIterator` and `ITcVnRandomAccessIterator` inherit from it, these can also be transferred to the function. If a forward iterator is to be decremented, an internal attempt is made to transform it into a bidirectional iterator and to decrement it.

Related functions

- F_VN_IncrementIterator [▶ 768]
- F_VN_AdvanceIterator [▶ 745]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	First iterator
ipIteratorLast	ITcVnForwardIterator [▶ 360]	Last iterator
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

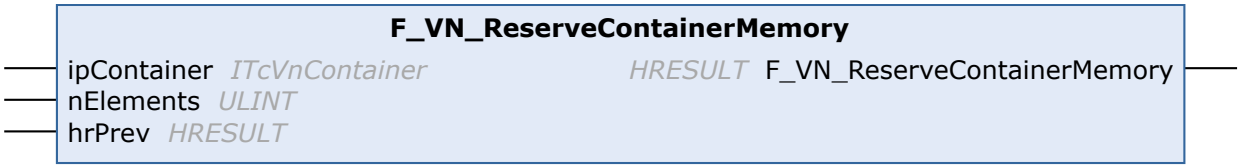
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container for which to reserve the memory
nElements	ULINT	Number of elements for which the container should reserve memory
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

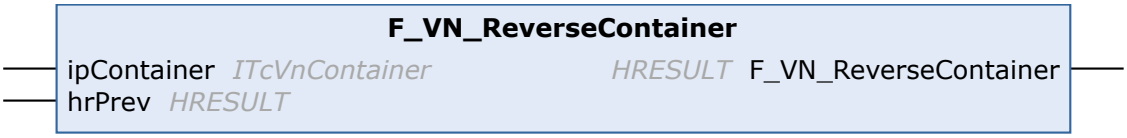
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container to be reversed
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
ipContainer	ITcVnContainer [▶ 370]	Container interface of which the content is to be assigned to the current element
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

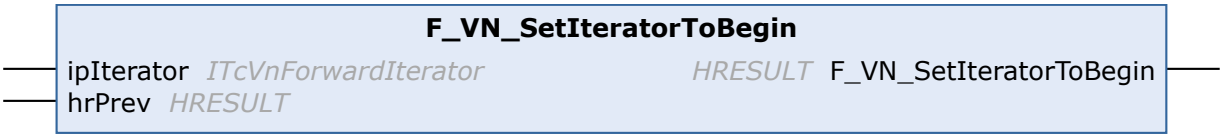
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 360]	Iterator
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5 Basic Image Operations

The group contains functions for handling images [▶ 149].

Functions

Creation of images

- F_VN_ConvertElementType(Exp) [▶ 775]
- F_VN_CreateAssociatedImage [▶ 782]
- F_VN_CreateEmptyImage [▶ 783]
- F_VN_CreateImage [▶ 784]
- F_VN_CreateImageAndSetPixels [▶ 785]
- F_VN_CreateImageFromArray [▶ 786]
- F_VN_CreateNDimensionalImage [▶ 814]
- F_VN_CreateNDimensionalImageAndSetPixels [▶ 815]
- F_VN_CreateNDimensionalImageFromArray [▶ 816]

Image information

- F_VN_GetImageHeight [▶ 796]
- F_VN_GetImageInfo [▶ 797]
- F_VN_GetImageWidth [▶ 798]
- F_VN_GetLayoutInfo [▶ 817]

- F_VN_GetPixelFormat [▶ 800]

Copying images

- F_VN_CopyImage [▶ 777]
- F_VN_CopyImageRegion [▶ 778]
- F_VN_CopyImageRegionToRegion [▶ 780]

Pixel manipulation

- F_VN_GetPixel [▶ 799]
- F_VN_SetPixel [▶ 805]
- F_VN_SetPixels(Exp) [▶ 806]

Region of Interest

- F_VN_GetRoi [▶ 801]
- F_VN_ResetRoi [▶ 803]
- F_VN_SetRoi [▶ 807]
- F_VN_SetRoi_TcVnRectangle_DINT [▶ 809]
- F_VN_SetRoi_TcVnRectangle_UDINT [▶ 809]

Exporting image content to memory area

- F_VN_ExportImage [▶ 787]
- F_VN_ExportImageAsBmp(Exp) [▶ 788]
- F_VN_ExportImageSize [▶ 791]
- F_VN_ExportImageAsBmpSize(Exp) [▶ 790]

Fusing images

- F_VN_FuseImages [▶ 792]
- F_VN_FuseImagesArray [▶ 794]

Handling image channels

- F_VN_CombineImageChannels [▶ 775]
- F_VN_GetImageChannel [▶ 795]
- F_VN_MixImageChannels [▶ 802]
- F_VN_SetImageChannel [▶ 804]
- F_VN_SplitImageChannels [▶ 811]

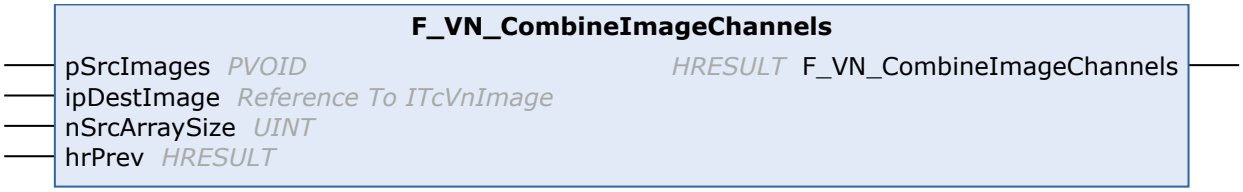
Displaying images

- F_VN_CopyIntoDisplayableImage [▶ 781]
- F_VN_TransformIntoDisplayableImage(Exp) [▶ 812]

Samples

Basic Image Operations [▶ 2757]

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 [▶ 411]	Multi-channel destination image (An appropriate destination image will be created if required.)
nSrcArraySize	UINT	pSrcImages array size
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

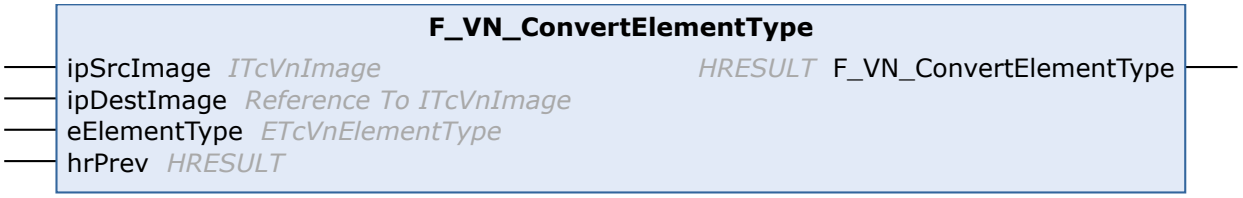
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 199]	Desired element type of the destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

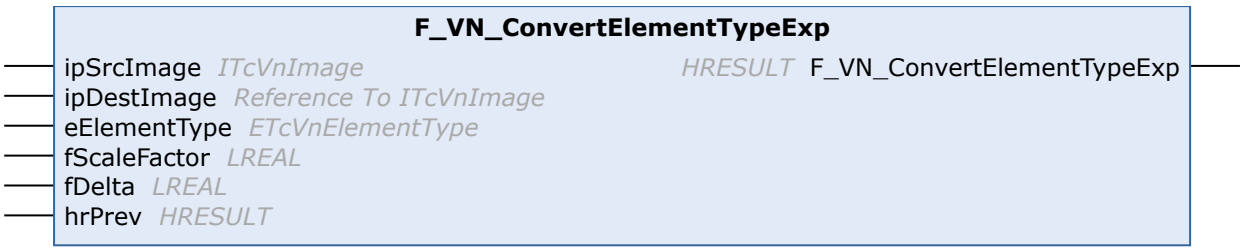
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyImage` creates a deep copy of the input image `ipSrcImage`. Alternatively, the function F_VN_CopyImageRegion [▶ 778] can be used to create a copy of an image region.

Computing time optimization

With regard to computing time optimization, check each image copy for its necessity.

Related functions

- F_VN_CopyImage [▶ 777] for copying the entire image
- F_VN_CopyImageRegion [▶ 778] for copying an image region
- F_VN_CopyImageRegionToRegion [▶ 780] for copying an image region to a certain 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 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 [▶ 411]	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 [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyImageRegion` creates a deep copy of an image region of the input image `ipSrcImage`. The image region is specified by a rectangle with position and size, as with an ROI [▶ 158].

Computing time optimization

With regard to computing time optimization, check each image copy for its necessity.

If a copy is not required, but instead only the image region is to be reduced in size, a Region of Interest [▶ 158] can be set instead.

Application

The copying of an image region of the size `[240, 120]` with the top left corner at the position `[50, 50]` looks like this, for example:

```
hr := F_VN_CopyImageRegion(
    ipSrcImage  :=  ipImageIn,
    nX          :=  50,
    nY          :=  50,
    nWidth      :=  240,
    nHeight     :=  120,
    ipDestImage :=  ipImageWork,
    hrPrev      :=  hr,
);
```

Related functions

- F_VN_CopyImage [▶ 777] for copying the entire image
- F_VN_CopyImageRegion [▶ 778] for copying an image region
- F_VN_CopyImageRegionToRegion [▶ 780] for copying an image region to a certain position

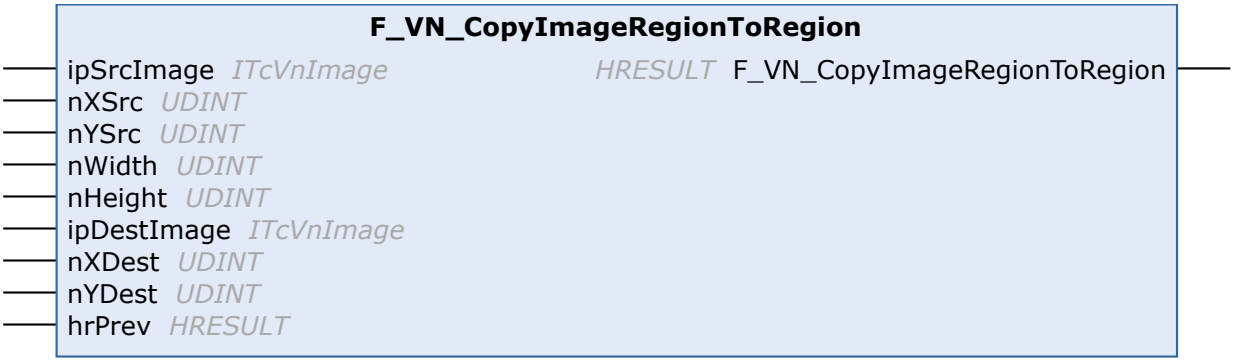
Required License

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 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 [▶ 411]	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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CopyImageRegionToRegion` copies an image region of the input image `ipSrcImage` to a place in the destination image `ipDestImage`. The image region is specified by a rectangle with position and size, as with an ROI [▶ 158].

Application

The copying of an image area with a size `[100, 100]` at the position `aPosition` in the top left corner of the result image `ipImageRes` looks like this, for example:

```
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage  :=  ipImageWork,
    nXSrc       :=  aPosition[0],
    nYSrc       :=  aPosition[1],
    nWidth      :=  100,
    nHeight     :=  100,
    ipDestImage :=  ipImageRes,
    nXDest      :=  0,
    nYDest      :=  0,
    hrPrev      :=  hr
);
```

Samples

- Copy image areas [▶ 2758]

Related functions

- F_VN_CopyImage [▶ 777] for copying the entire image
- F_VN_CopyImageRegion [▶ 778] for copying an image region
- F_VN_CopyImageRegionToRegion [▶ 780] for copying an image region to a certain position

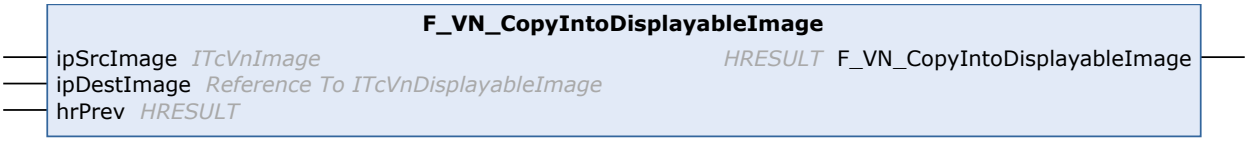
Required License

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 411]	Returns the displayable image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

For further details, see Displayable images [▶ 151].

Related functions

- F_VN_TransformIntoDisplayableImage [▶ 812]
- F_VN_TransformIntoDisplayableImageExp [▶ 813]

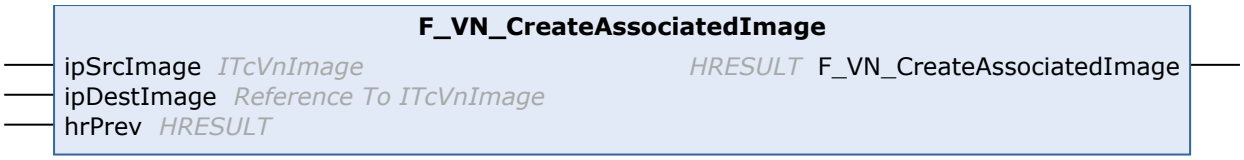
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

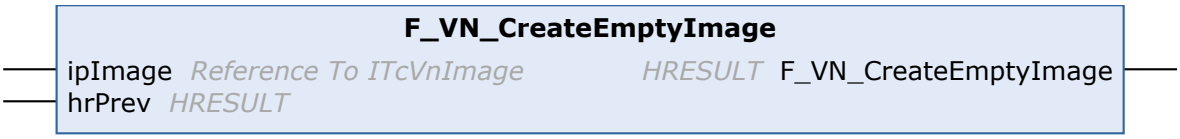
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

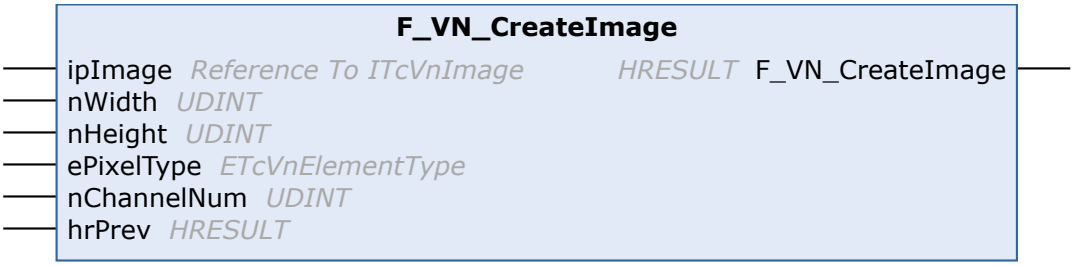
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 199]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 255)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 199]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 4)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Pixel value

Return value

HRESULT [▶ 141]

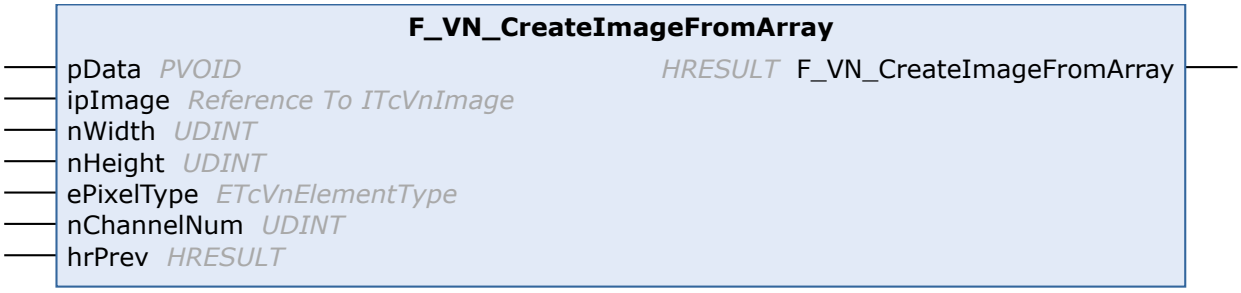
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 199]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 255)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

Note the conversion properties of the interface ITcVnBitmapExport [▶ 394].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

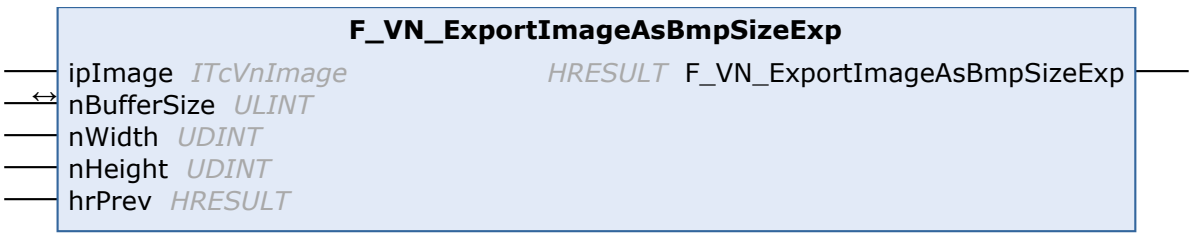
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

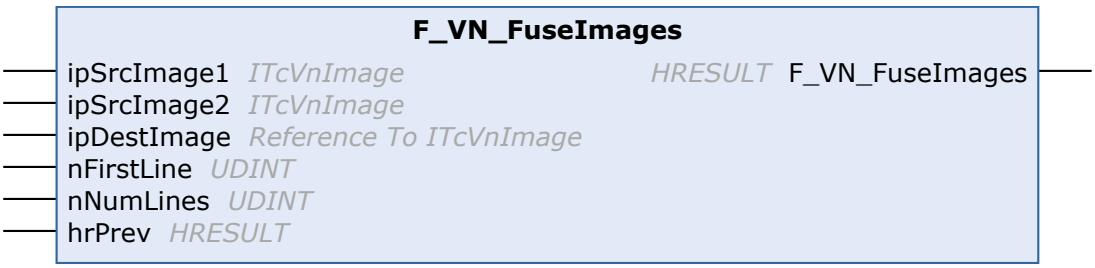
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image 1
ipSrcImage2	ITcVnImage [▶ 411]	Source image 2
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

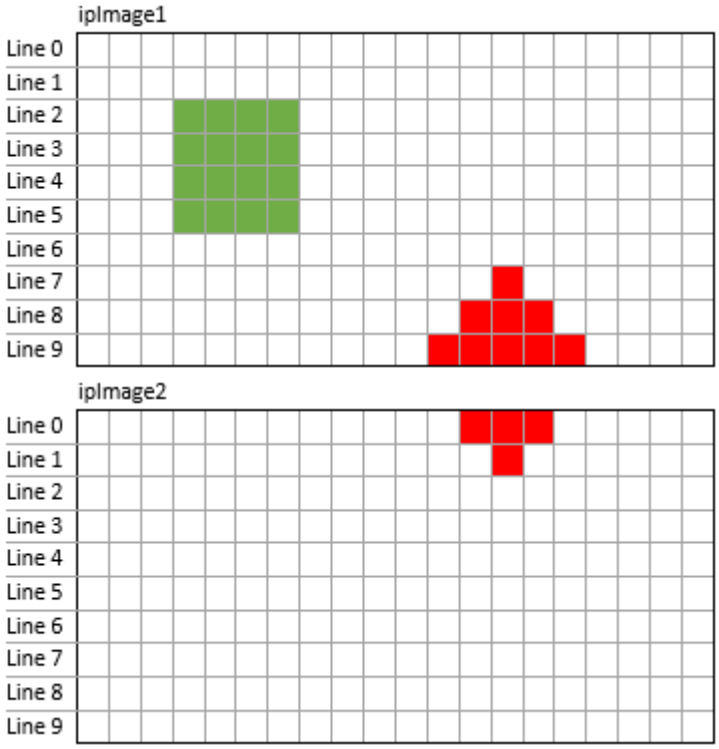
Further information

The function `F_VN_FuseImages` joins two images to form a new contiguous image. This is of particular interest for line scan cameras.

The function F_VN_FuseImagesArray [▶ 794] can be used to fuse up to 10 images.

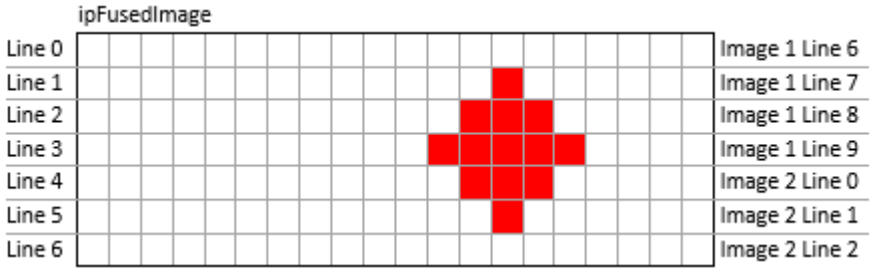
Application

Line scan cameras usually return a given number of successively captured lines as a 2D image. This 2D image can contain an entire object of interest (shown in green in the figure below), or it can contain part of an object (shown in red in the figure).



To analyze the red object, the function `F_VN_FuseImages` can be used to fuse the two object parts to form an complete image. If it is known which lines relate to the object, the linking can be limited to the relevant lines.

```
F_VN_FuseImages(ipImage1, ipImage2, ipFusedImage, 6, 7, S_OK);
```



Samples

- Fusing several images [▶ 2757]

Related functions

- F_VN_FuseImages [▶ 792] for fusing two images
- F_VN_FuseImagesArray [▶ 794] for fusing up to 10 images

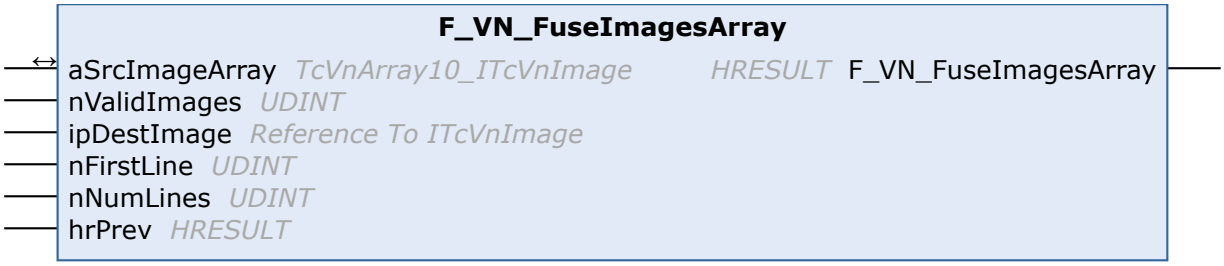
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcImageArray	TcVnArray10_ITcVnImage [▶ 162]	Ten-element source image array (not all elements need to be filled, nValidImages specifies the actual amount of images)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FuseImageArray` is a modified version of the function F_VN_FuseImages [▶ 792] and can fuse up to 10 images instead of two images.

Samples

- Fusing several images [▶ 2757]

Related functions

- F_VN_FuseImages [▶ 792] for fusing two images
- F_VN_FuseImagesArray [▶ 794] for fusing up to 10 images

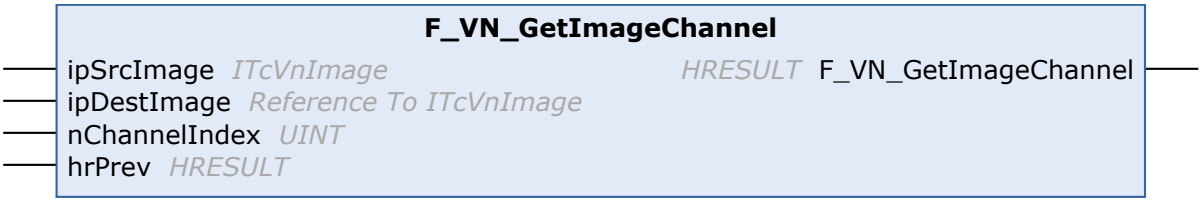
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
nChannelIndex	UINT	Index of the requested source image channel
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

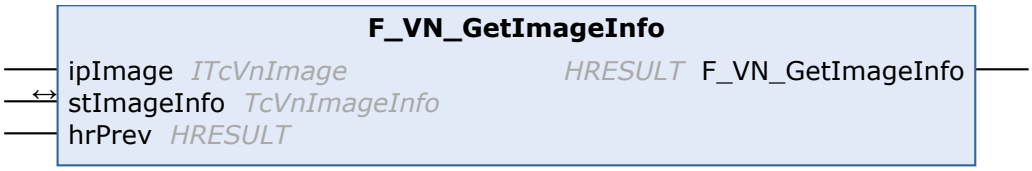
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stImageInfo	TcVnImageInfo [▶ 230]	Returns a struct describing the image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
nX	UDINT	x coordinate of the pixel
nY	UDINT	y coordinate of the pixel
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Returns the pixel value (Unused channels are set to 0.)

Return value

HRESULT [▶ 141]

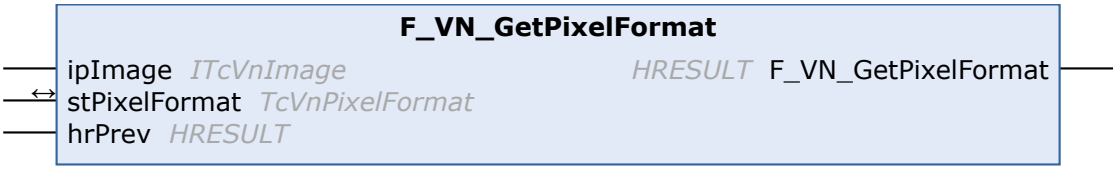
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stPixelFormat	TcVnPixelFormat [▶ 243]	Returns a struct describing the pixel format

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_UDINT [▶ 245]	Returns the coordinates of the region of interest

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetRoi` indicates which Region Of Interest [▶ 158] is currently set on the input image `ipSrcImage`. The information is required, for example, if points in the image are to be specified in relation to the total image or if the ROI is to be dynamically enlarged.

Application

The referencing of a point to the coordinates of the original image looks like this, for example:

```
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;
```

Related functions

- F_VN_SetRoi [▶ 807] for setting a Region of Interest
- F_VN_GetRoi [▶ 801] for retrieving the set Region of Interest
- F_VN_ResetRoi [▶ 803] for resetting the Region of Interest on the entire image

- F_VN_CopyImageRegion [▶ 779] for copying an image region

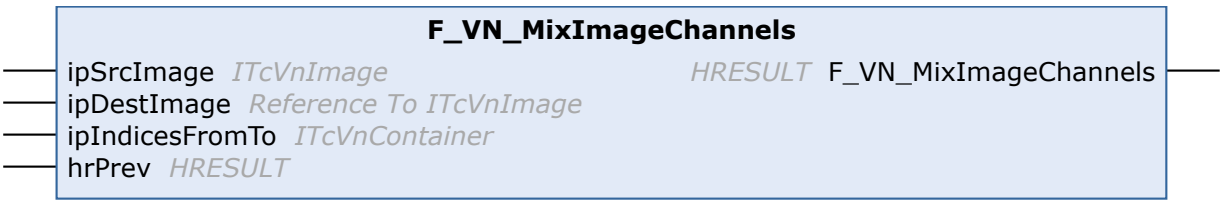
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
ipIndicesFromTo	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

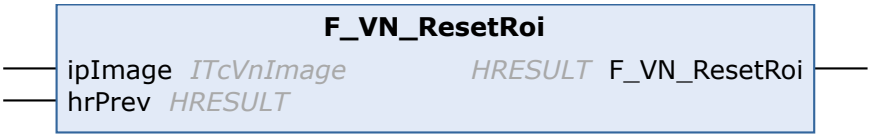
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ResetRoi` resets the Region of Interest [▶ 158] of an image. After executing the function, the ROI of the image is at the position `[0, 0]` and has the size `[ImageWidth, ImageHeight]`.

Application

The resetting of an ROI looks like this, for example:

```
Hr:= F_VN_ResetRoi(ipImageWork, hr);
```

Related functions

- F_VN_SetRoi [▶ 807] for setting a Region of Interest
- F_VN_GetRoi [▶ 801] for retrieving the set Region of Interest
- F_VN_ResetRoi [▶ 803] for resetting the Region of Interest on the entire image
- F_VN_CopyImageRegion [▶ 779] for copying an image region

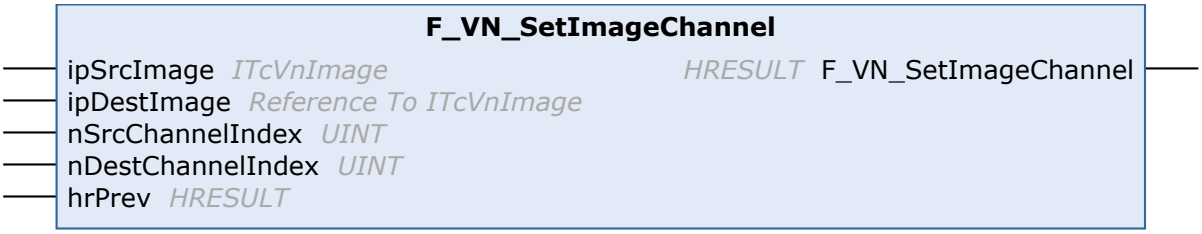
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

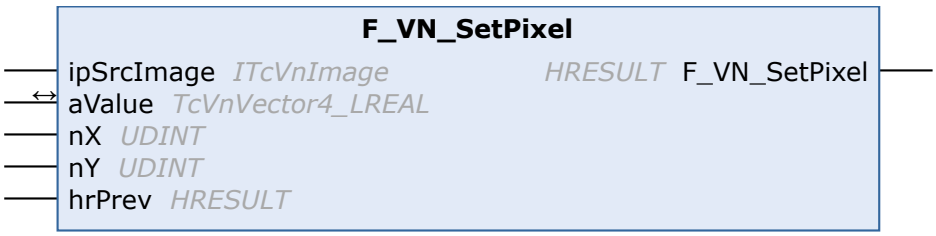
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
nX	UDINT	x coordinate of the pixel
nY	UDINT	y coordinate of the pixel
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	The pixel value to set (additional channels are ignored.)

Return value

HRESULT [▶ 141]

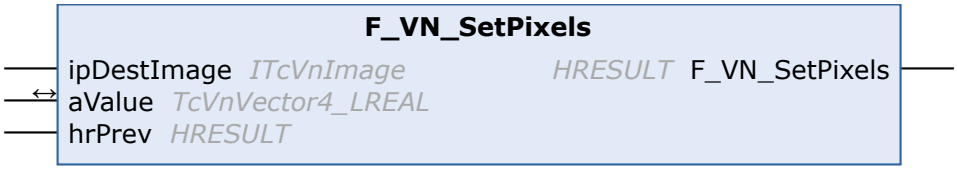
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Pixel value

Return value

HRESULT [▶ 141]

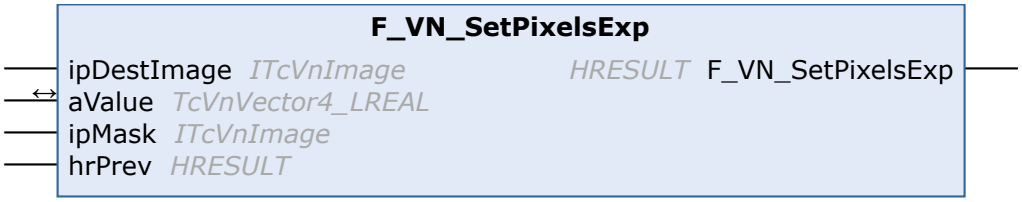
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
ipMask	ITcVnImage [▶ 411]	Mask image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 162]	Pixel value

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_SetRoi` sets the Region of Interest [▶ 158] of an image. The function F_VN_SetRoi_TcVnRectangle_UDINT [▶ 809] can be used if the rectangle parameters are not to be transferred individually, but as a structure of the type TcVnRectangle_UDINT [▶ 245].

Application

The setting of an ROI with the size `[240, 120]` at the position `[50, 50]` looks like this, for example:

```
hr := F_VN_SetRoi(50, 50, 240, 120, ipImageWork, hr);
```

Related functions

- F_VN_SetRoi [▶ 807] for setting a Region of Interest
- F_VN_GetRoi [▶ 801] for retrieving the set Region of Interest
- F_VN_ResetRoi [▶ 803] for resetting the Region of Interest on the entire image
- F_VN_CopyImageRegion [▶ 779] for copying an image region

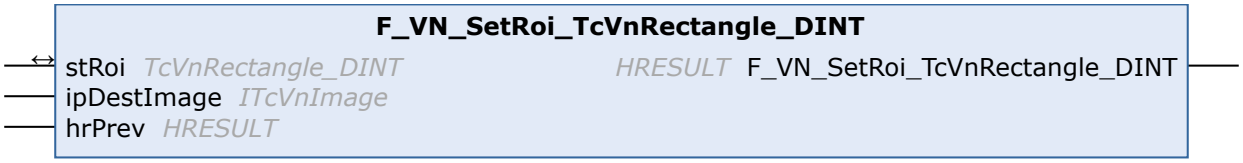
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_DINT [▶ 244]	Region of interest

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_UDINT [▶ 245]	Region of interest

Return value

HRESULT [▶ 141]

Further information

Like the function F_VN_SetRoi [▶ 807], the function `F_VN_SetRoi_TcVnRectangle_UDINT` sets a Region of Interest [▶ 158]. However, it doesn't accept the rectangle parameters individually, but as a structure of the type TcVnRectangle_UDINT [▶ 245].

This makes it possible, for example, to set the ROI directly with the result of the function F_VN_UprightBoundingRectangle [▶ 988].

Application

The setting of an ROI around a found contour [▶ 155] looks like this:

```
hr := F_VN_UprightBoundingRectangle(ipContour, stRoi, hr);
hr := F_VN_SetRoi_TcVnRectangle_UDINT(stRoi, ipImageWork, hr);
```

Related functions

- F_VN_SetRoi [▶ 807] for setting a Region of Interest
- F_VN_GetRoi [▶ 801] for retrieving the set Region of Interest
- F_VN_ResetRoi [▶ 803] for resetting the Region of Interest on the entire image
- F_VN_CopyImageRegion [▶ 779] for copying an image region

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 411]	Returns the displayable image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

For further details, see Displayable images [▶ 151].

NOTICE

Use with Ads Communicator object

Note the case where the image to be transformed originates from a camera object whose Ads Communicator object is active! It may then be the case that the image is not internally released in the object and therefore cannot be transformed into a displayable image. In this case, use F_VN_CopyIntoDisplayableImage [▶ 781] or F_VN_TransformIntoDisplayableImageExp [▶ 813] with `bAllowDeepCopy := TRUE`.

Related functions

- F_VN_CopyIntoDisplayableImage [▶ 781]
- F_VN_TransformIntoDisplayableImageExp [▶ 813]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 411]	Returns the displayable image
bAllowDeepCopy	BOOL	Allow deep image copies, if required
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

For further details, see Displayable images [▶ 151].

Parameter

Permission for deep copy

The permission (`TRUE`) for a deep copy can be given with the parameter `bAllowDeepCopy`. This is relevant if the image to be transformed originates from a camera object whose Ads Communicator object [▶ 124] is activated and the image therefore can be internally referenced.

The standard version F_VN_TransformIntoDisplayableImage [▶ 812] can also be used if no permission is to be given (`FALSE`).

Related functions

- F_VN_CopyIntoDisplayableImage [▶ 781]
- F_VN_TransformIntoDisplayableImage [▶ 812]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 199]	Pixel type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 162]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 199]	Pixel type
fValue	LREAL	Pixel value
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 162]	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 [▶ 141]

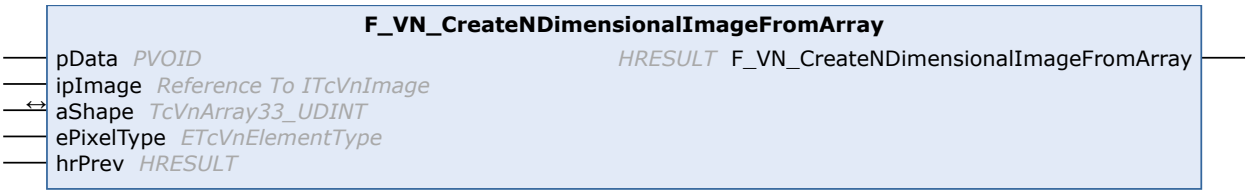
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 199]	Pixel type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 162]	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 [▶ 141]

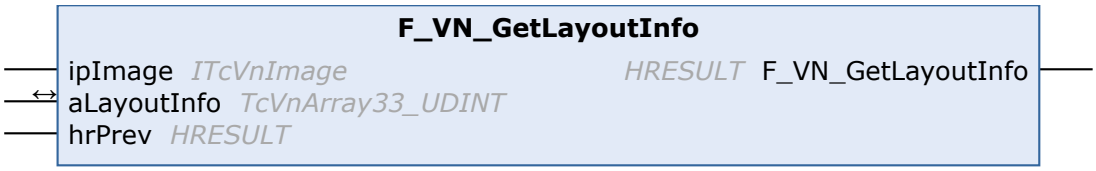
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLayoutInfo	TcVnArray33_UDINT [▶ 162]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6 Code Quality

This group contains functions for quality assessment of various 1D and 2D codes. The assessment is based on international standards. The code quality check is intended to ensure that the corresponding code can be reliably decoded at the time of use.

In order to obtain comparable results, the standards define methods for measuring and evaluating specific attributes and for deriving an overall assessment of code quality. The aim is to predict, as far as possible, the expected readability of the codes, using different types of readers, in different applications, environments and under different conditions. A minimum acceptable quality level is therefore usually defined for each individual application.

On the one hand, the standards describe requirements that must be met by the systems used in order to obtain usable results. On the other hand, the standards provide recommendations such as illumination, lens settings and geometric structure in order to be able to compare the ratings with other systems. In order to set up a system that conforms to the standard, it is advisable to obtain the relevant quality assessment specifications and the codes to be tested.

The functions can also be used with non-standard systems. The only consequence of this is that the quality determined is not comparable with other systems. In general, minor differences in the individual evaluation categories are to be expected for different systems.

In accordance with the ISO/IEC 15416:2016 standard, the print quality of barcodes for the code types Code 39, Code 128, EAN 8, EAN 13, ITF, UPC-A, UPC-E can be determined using the following functions:

- F_VN_GradeBarcode(Exp) [▶ 819]

In accordance with the ISO/IEC 15415:2011 standard, the print quality of two-dimensional matrix symbols of the ECC200 Data Matrix code and QR code types can be determined using the following functions:

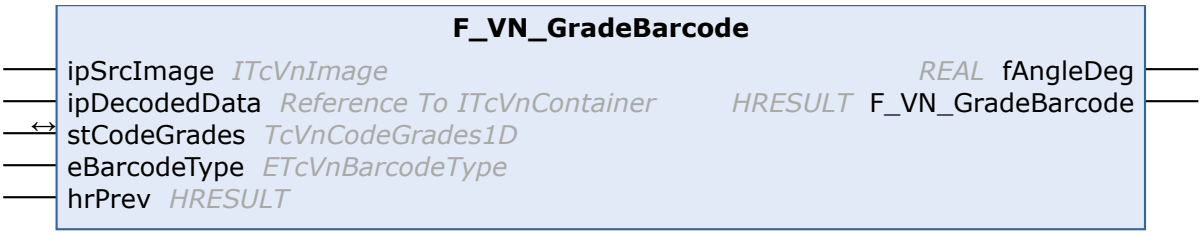
- F_VN_GradeDataMatrixCode(Exp) [▶ 826]
- F_VN_GradeQRCode(Exp) [▶ 831]

Interpretation of the HRESULT

For the Code Grading functions the HRESULT [▶ 141] return values are used as follows:

Code	Name	Description
16#000	S_OK	Function was executed successfully and all evaluations were carried out in accordance with the standard.
16#001	S_FALSE	Function was executed successfully, not all scan lines could be decoded for 1D codes and not all checks could be performed according to the standard for 2D codes. The overall score was therefore set to 0, but the individual results can be evaluated.
16#256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. Individual results can be evaluated depending on the time of termination
16#70C	NOTFOUND	Function was not executed successfully, no scan line could be decoded for 1D codes and the code was not found or could not be decoded for 2D codes.
16#7xx	All error codes	Function was not executed successfully. See: ADS Return Codes [▶ 2879]

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 [▶ 411]	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 [▶ 370]	Returns the decoded data (ContainerType_String_SINT)
eBarcodeType	ETcVnBarcodeType [▶ 164]	Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGrades1D [▶ 228]	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 [▶ 141]

Further information

The `F_VN_GradeBarcode` function determines the print quality of barcodes in accordance with the ISO/IEC 15416:2016 standard.

Algorithm

The function first searches for horizontally aligned codes and then for vertically aligned codes in the image.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones. The barcode must be aligned horizontally or vertically and must not have any text or blank space more than 10% of the total height above or below it. The quiet zones are located to the left and right of the code when aligned horizontally.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough to match the line width. For barcodes, the minimum bar width should be covered by at least 3 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGrades1D contains the determined individual results and an overall total result `fOverall`. The quality grades ranges from 0 (very poor) to 4 (very good). The results are averaged over several scan lines.

Parameter	Description
fDecode	Evaluates whether the code could be decoded with the reference algorithm.
fSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value on the entire scan line.
fMinReflectance	Evaluates the ratio of the darkest to the brightest value of the scan lines.
fMinEdgeContrast	Evaluates the minimum local contrast of neighboring elements of the scan lines.
fModulation	Evaluates the ratio between the minimum local contrast and the global symbol contrast.
fDefects	Evaluates irregularities within elements and quiet zones.
fDecodability	Evaluates how reliably the code can be decoded with the reference algorithm.
fOverall	The overall score of the code is the lowest score achieved in the individual tests, averaged over all scan lines. If different scan lines lead to different decoded data, the overall score is 0.

Barcode type

The parameter `eBarcodeType` defines the barcode type whose print quality is to be assessed. The following barcode types from the enum ETcVnBarcodeType [▶ 164] are supported:

- `TCVN_BT_CODE39`
- `TCVN_BT_CODE128`
- `TCVN_BT_EAN8`
- `TCVN_BT_EAN13`
- `TCVN_BT_ITF`
- `TCVN_BT_UPCA`
- `TCVN_BT_UPCE`

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in 90° steps.

Expert parameters

The expert variant F_VN_GradeBarcodeExp [▶ 822] contains additional parameters.

Application

The print quality of a barcode can be determined in this way, for example:

```
hr := F_VN_GradeBarcode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
    eBarcodeType     := eBarcodeType,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

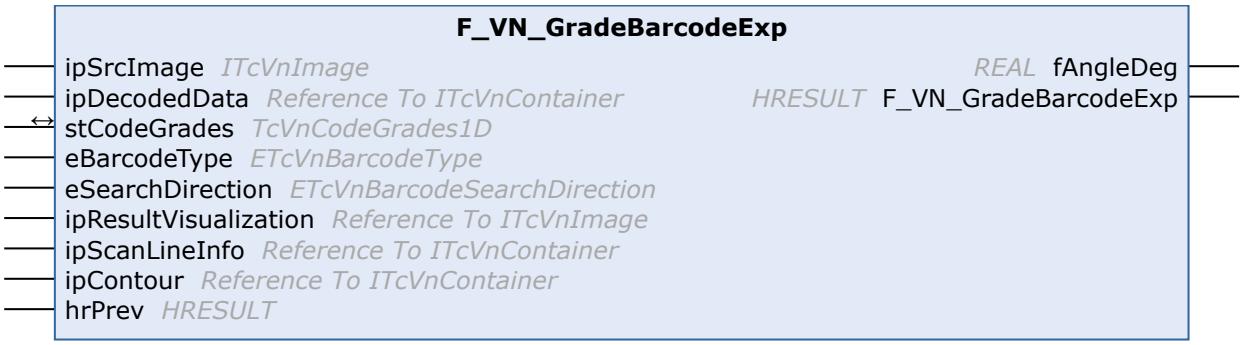
Required License

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 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)

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 [▶ 411]	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 [▶ 370]	Returns the decoded data (ContainerType_String_SINT)
eBarcodeType	ETcVnBarcodeType [▶ 164]	Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 164]	Barcode search direction (BSD_ANY first tries horizontal, then vertical)
ipResultVisualization	Reference To ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 370]	Returns the 4 approximate code corner points, including the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGrades1D [▶ 228]	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 [▶ 141]

Further information

The function `F_VN_GradeBarcodeExp` is an expert variant of F_VN_GradeBarcode [▶ 819]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones. The barcode must be aligned horizontally or vertically and must not have any text or blank space more than 10% of the total height above or below it. The quiet zones are located to the left and right of the code when aligned horizontally.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough to match the line width. For barcodes, the minimum bar width should be covered by at least 3 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGrades1D contains the determined individual results and an overall total result `fOverall`. The quality grades ranges from 0 (very poor) to 4 (very good). The results are averaged over several scan lines.

Parameter	Description
fDecode	Evaluates whether the code could be decoded with the reference algorithm.
fSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value on the entire scan line.
fMinReflectance	Evaluates the ratio of the darkest to the brightest value of the scan lines.
fMinEdgeContrast	Evaluates the minimum local contrast of neighboring elements of the scan lines.
fModulation	Evaluates the ratio between the minimum local contrast and the global symbol contrast.
fDefects	Evaluates irregularities within elements and quiet zones.
fDecodability	Evaluates how reliably the code can be decoded with the reference algorithm.
fOverall	The overall score of the code is the lowest score achieved in the individual tests, averaged over all scan lines. If different scan lines lead to different decoded data, the overall score is 0.

Barcode type

The parameter `eBarcodeType` defines the barcode type whose print quality is to be assessed. The following barcode types from the enum ETcVnBarcodeType [▶ 164] are supported:

- `TCVN_BT_CODE39`
- `TCVN_BT_CODE128`
- `TCVN_BT_EAN8`
- `TCVN_BT_EAN13`
- `TCVN_BT_ITF`
- `TCVN_BT_UPCA`
- `TCVN_BT_UPCE`

Search direction

The search direction can be specified via `eSearchDirection`:

- With `TCVN_BSD_ANY`, the search is first carried out in the horizontal direction and then in the vertical direction.
- With `TCVN_BSD_HORIZONTAL`, the search is only carried out in a horizontal direction.
- With `TCVN_BSD_VERTICAL`, the search is only carried out in the vertical direction.

Presentation of results

`ipResultVisualization` returns an image with the visualization of the scan lines and the respective evaluation results. The lines are color-coded according to the individual grades: `>= 3` as green, `>= 2` as light blue, `>= 1` as orange, `< 1` as red. This parameter is optional and can be set to 0 if it is not required.

Scan line information

`ipScanLineInfo` returns additional information about the individual scan lines for a more detailed evaluation. The return values are of the type `ContainerType_Vector_TcVnVector3_REAL` and specify the index of the image row or image column, the total score and the global threshold determined for each scan line, depending on the orientation. The individual lines can be visualized with F_VN_PlotIntensityProfile [▶ 1049], for example. This parameter is optional and can be set to 0 if it is not required.

Position of the code

The parameter `ipContour` returns the code corner points, including the quiet zone, as four contour points of the type `ContainerType_Vector_TcVnPoint2_DINT`. The calculation is based only on the scan lines that could be decoded. The contour can be drawn directly with F_VN_DrawContours [▶ 1009], for example, or further analyzed with the Contour Analysis [▶ 955] functions. This parameter is optional and can be set to 0 if it is not required.

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in 90° steps.

Application

The print quality of a barcode can be determined in this way, for example:

```
hr := F_VN_GradeBarcodeExp(
    ipSrcImage            := ipSrcImage,
    ipDecodedData         := ipDecodedData,
    stCodeGrades          := stCodeGrades,
    eBarcodeType          := eBarcodeType,
    eSearchDirection      := eSearchDirection,
    ipResultVisualization := ipResultVisualization,
    ipScanLineInfo        := ipScanLineInfo,
    ipContour             := ipContour,
    hrPrev                := hr,
    fAngleDeg             => fAngleDeg);
```

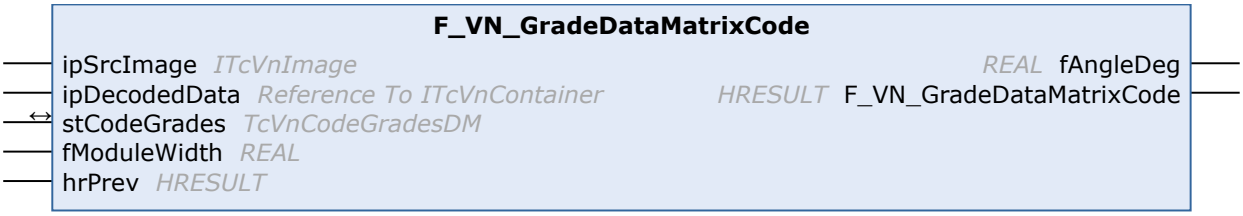
Required License

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 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.

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 [▶ 411]	Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesDM [▶ 228]	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 [▶ 141]

Further information

The function `F_VN_GradeDataMatrixCode` determines the print quality of ECC200 Data Matrix Codes in accordance with the ISO/IEC 15415:2011 standard.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough for the module size. For 2D codes, the smallest module should be covered with at least 5 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGradesDM contains the determined individual results and an overall total result `nOverall`. The quality grades ranges from 0 (very poor) to 4 (very good).

Parameter	Description
nDecode	Evaluates whether the code could be decoded with the reference algorithm.
nSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value in the test area.
nModulation	Evaluates the uniformity of the dark or bright modules.
nReflectanceMargin	Evaluates how well each module can be correctly distinguished as bright or dark compared to the global threshold.
nFixedPatternDamage	Evaluates the various fixed code components such as the finder or alignment pattern.
nAxialNonuniformity	Evaluates the uniform scaling of the code.
nGridNonuniformity	Evaluates deviations of the code grid.
nUnusedErrorCorrection	Evaluates the remaining capacity of the error correction.
nOverall	The overall grade of the code is the lowest grade achieved in the individual examinations.

Minimum module width

The expected minimum module width of the code in pixels is specified with `fModuleWidth`. If the module size in the image differs significantly from this, this can lead to inaccurate or no results and increased computing time.

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in degrees.

Expert parameters

The expert variant F_VN_GradeDataMatrixCodeExp [▶ 828] contains additional parameters.

Application

The print quality of a Data Matrix Code can be determined in this way, for example:

```
hr := F_VN_GradeDataMatrixCode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
    fModuleWidth     := fModuleWidth,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

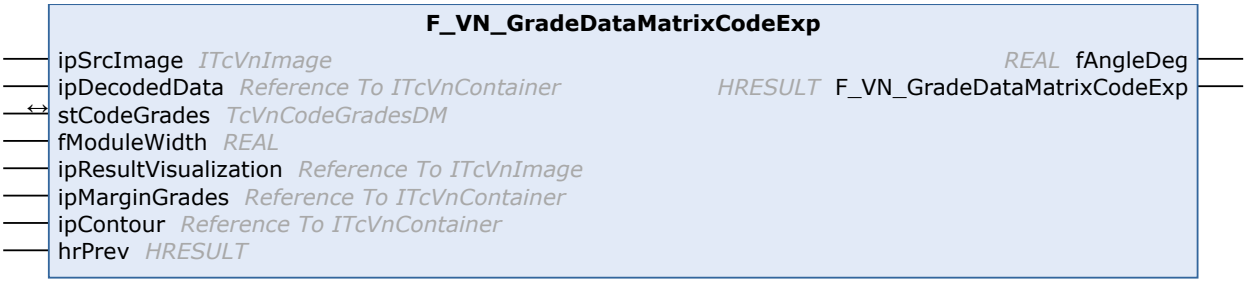
Required License

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 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)

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 [▶ 411]	Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	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 [▶ 411]	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 [▶ 370]	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 [▶ 370]	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesDM [▶ 228]	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 [▶ 141]

Further information

The function `F_VN_GradeDataMatrixCodeExp` is an expert variant of F_VN_GradeDataMatrixCode [▶ 826]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough for the module size. For 2D codes, the smallest module should be covered with at least 5 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGradesDM contains the determined individual results and an overall total result `nOverall`. The quality grades ranges from 0 (very poor) to 4 (very good).

Parameter	Description
nDecode	Evaluates whether the code could be decoded with the reference algorithm.
nSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value in the test area.
nModulation	Evaluates the uniformity of the dark or bright modules.
nReflectanceMargin	Evaluates how well each module can be correctly distinguished as bright or dark compared to the global threshold.
nFixedPatternDamage	Evaluates the various fixed code components such as the finder or alignment pattern.
nAxialNonuniformity	Evaluates the uniform scaling of the code.
nGridNonuniformity	Evaluates deviations of the code grid.
nUnusedErrorCorrection	Evaluates the remaining capacity of the error correction.
nOverall	The overall grade of the code is the lowest grade achieved in the individual examinations.

Minimum module width

The expected minimum module width of the code in pixels is specified with `fModuleWidth`. If the module size in the image differs significantly from this, this can lead to inaccurate or no results and increased computing time.

Presentation of results

`ipResultVisualization` returns an image with the visualization of the scan grid and the respective evaluation results. The modules are color-coded according to the individual grades: `>= 3` as green, `>= 2` as light blue, `>= 1` as orange, `< 1` as red. This parameter is optional and can be set to 0 if it is not required.

Margin information

`ipMarginGrades` returns additional information on the individual margin ratings of each module for a more detailed evaluation. The return vector of the type `ContainerType_Vector_TcVnVector3_REAL` contains the X and Y position of the center points and the grade for each module. This parameter is optional and can be set to 0 if it is not required.

Position of the code

The parameter `ipContour` returns the code corner points as four contour points of the type `ContainerType_Vector_TcVnPoint2_DINT`. The corner points are at the edge of the outer modules. The contour can be drawn directly with F_VN_DrawContours [▶ 1009], for example, or further analyzed with the Contour Analysis [▶ 955] functions. This parameter is optional and can be set to 0 if it is not required.

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in degrees.

Application

The print quality of a Data Matrix Code can be determined in this way, for example:

```
hr := F_VN_GradeDataMatrixCodeExp(
    ipSrcImage              := ipSrcImage,
    ipDecodedData           := ipDecodedData,
    stCodeGrades            := stCodeGrades,
    fModuleWidth            := fModuleWidth,
    ipResultVisualization   := ipResultVisualization,
    ipMarginGrades          := ipMarginGrades,
    ipContour               := ipContour,
    hrPrev                  := hr,
    fAngleDeg               => fAngleDeg);
```

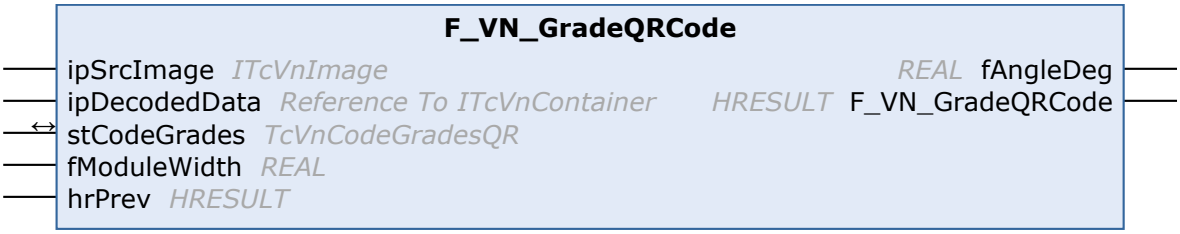
Required License

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 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.

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 [▶ 411]	Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesQR [▶ 229]	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 [▶ 141]

Further information

The `F_VN_GradeQRCode` function determines the print quality of QR codes in accordance with the ISO/IEC 15415:2011 standard.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough for the module size. For 2D codes, the smallest module should be covered with at least 5 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGradesQR contains the determined individual results and an overall total result `nOverall`. The quality grades ranges from 0 (very poor) to 4 (very good).

Parameter	Description
nDecode	Evaluates whether the code could be decoded with the reference algorithm.
nSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value in the test area.
nModulation	Evaluates the uniformity of the dark or bright modules.
nReflectanceMargin	Evaluates how well each module can be correctly distinguished as bright or dark compared to the global threshold.
nFixedPatternDamage	Evaluates the various fixed code components such as the finder or alignment pattern.
nAxialNonuniformity	Evaluates the uniform scaling of the code.
nGridNonuniformity	Evaluates deviations of the code grid.
nUnusedErrorCorrection	Evaluates the remaining capacity of the error correction.
nFormatInfo	Evaluates the format information encoded around the finder pattern.
nVersionInfo	Evaluates the version information contained in the code from version 7 onwards.
nOverall	The overall grade of the code is the lowest grade achieved in the individual examinations.

Minimum module width

The expected minimum module width of the code in pixels is specified with `fModuleWidth`. If the module size in the image differs significantly from this, this can lead to inaccurate or no results and increased computing time.

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in degrees.

Expert parameters

The expert variant F_VN_GradeQRCodeExp [▶ 834] contains additional parameters.

Application

The print quality of a QR code can be determined in this way, for example:

```
hr := F_VN_GradeQRCode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
    fModuleWidth     := fModuleWidth,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

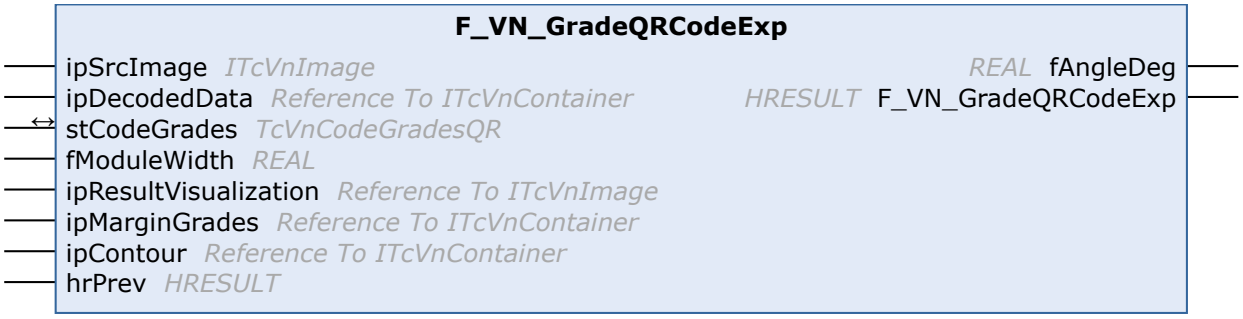
Required License

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 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)

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 [▶ 411]	Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	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 [▶ 411]	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 [▶ 370]	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 [▶ 370]	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesQR [▶ 229]	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 [▶ 141]

Further information

The function `F_VN_GradeQRCodeExp` is an expert variant of F_VN_GradeQRCode [▶ 831]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be an 1-channel image with element type `USINT` (8 bits). The input image may only contain the code to be checked, including the corresponding quiet zones.

To achieve meaningful results, uniform illumination, not underexposed or overexposed, good focusing on the code plane and an image sensor plane aligned parallel to it are required. Furthermore, the image resolution must be high enough for the module size. For 2D codes, the smallest module should be covered with at least 5 pixels in order to achieve good results.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_String_SINT`. Once the code has been successfully found and decoded, the content can be exported to a string using the function `F_VN_ExportContainer_String`.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Different decoding results

Decoding with the respective reference algorithm can lead to deviating results. It is therefore possible that the code content can be read by other devices or functions, but the reference algorithm fails.

Return of the quality assessment

The return value `stCodeGrades` of type TcVnCodeGradesQR contains the determined individual results and an overall total result `nOverall`. The quality grades ranges from 0 (very poor) to 4 (very good).

Parameter	Description
nDecode	Evaluates whether the code could be decoded with the reference algorithm.
nSymbolContrast	Evaluates the global symbol contrast, i.e. the difference between the brightest and darkest value in the test area.
nModulation	Evaluates the uniformity of the dark or bright modules.
nReflectanceMargin	Evaluates how well each module can be correctly distinguished as bright or dark compared to the global threshold.
nFixedPatternDamage	Evaluates the various fixed code components such as the finder or alignment pattern.
nAxialNonuniformity	Evaluates the uniform scaling of the code.
nGridNonuniformity	Evaluates deviations of the code grid.
nUnusedErrorCorrection	Evaluates the remaining capacity of the error correction.
nFormatInfo	Evaluates the format information encoded around the finder pattern.
nVersionInfo	Evaluates the version information contained in the code from version 7 onwards.
nOverall	The overall grade of the code is the lowest grade achieved in the individual examinations.

Minimum module width

The expected minimum module width of the code in pixels is specified with `fModuleWidth`. If the module size in the image differs significantly from this, this can lead to inaccurate or no results and increased computing time.

Presentation of results

`ipResultVisualization` returns an image with the visualization of the scan grid and the respective evaluation results. The modules are color-coded according to the individual grades: `>= 3` as green, `>= 2` as light blue, `>= 1` as orange, `< 1` as red. This parameter is optional and can be set to 0 if it is not required.

Margin information

`ipMarginGrades` returns additional information on the individual margin ratings of each module for a more detailed evaluation. The return vector of the type `ContainerType_Vector_TcVnVector3_REAL` contains the X and Y position of the center points and the grade for each module. This parameter is optional and can be set to 0 if it is not required.

Position of the code

The parameter `ipContour` returns the code corner points as four contour points of the type `ContainerType_Vector_TcVnPoint2_DINT`. The corner points are in the middle of the outer modules. The contour can be drawn directly with F_VN_DrawContours [▶ 1009], for example, or further analyzed with the Contour Analysis [▶ 955] functions. This parameter is optional and can be set to 0 if it is not required.

Rotation angle (return value)

`fAngleDeg` returns the clockwise rotation angle in degrees.

Application

The print quality of a QR code can be determined in this way, for example:

```
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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7 Code Reading

This group contains functions for detecting and reading 1D and 2D codes.

Functions

1D codes

- F_VN_ReadBarcode(Exp) [▶ 841]
  - CodaBar
  - Code39
  - Code93
  - Code128
  - EAN8
  - EAN13
  - ITF
  - UPCA
  - UPCE
  - Code39Extended
- F_VN_ReadPharmaCode(Exp) [▶ 852]

2D codes

- F_VN_ReadDataMatrixCode(Exp) [▶ 847]
- F_VN_ReadQRCode(Exp) [▶ 858]

Code requirements

In general, a module size of at least three pixels is required; more than three pixels are recommended for more stable results. The image should be sharp and have a high contrast. In addition, a code orientation of approx. 0° or 90° is recommended so that the code is parallel to an image axis.

Interpretation of the HRESULT

For the Code Reading functions the HRESULT [▶ 141] return values are used as follows:

Code	Name	Description
16#000	S_OK	Function was successfully executed and all expected codes were found (at least one code for the standard functions and nCodeNumber for the expert functions).
16#001	S_FALSE	Function was executed successfully, not all scan lines could be decoded for 1D codes and not all checks could be performed according to the standard for 2D codes. The overall score was therefore set to 0, but the individual results can be evaluated.
16#256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. Some Code Reading functions can return partial results (see individual function descriptions).
16#7xx	All error codes	Function was not executed successfully. See also: ADS Return Codes [▶ 2879]

Therefore, the usual check with `SUCCEEDED()` is not sufficient to deduce a found code and thus existing data in `ipDecodedData`. The following query can be used for this purpose:

```
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
```

Search strategies for 2D codes

For 2D codes the search algorithms are applied to different variants of the input image in order to recognize as many codes as possible. Both inversion and mirroring of the image are used to generate various versions. The decision as to which variant is tested is called the search strategy. So that you have greater control over the runtime of the functions, this search strategy can be configured. The enum ETcVn2dCodeSearchStrategy [▶ 162] is available for this purpose. With this you can specify for each possible image transformation (inversion and mirroring) whether it is to be applied and whether the original image or the transformed image should be checked first for codes.

The function parameter `eSearchStrategy` defines precisely this search strategy for corresponding code reading expert functions. The tables below show the different combinations:

Table 1: Inversion

eSearchStrategy	Original image	Inverted image
TCVN_CSS_ONLY_NOT_INVERTED	1.	-
TCVN_CSS_FIRST_NOT_INVERTED	1.	2.
TCVN_CSS_ONLY_INVERTED	-	1.
TCVN_CSS_FIRST_INVERTED	2.	1.

Table 2: Mirroring

eSearchStrategy	Original image	Mirrored image
TCVN_CSS_ONLY_NOT_FLIPPED	1.	-
TCVN_CSS_FIRST_NOT_FLIPPED	1.	2.
TCVN_CSS_ONLY_FLIPPED	-	1.
TCVN_CSS_FIRST_FLIPPED	2.	1.

A search strategy can be defined for each type of transformation. The search strategies of the different search strategies can be linked as follows:

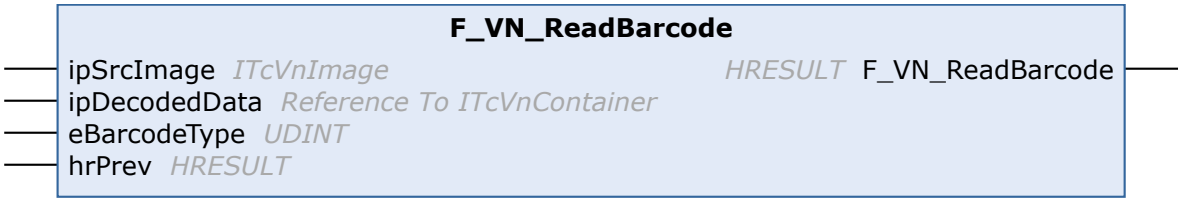
```
eSearchStrategy := TCVN_CSS_ONLY_FLIPPED + TCVN_CSS_FIRST_INVERTED;
```

The `TCVN_CSS_DEFAULT` option can be used to select a default setting that varies depending on the code type.

Linking of search strategies not possible

The linking of several search strategies for the same transformation type is not permissible. Only one strategy for inversion and one for mirroring can be linked to each other.

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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
eBarcodeType	UDINT	Types of barcode to search for (ETcVnBarcodeType)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ReadBarcode` detects a barcode in the image and reads it.

Algorithm

The function first searches for horizontally aligned codes and then for vertically aligned codes in the image.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image with the element type `USINT` (8-bit). 3-channel images will be interpreted as RGB and converted internally into grayscale images.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_Vector_String_SINT`.

Barcode type

The parameter `eBarcodeType` defines the barcode types to be searched for in the image. All barcode types in the enum ETcVnBarcodeType [▶ 164] are supported. It is possible to select more than one type. It is recommended to restrict the selection to the types actually occurring in the end application in order to improve the runtime.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Use `TCVN_BT_ANY` to search for all available barcode types.

Expert parameters

The expert version F_VN_ReadBarcodeExp [▶ 841] contains additional parameters.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function F_VN_ExportSubContainer_String [▶ 760].

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

Samples

- EAN-13 Barcode Reading [▶ 2759]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadBarcode [▶ 839]
- F_VN_ReadPharmaCode [▶ 850]
- F_VN_ReadDataMatrixCode [▶ 845]
- F_VN_ReadQRCode [▶ 855]

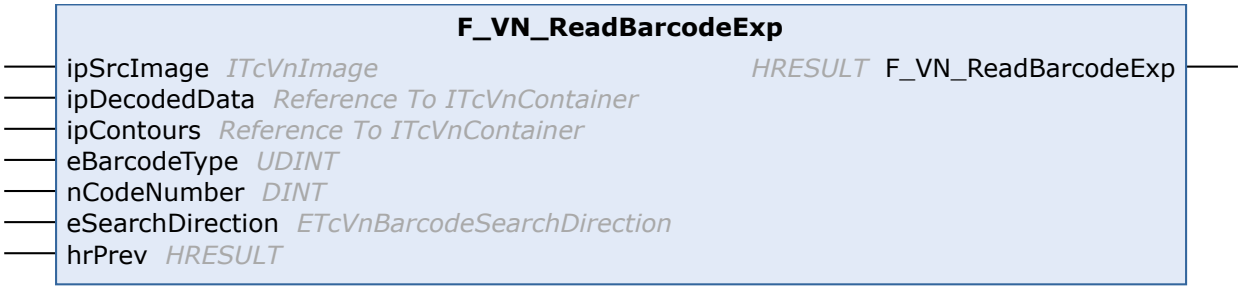
Required License

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 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 164]	Barcode search direction.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ReadBarcodeExp` is the expert version of F_VN_ReadBarcode [▶ 839]. It contains additional parameters for returning the code position and for setting the search direction.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image with the element type `USINT` (8-bit). 3-channel images will be interpreted as RGB and converted internally into grayscale images.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_Vector_String_SINT`.

Barcode type

The parameter `eBarcodeType` defines the barcode types to be searched for in the image. All barcode types in the enum ETcVnBarcodeType [▶ 164] are supported. It is possible to select more than one type. It is recommended to restrict the selection to the types actually occurring in the end application in order to improve the runtime.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Use `TCVN_BT_ANY` to search for all available barcode types.

Position of the code

The parameter `ipContour` returns the code position that is found as a contour.

Number of codes

The parameter `nCodeNumber` defines how many codes to search for.

Search for more than one barcode not supported

Searching for several barcodes in one image is currently not supported. The parameter nCodeNumber therefore has to be set to 1.

Search direction

The search direction can be specified via `eSearchDirection`:

- `TCVN_BSD_ANY` first searches in horizontal direction, then in vertical direction.
- `TCVN_BSD_HORIZONTAL` only searches in horizontal direction.
- `TCVN_BSD_VERTICAL` only searches in vertical direction.

Samples

- EAN-13 Barcode Reading [▶ 2759]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadBarcode [▶ 839]
- F_VN_ReadPharmaCode [▶ 850]
- F_VN_ReadDataMatrixCode [▶ 845]
- F_VN_ReadQRCode [▶ 855]

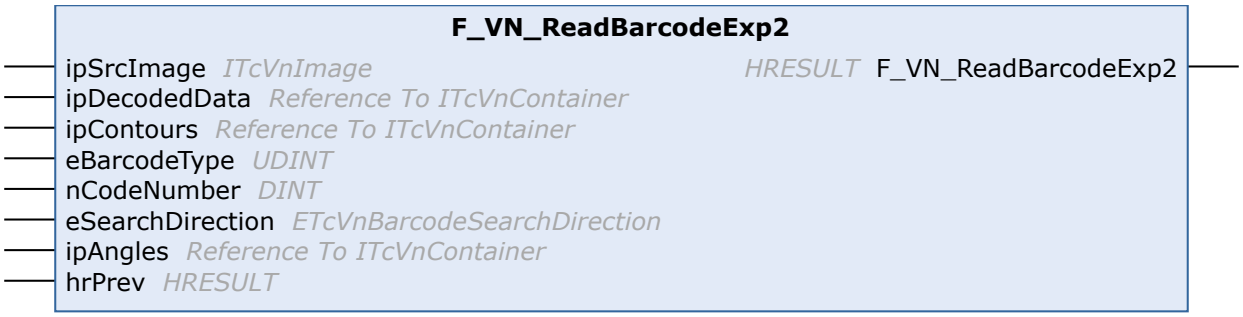
Required License

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 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 164]	Barcode search direction.
ipAngles	Reference To ITcVnContainer [▶ 370]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ReadBarcodeExp` is the expert variant of F_VN_ReadBarcode [▶ 839]. It contains additional parameters for returning the code position and for setting the search direction.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image with the element type `USINT` (8-bit). 3-channel images will be interpreted as RGB and converted internally into grayscale images.

Read data

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_Vector_String_SINT`.

Barcode type

The parameter `eBarcodeType` defines the barcode types to be searched for in the image. All barcode types in the enum ETcVnBarcodeType [▶ 164] are supported. It is possible to select more than one type. It is recommended to restrict the selection to the types actually occurring in the end application in order to improve the runtime.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Use `TCVN_BT_ANY` to search for all available barcode types.

Position of the code

The parameter `ipContour` returns the code position that is found as a contour.

Number of codes

The parameter `nCodeNumber` defines how many codes are to be searched for.

Search for more than one barcode not supported

Searching for several barcodes in one image is currently not supported. The parameter nCodeNumber := 1 must therefore be set.

Search direction

The search direction can be specified via `eSearchDirection`:

- With `TCVN_BSD_ANY`, the search is first carried out in the horizontal direction and then in the vertical direction.
- With `TCVN_BSD_HORIZONTAL`, the search is only carried out in a horizontal direction.
- With `TCVN_BSD_VERTICAL`, the search is only carried out in the vertical direction.

Angle of rotation

The parameter `ipAngles` returns the rotation angle of the codes found in a clockwise direction in 90° steps. The container is of the type `ContainerType_Vector_REAL`. This parameter is optional and can be set to 0 if it is not required.

Samples

- EAN-13 Barcode Reading [▶ 2759]

- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadBarcode [▶ 839]
- F_VN_ReadPharmaCode [▶ 850]
- F_VN_ReadDataMatrixCode [▶ 845]
- F_VN_ReadQRCode [▶ 855]

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 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 [▶ 411]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

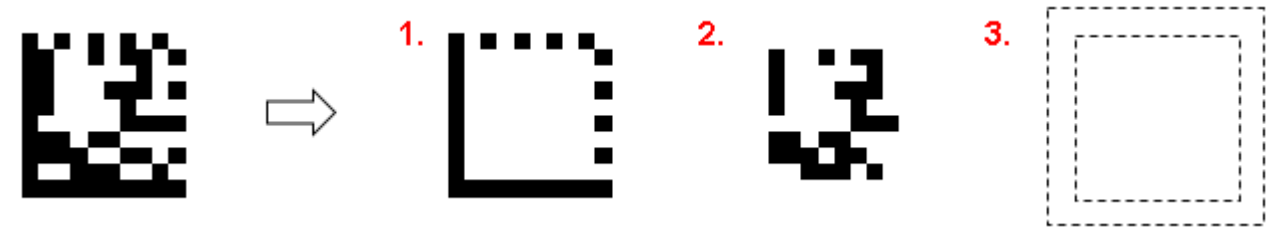
Further information

The data matrix barcode according to IEC 16022 is a 2D code that facilitates storage of large data quantities with minimum space requirements. It is therefore often used for component and product marking.

The function F_VN_ReadDataMatrixCode can only recognize and decode ECC 200 code types in the image. This is based on one code per image. If several Data Matrix Codes are to be recognized and decoded in one image, the function F_VN_ReadDataMatrixCodeExp [▶ 847] must be used.

Data Matrix Code structure

The Data Matrix Code consists of rectangles or dots that make up a square or rectangular symbol, which in turn can be divided into three areas:



1. Finder Pattern
   The solid-line locator (L) is used to locate and align the code, while the pattern-line locator (P) is used to determine the cell size and arrangement.
2. Data area
   The data area is surrounded by the Finder Pattern.
3. Quiet zone
   The quiet zone is an area at least 1.5 cells wide around the Finder Pattern in the background color of the code.

Flaws / damage in the Finder Pattern or in the quiet zone can prevent detection of the code.

Error correction

Data Matrix Code ECC 200 (ECC = Error Correction Code) is the generally accepted version due to its better error correction. In addition to the actual data, they are coded redundantly based on the Reed-Solomon code. Data matrix codes of this version can be decoded by the function. In case of damage an error correction can be performed.

Code search

The function F_VN_ReadDataMatrixCode searches for a Data Matrix Code in the image, starting from the center line of the image.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function F_VN_ExportSubContainer_String [▶ 760].

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

HRESULT

Like all TwinCAT Vision API functions, this function also returns an HRESULT [▶ 141] to indicate whether the execution was successful or not. In the event of success, a distinction can be made between the following success codes in the subsequent handling of the results. To distinguish between these cases, the `HRESULT` variable can be compared directly with `S_OK` or `S_FALSE`.

Code	Name	Description
16#000	S_OK	The function was executed successfully. The specified number of codes was found in the image.
16#001	S_FALSE	The function was executed successfully. Fewer codes than indicated in the image were found.
16#256	S_WATCHDOG TIMEOUT	The function was aborted by the watchdog [▶ 146].

Examples

- Data Matrix Code Reading [▶ 2763]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]
- F_VN_ReadQRCode [▶ 855] or F_VN_ReadQRCodeExp [▶ 858]

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 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 [▶ 411]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function F_VN_ReadDataMatrixCodeExp corresponds to the function F_VN_ReadDataMatrixCode [▶ 845] extended by the following parameters.

ipContours

The parameter `ipContour` returns the code positions that are found as a contour.

nCodeNumber

The parameter `nCodeNumber` defines how many codes to search for. At *-1* the search continues until no further code is found in the image or a watchdog aborts the function.

eSearchStrategy

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategy `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [▶ 838].

Search strategies

Currently, this function only supports the settings of the search strategy for mirroring. Inversion is always performed automatically if not all expected codes are found in the original image.

Samples

- Data Matrix Code Reading [▶ 2763]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadDataMatrixCode [▶ 845]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]
- F_VN_ReadQRCode [▶ 855] or F_VN_ReadQRCodeExp [▶ 858]

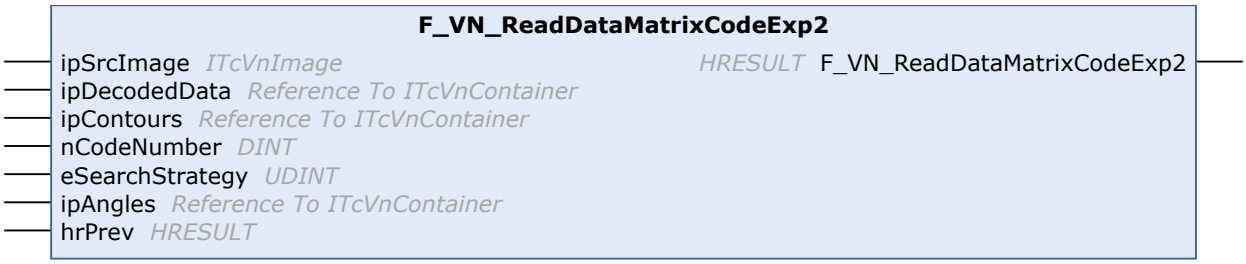
Required License

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 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 [▶ 411]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function F_VN_ReadDataMatrixCodeExp corresponds to the function F_VN_ReadDataMatrixCode [▶ 845] extended by the following parameters.

ipContours

The parameter `ipContour` returns the code positions that are found as a contour.

nCodeNumber

The parameter `nCodeNumber` defines how many codes are to be searched for. At *-1* the search continues until no further code is found in the image or a watchdog aborts the function.

eSearchStrategy

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategy `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [▶ 838].

Search strategies

Currently, this function only supports the settings of the search strategy for mirroring. Inversion is always performed automatically if not all expected codes are found in the original image.

Angle of rotation

The parameter `ipAngles` returns the clockwise rotation angle of the codes found in degree steps. The container is of the type `ContainerType_Vector_REAL`. This parameter is optional and can be set to 0 if it is not required.

Examples

- Data Matrix Code Reading [▶ 2763]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadDataMatrixCode [▶ 845]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]
- F_VN_ReadQRCode [▶ 855] or F_VN_ReadQRCodeExp [▶ 858]

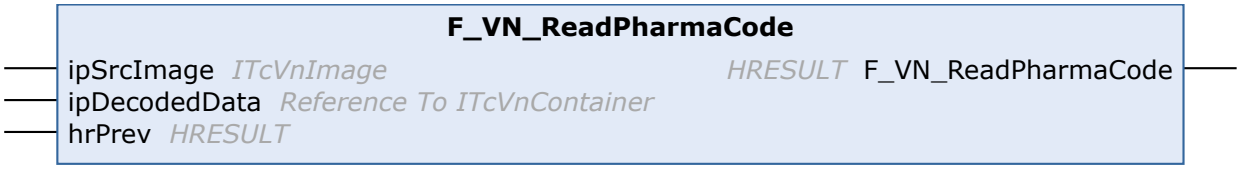
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.7 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ReadPharmaCode` detects and reads a pharmacode in the input image `ipSrcImage`. The Pharmacode is a barcode standard. It is designed for good legibility but can only encode integers in the range 3 to 131070. The aim of this standard is high reading speed with few misreadings.

Do not mix up pharmacodes

The Pharmacode Standard should not be confused with the Italian or Swiss pharmacodes. These only designate drug identification numbers with a Code39 representation.

Structure of Pharmacodes

A Pharmacode consists of thick and thin bars (dark) and gaps between them (light). For good legibility, sufficient size and contrast of the code in the image are recommended. Decoding is based on the following table:

Position	7	6	5	4	3	2	1	0
Thin line	128	64	32	16	8	4	2	1
Thick line	256	128	64	32	16	8	4	2

Detailed information on the specification of Pharmacodes can be found online: Pharmacode Guide from Laetus

Parameter

Input image

The input image `ipSrcImage` must have 1 or 3 channels. 3-channel images must be of the type `USINT`, `UINT` or `REAL` and must be in the RGB color space. They are converted internally into a grayscale image.

Read data (Return value)

The data read from the code are returned as a string in the container `ipDecodedData`. The container is of the type `ContainerType_Vector_String_SINT`.

Expert parameters

The expert version F_VN_ReadPharmaCodeExp [▶ 852] contains additional parameters.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function F_VN_ExportSubContainer_String [▶ 760].

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

Samples

- Pharma-Code Reading [▶ 2764]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadPharmaCode [▶ 850] for reading pharmacodes
- F_VN_ReadBarcode [▶ 839] for reading barcodes
- F_VN_ReadDataMatrixCode [▶ 845] for reading data matrix codes
- F_VN_ReadQRCode [▶ 855] for reading QR codes

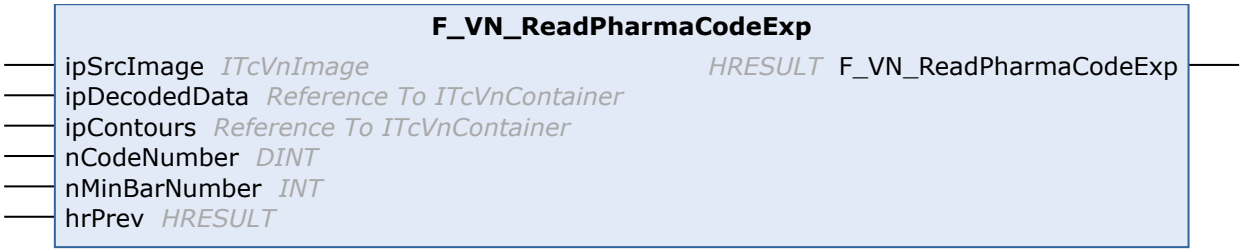
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.8 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This Expert function extends the F_VN_ReadPharmaCode [▶ 850] function as follows:

- Return of the position of found codes via the parameter `ipContours`
- Specifies the number of codes to be found in the image. The search is terminated once `nCodeNumber` or more codes have been found in the image. If less than `nCodeNumber` codes are found, the function returns `S_FALSE`.
- Minimum number of dark bars that the code must have to be found by the function via the parameter `nMinBarNumber`.

Samples

- Pharma-Code Reading [▶ 2764]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadPharmaCode [▶ 850]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 842]
- F_VN_ReadDataMatrixCode [▶ 845] or F_VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadQRCode [▶ 855] or F_VN_ReadQRCodeExp [▶ 858]

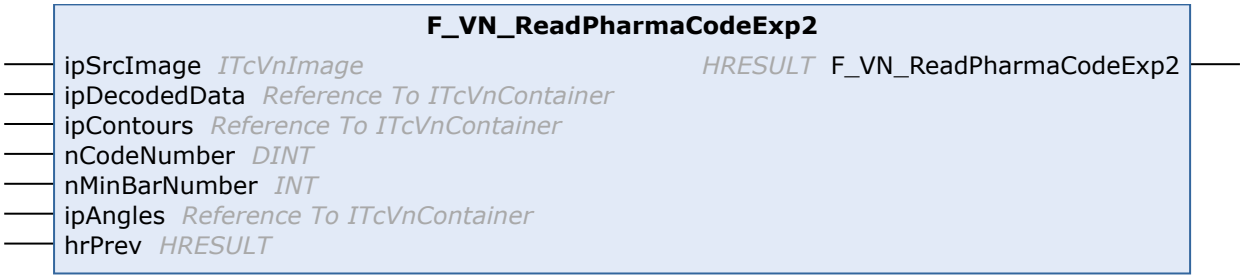
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.9 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This Expert function extends the function F_VN_ReadPharmaCode [▶ 850] as follows:

- Return of the position of found codes via the parameter `ipContours`
- Specifies the number of codes to be found in the image. The search is terminated once `nCodeNumber` or more codes have been found in the image. If less than `nCodeNumber` codes are found, the function returns `S_FALSE`.

- Minimum number of dark bars that the code must have to be found by the function via the parameter `nMinBarNumber`.

Angle of rotation

The parameter `ipAngles` returns the approximate rotation angle of the codes found in a clockwise direction from -90° to +90° in degrees. The container is of the type `ContainerType_Vector_REAL`. This parameter is optional and can be set to 0 if it is not required.

Differences in the reading direction

As a Pharmacode does not contain a start or stop identifier, the result varies depending on the reading direction. Therefore, whether the code has been rotated 180° cannot be determined from the rotation angle. To ensure that the code is in the expected orientation, the image must be rotated by 180° and the code read again. Both read results can then be compared with an expected value in order to clearly determine the orientation.

Examples

- Pharma-Code Reading [▶ 2764]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadPharmaCode [▶ 850]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 842]
- F_VN_ReadDataMatrixCode [▶ 845] or VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadQRCode [▶ 855] or F_VN_ReadQRCodeExp [▶ 858]

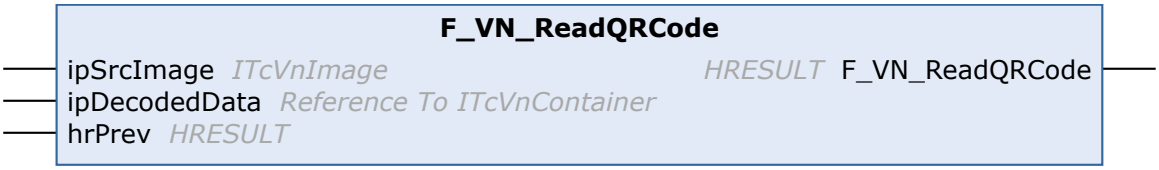
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.10 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The QR code (Quick-Response code) according to IEC 18004 is a 2D code, which is particularly common in Germany in the field of mobile tagging. Thanks to the option to encode Kanji/Kana characters, the QR code has also become established in the industry in Asia.

The F_VN_ReadQRCode function can recognize and decode this code type in the image. This is based on one code per image.

QR code structure



1.		Position markings
2.		Alignment mark
3.		Timing rows
4.		Version information
5.		Data format
6.		Data field
7.		Quiet zone, at least 4 cells wide

Flaws / damage can prevent detection and decoding.

Code search

The function F_VN_ReadQRCode searches line by line for a QR code in the image from top left to bottom right.

Read data from container

If the code was successfully found and decoded, the code content is returned in the container `ipDecodedData` and can be exported to a string using the function F_VN_ExportSubContainer_String [▶ 760].

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

HRESULT

Like all TwinCAT Vision API functions, this function also returns an HRESULT [▶ 141] to indicate whether the execution was successful or not. In the event of success, a distinction can be made between the following success codes in the subsequent handling of the results. To distinguish between these cases, the `HRESULT` variable can be compared directly with `S_OK` or `S_FALSE`.

Code	Name	Description
16#000	S_OK	The function was executed successfully. The specified number of codes was found in the image.
16#001	S_FALSE	The function was executed successfully. Fewer codes than indicated in the image were found.
16#256	S_WATCHDOG TIMEOUT	The function was aborted by the watchdog [▶ 146].

Samples

- QR-Code Reading [▶ 2766]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadQRCodeExp [▶ 858]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadDataMatrixCode [▶ 845] or F_VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]

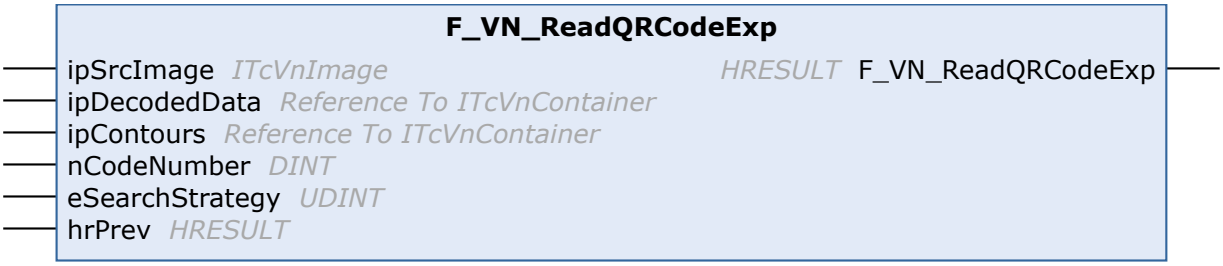
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.11 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

NOTICE

Floating point exceptions

This function can cause errors if the option **Floating point exceptions** of the executing PLC task [▶ 61] is active. Therefore, deactivate this option.

The function F_VN_ReadQRCodeExp corresponds to the function F_VN_ReadQRCode [▶ 855], extended by the following parameters.

ipContour

The parameter `ipContour` returns the code position that is found as a contour.

nCodeNumber

The parameter `nCodeNumber` defines how many codes to search for.

> **Searching for more than one QR code not currently supported**
>
> Searching for several QR codes in one image is currently not supported. The parameter nCodeNumber therefore has to be set to 1.

eSearchStrategy

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategies `TCVN_CSS_FIRST_NOT_INVERTED` and `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [▶ 838].

Samples

- QR-Code Reading [▶ 2766]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadQRCode [▶ 855]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadDataMatrixCode [▶ 845] or F_VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]

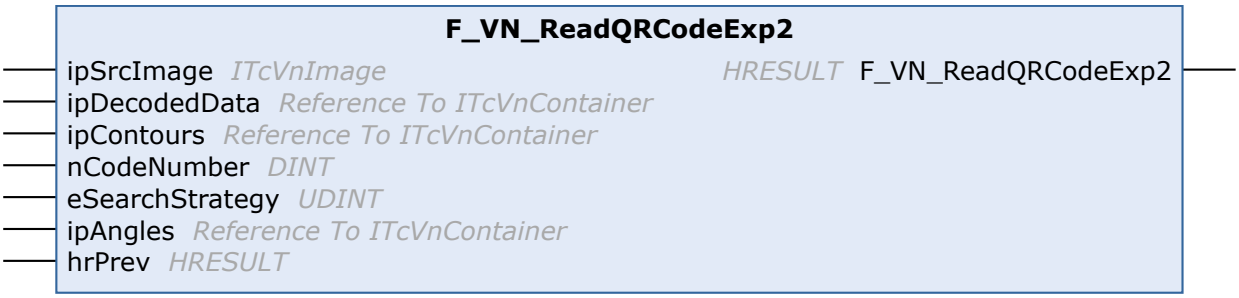
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.12 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 [▶ 411]	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 [▶ 370]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

NOTICE

Floating point exceptions

This function can cause errors if the option **Floating point exceptions** of the executing PLC task [▶ 61] is active. Therefore, deactivate this option.

The function F_VN_ReadQRCodeExp corresponds to the function F_VN_ReadQRCode [▶ 855], extended by the following parameters.

ipContour

The parameter `ipContour` returns the code position that is found as a contour.

nCodeNumber

The parameter `nCodeNumber` defines how many codes are to be searched for.

Searching for several QR codes not currently supported

Searching for several QR codes in one image is currently not supported. The parameter nCodeNumber := 1 must therefore be set.

eSearchStrategy

The parameter `eSearchStrategy` defines the search strategies for the code search. The standard option `TCVN_CSS_DEFAULT` leads to the search strategies `TCVN_CSS_FIRST_NOT_INVERTED` and `TCVN_CSS_FIRST_NOT_FLIPPED`. For a description of the search strategies see Code Reading search strategies [▶ 838].

Angle of rotation

The parameter `ipAngles` returns the clockwise rotation angle of the codes found in degrees. The container is of the type `ContainerType_Vector_REAL`. This parameter is optional and can be set to 0 if it is not required.

Examples

- QR-Code Reading [▶ 2766]
- Result evaluation during Code Reading [▶ 2768]

Related functions

- F_VN_ReadQRCode [▶ 855]
- F_VN_ReadBarcode [▶ 839] or F_VN_ReadBarcodeExp [▶ 841]
- F_VN_ReadDataMatrixCode [▶ 845] or VN_ReadDataMatrixCodeExp [▶ 847]
- F_VN_ReadPharmaCode [▶ 850] or F_VN_ReadPharmaCodeExp [▶ 852]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8 Container Statistics

This group contains functions for statistical analysis of containers [▶ 152].

Functions

With Euclidean norm

- F_VN_ContainerAverage [▶ 943]
- F_VN_ContainerAverageVariance [▶ 947]
- F_VN_MaxElement [▶ 952]
- F_VN_MedianElement [▶ 953]
- F_VN_MinElement [▶ 954]

Element-wise

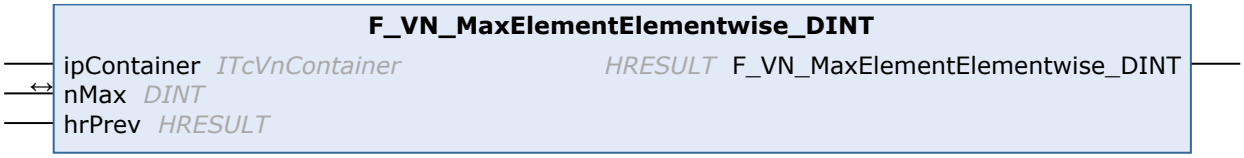
- F_VN_ContainerAverageElementwise [▶ 944]
- F_VN_ContainerAverageVarianceElementwise [▶ 948]
- F_VN_MaxElementElementwise [▶ 861]
- F_VN_MedianElementElementwise [▶ 889]
- F_VN_MinElementElementwise [▶ 915]

6.1.4.8.1 F_VN_MaxElementElementwise

This chapter contains functions with which the max value is determined according to the data type for each element. This function is available for containers with basic elements.

For example, with a TcVnPoint2_DINT, the maximum X-value and the maximum Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

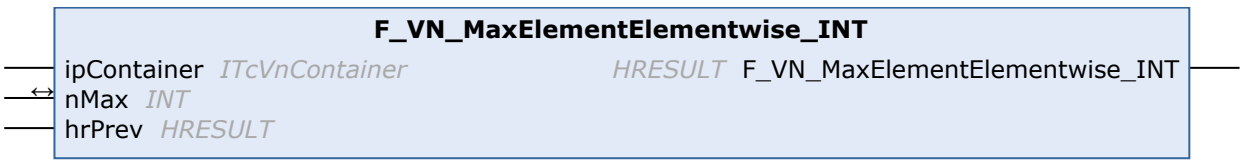
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

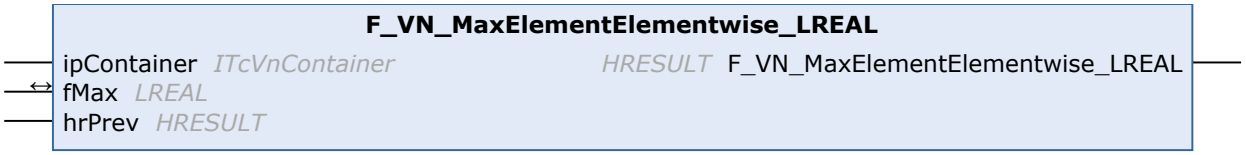
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

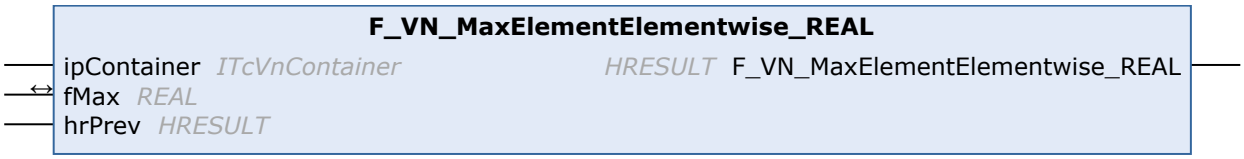
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

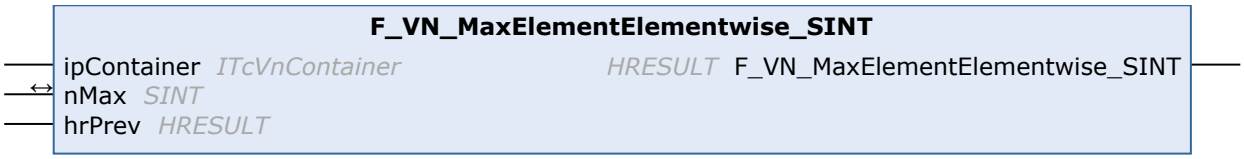
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

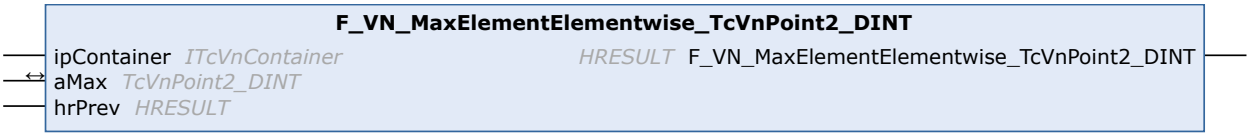
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_DINT [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

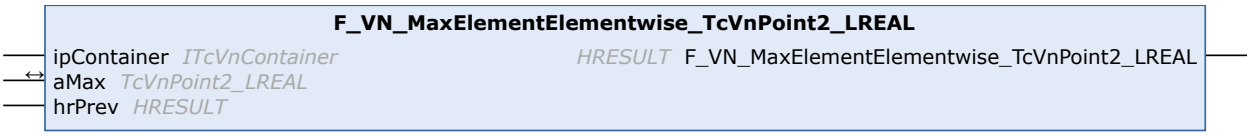
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

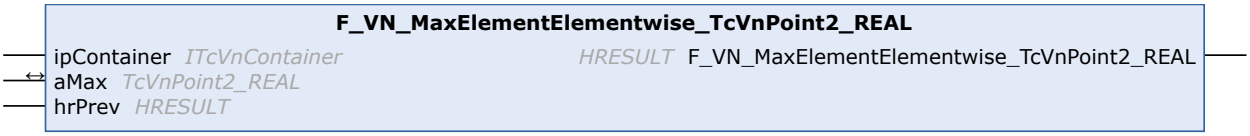
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

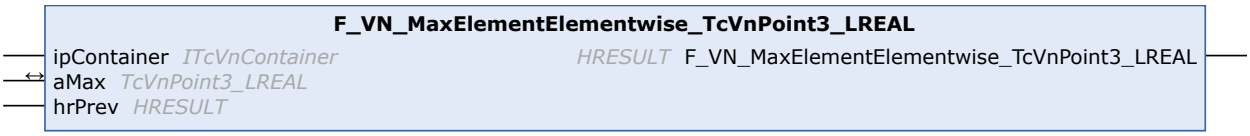
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint3_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

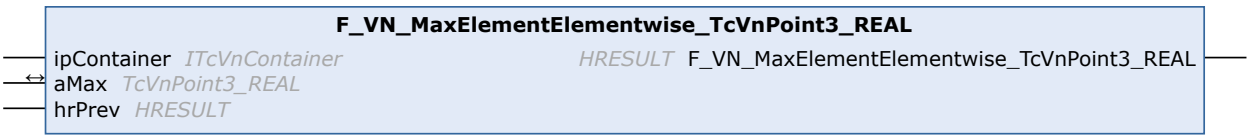
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint3_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

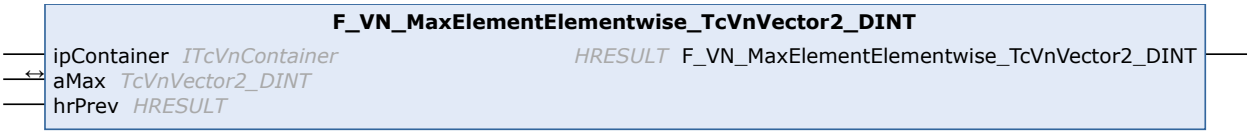
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

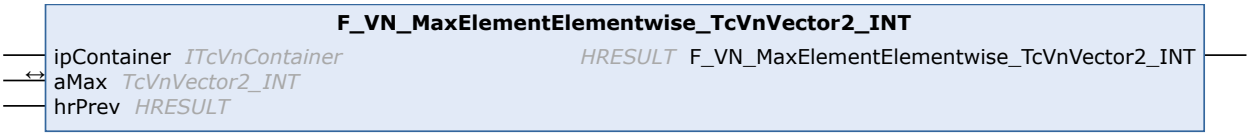
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

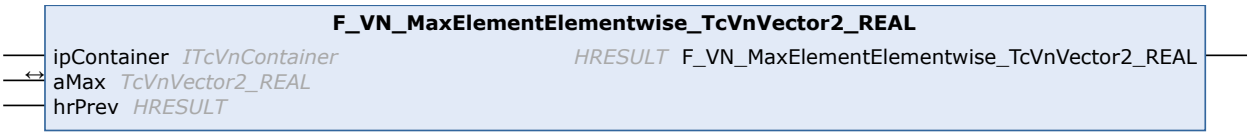
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

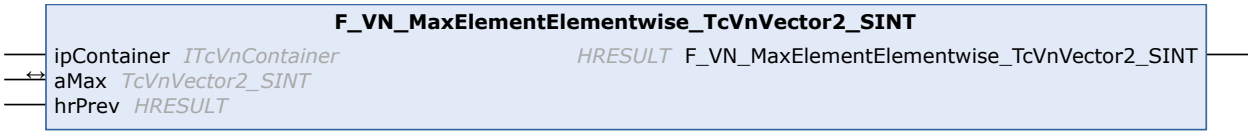
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

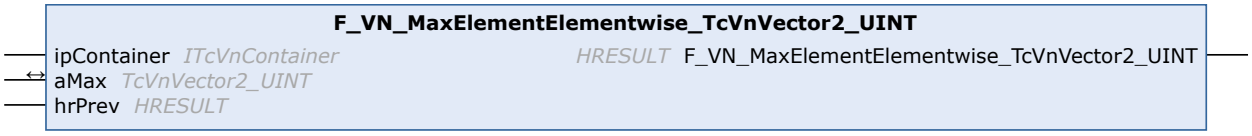
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

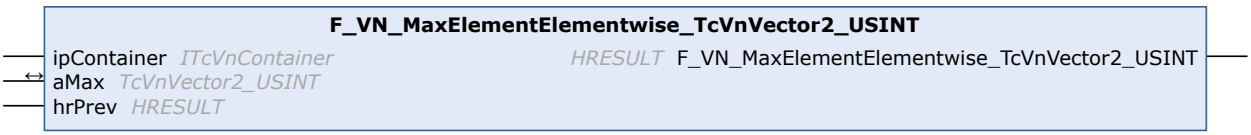
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

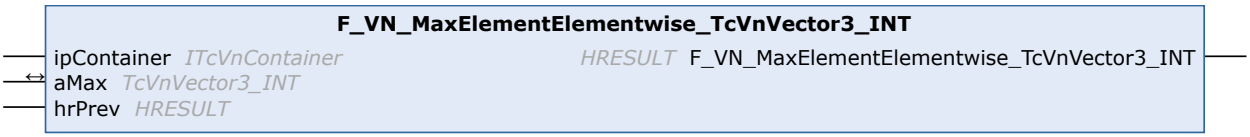
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

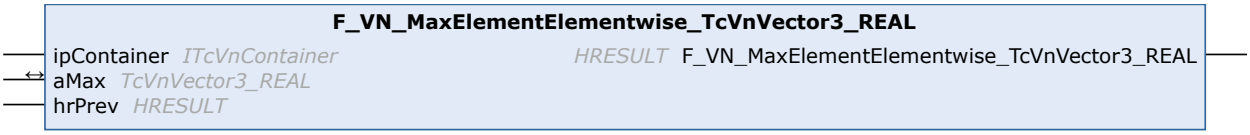
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

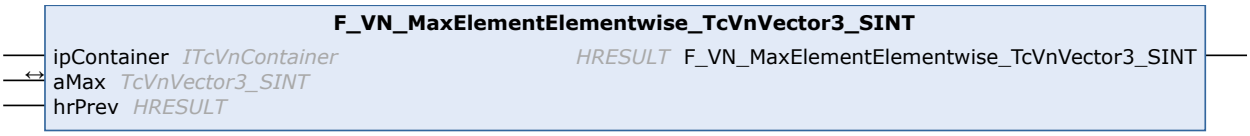
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

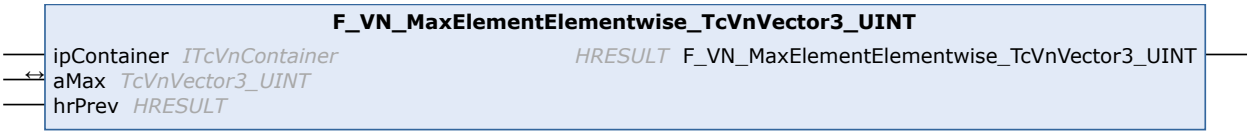
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

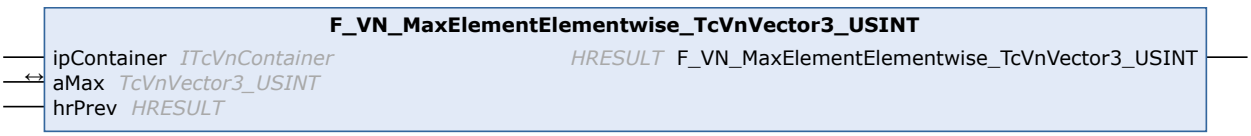
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

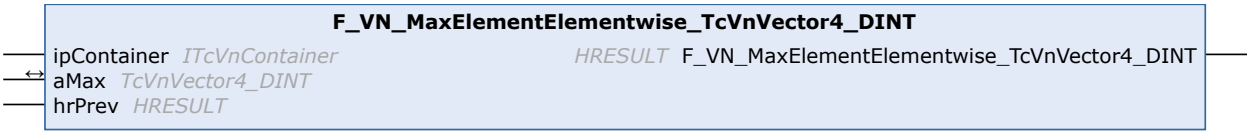
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

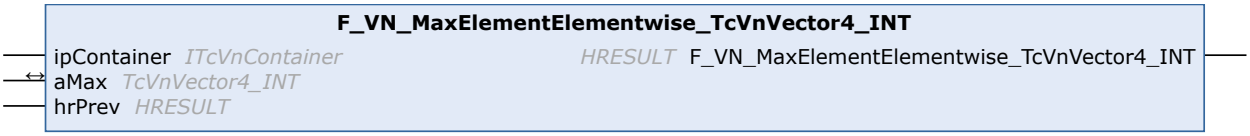
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

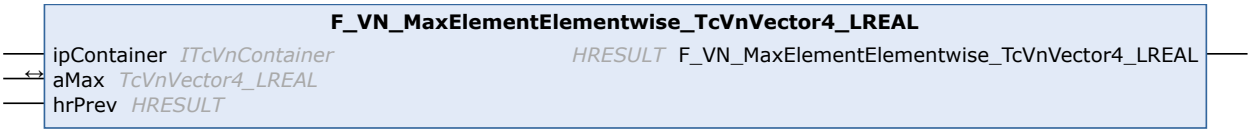
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_LREAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

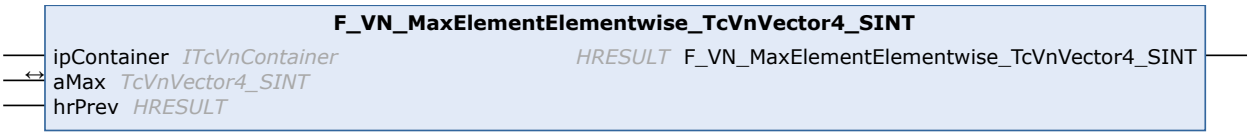
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

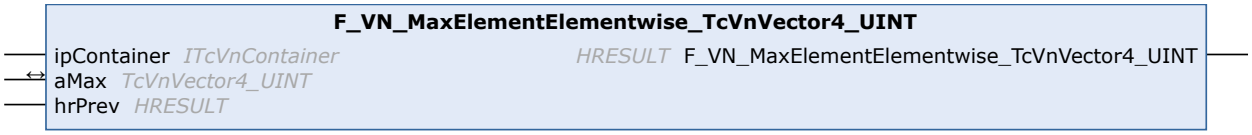
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

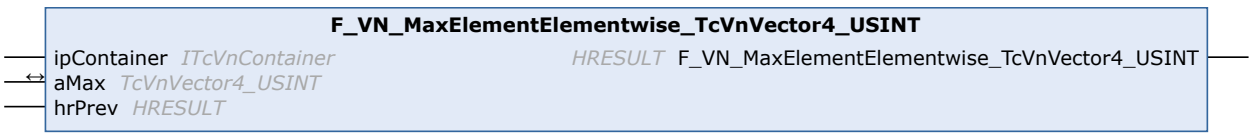
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

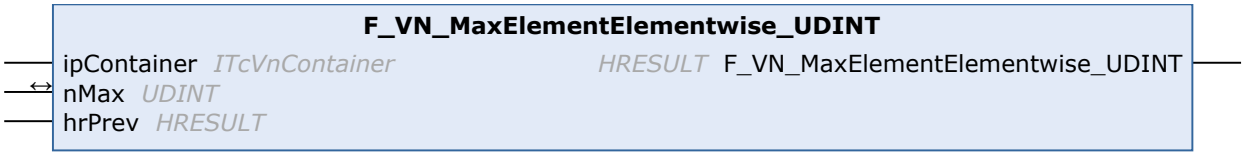
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

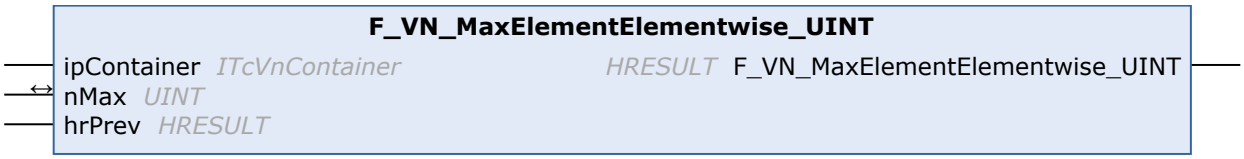
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

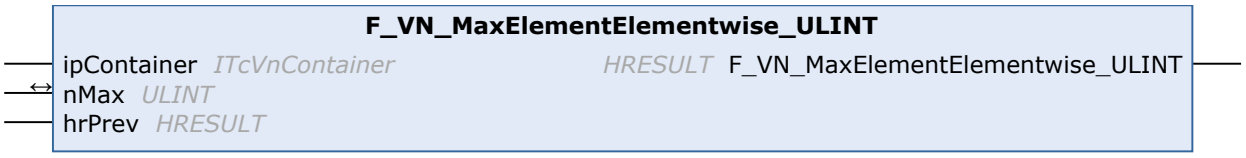
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

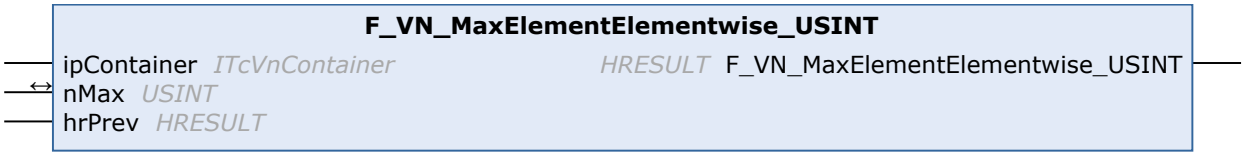
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

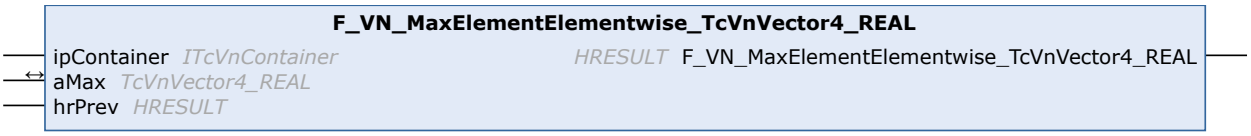
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

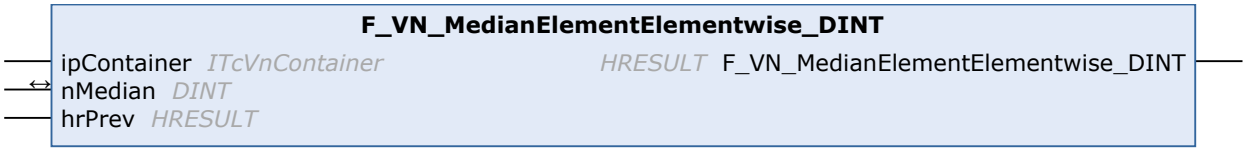
Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2 F_VN_MedianElementElementwise

This chapter contains functions to determine the median value according to the data type for each element. This function is available for containers with basic elements.

For example, for a TcVnPoint2_DINT, the median X-value and the median Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

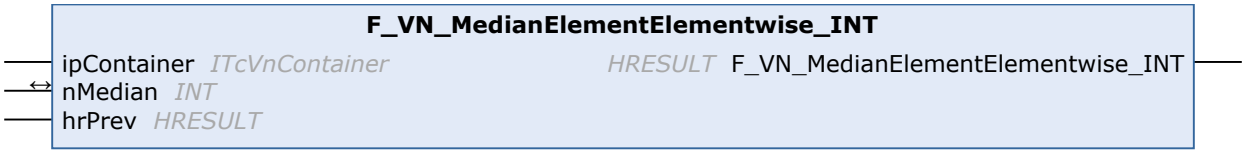
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

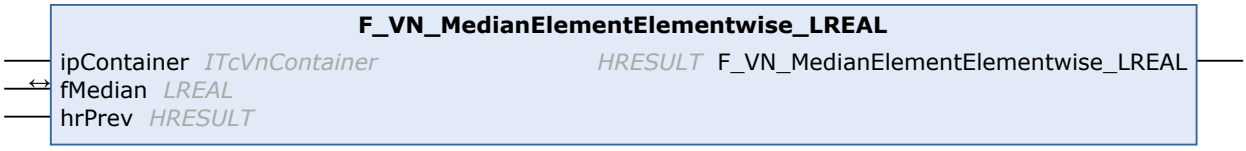
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

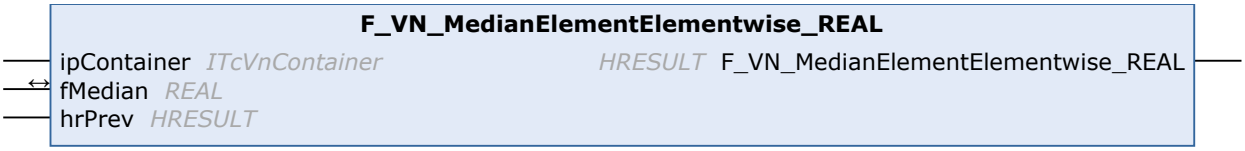
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

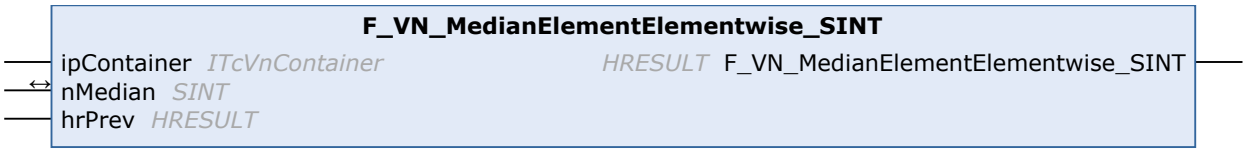
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

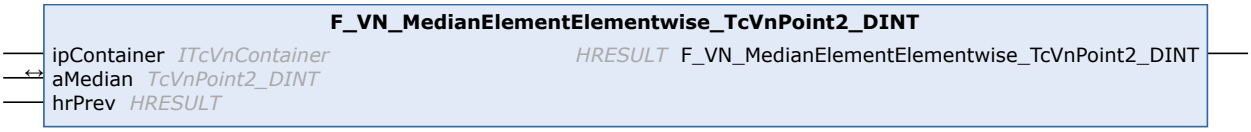
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_DINT [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

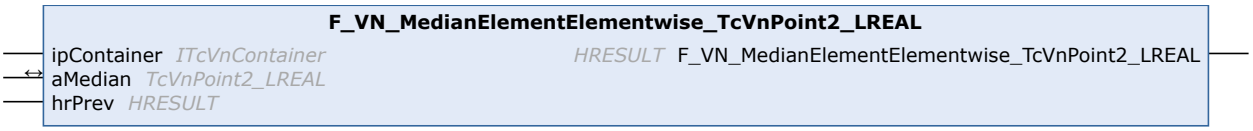
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

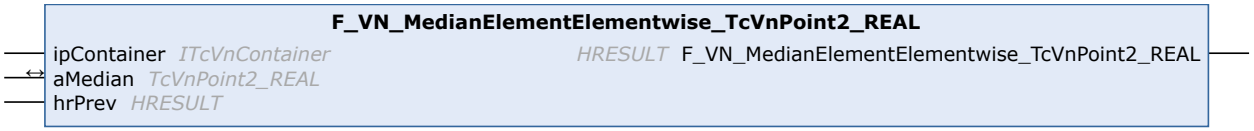
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

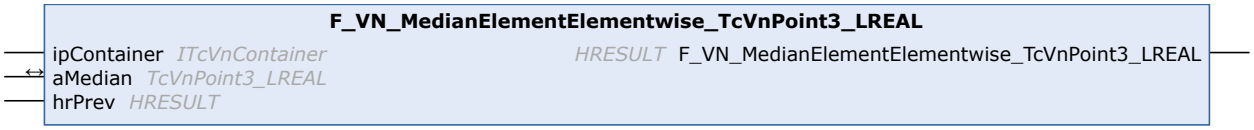
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint3_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

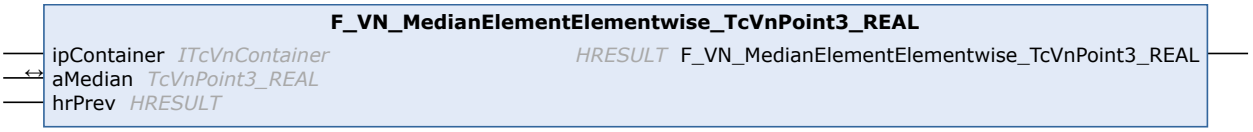
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint3_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

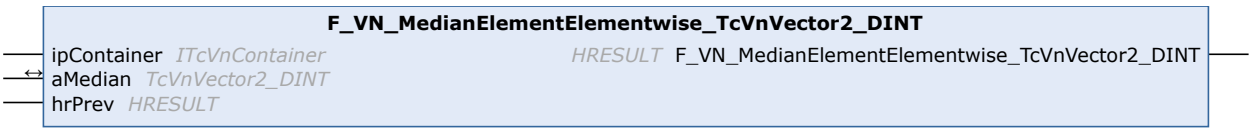
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

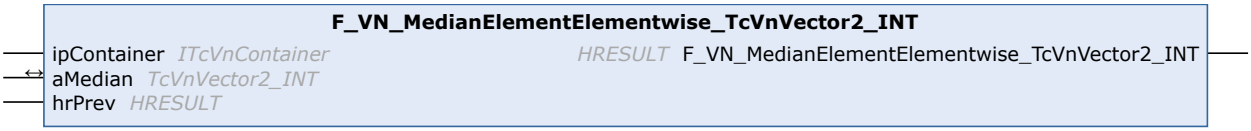
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

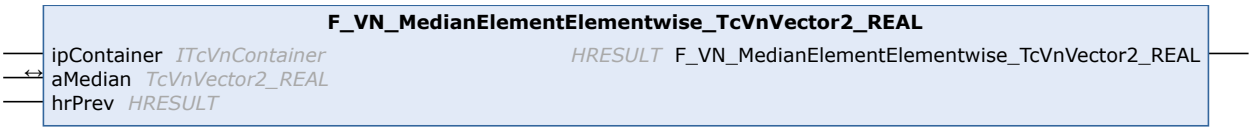
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

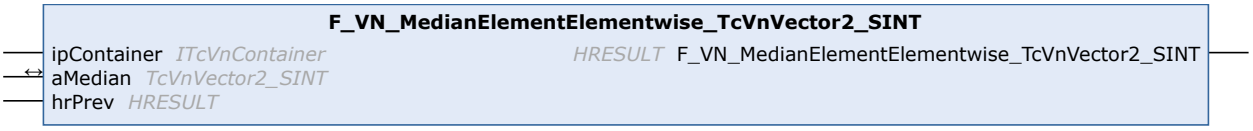
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

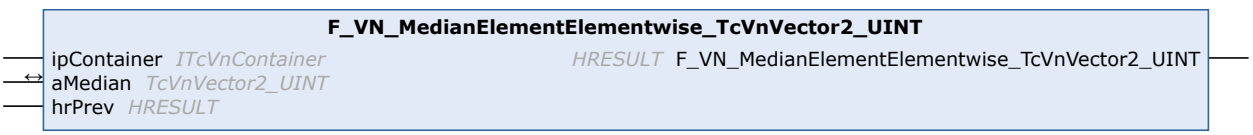
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

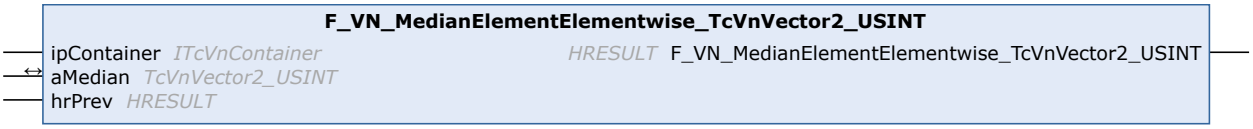
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

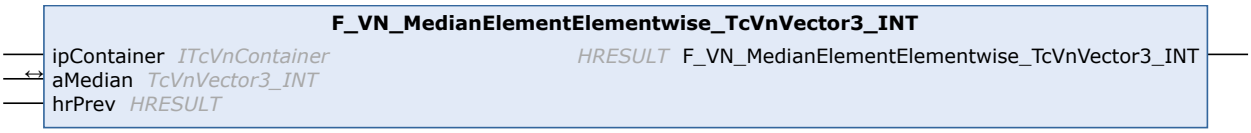
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

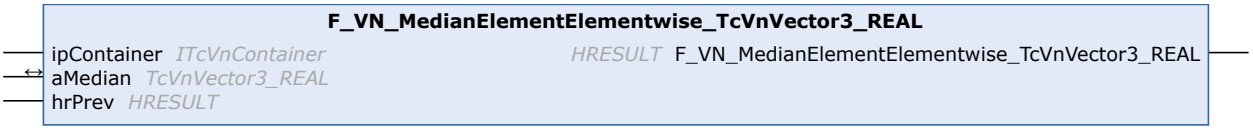
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

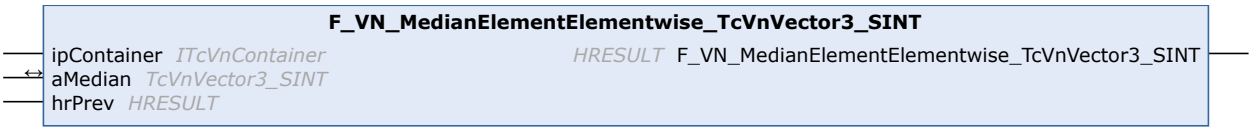
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

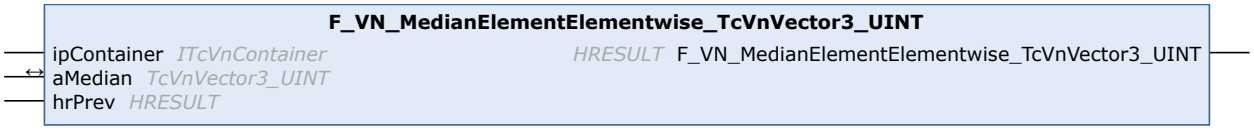
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

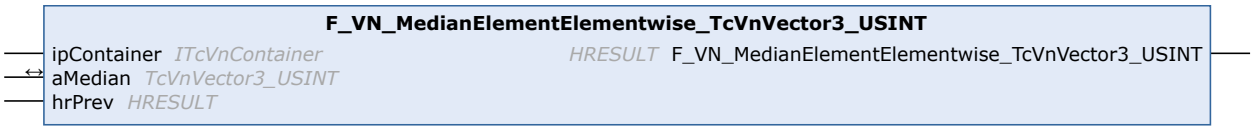
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

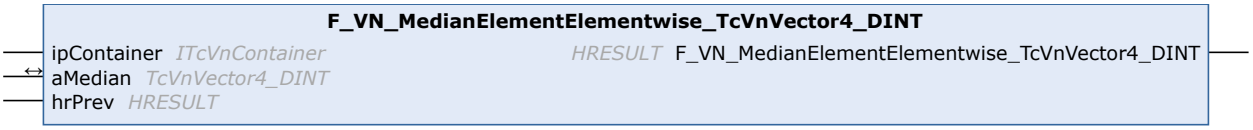
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

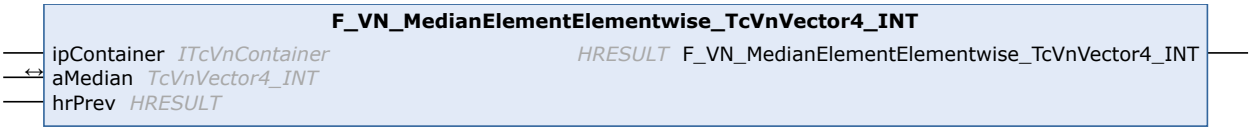
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

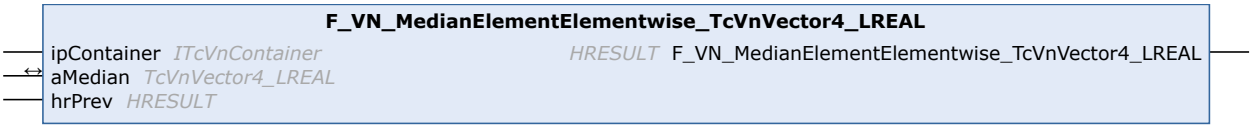
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

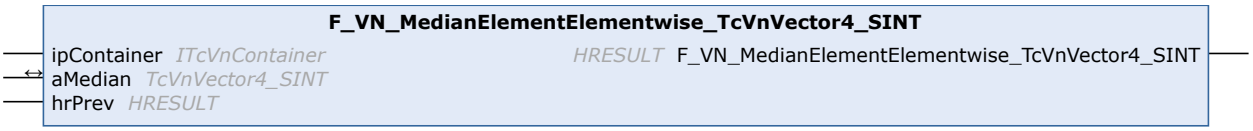
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

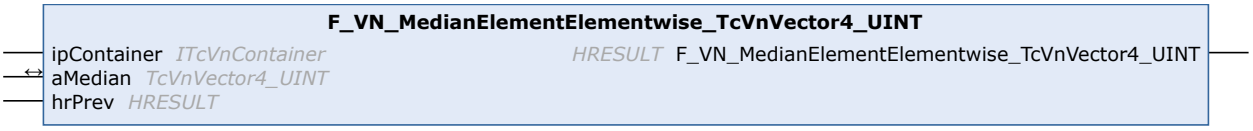
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

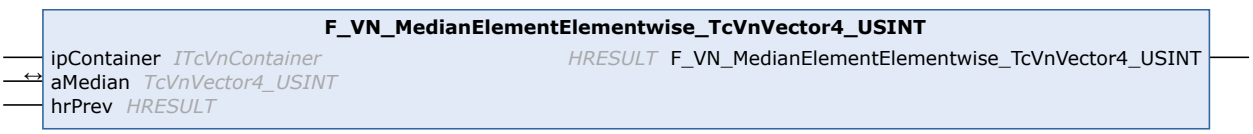
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

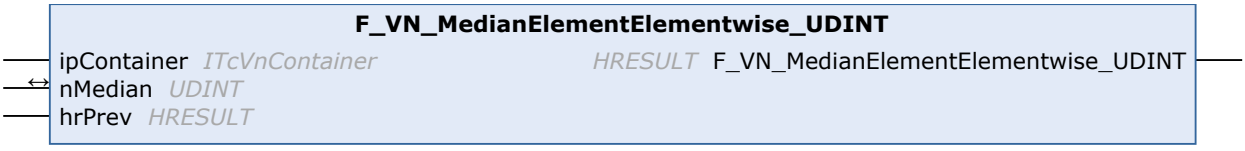
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

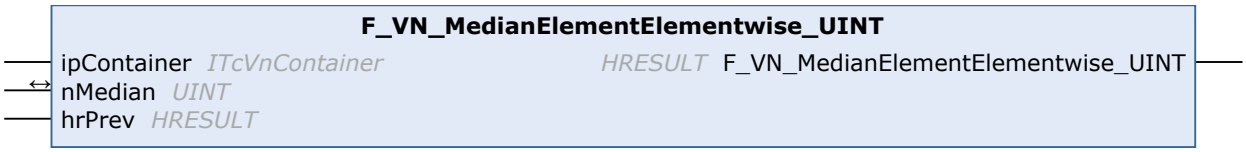
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

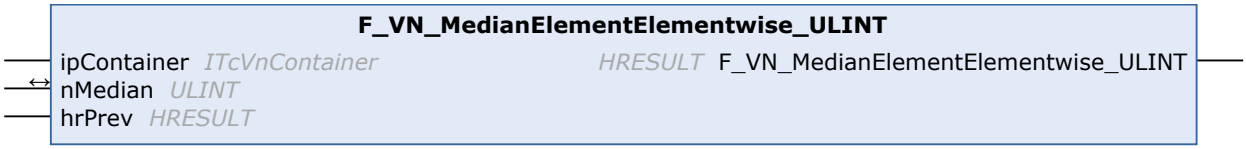
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

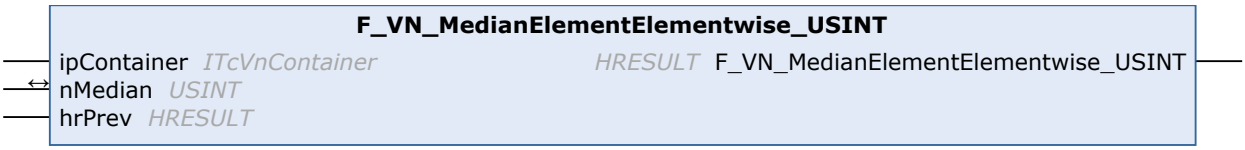
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

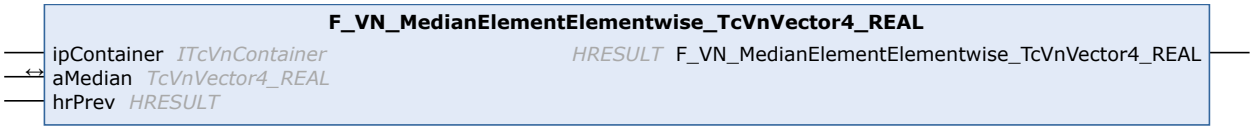
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

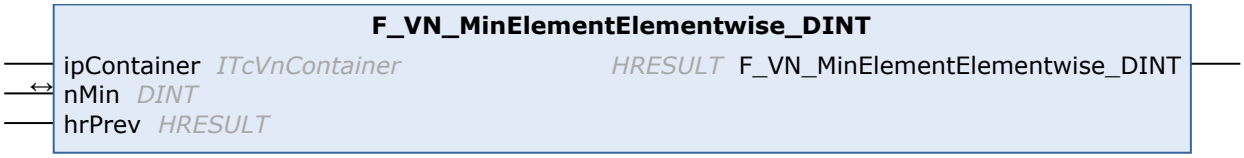
Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3 F_VN_MinElementElementwise

This chapter contains functions to determine the min value according to the data type for each element. This function is available for containers with basic elements.

For example, with a TcVnPoint2_DINT, the minimum X-value and the minimum Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

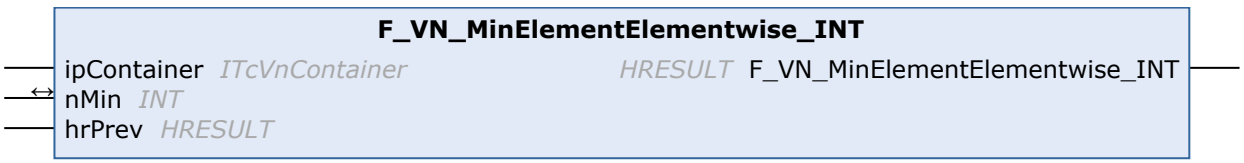
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

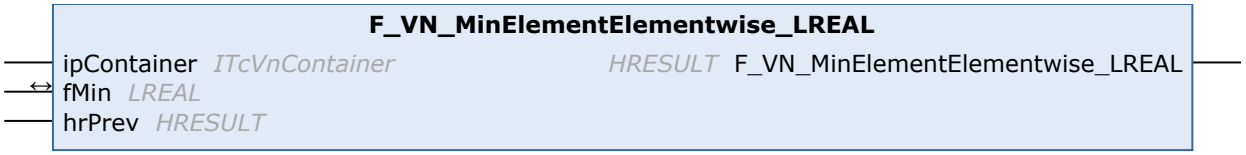
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

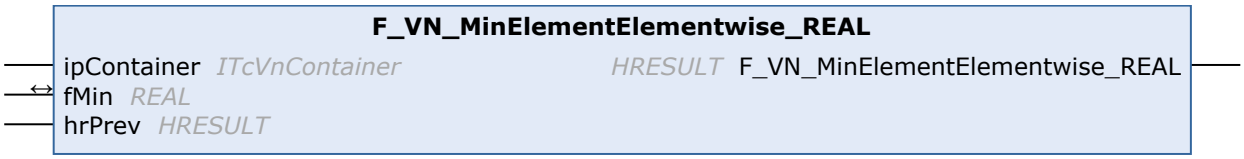
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

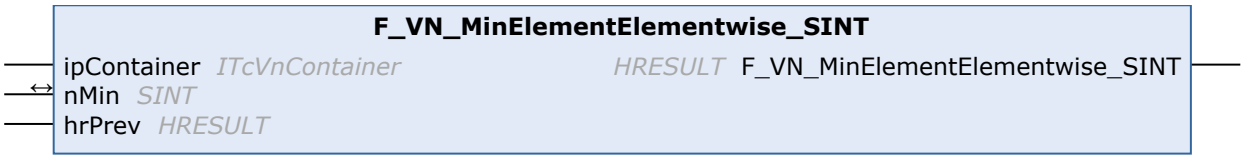
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

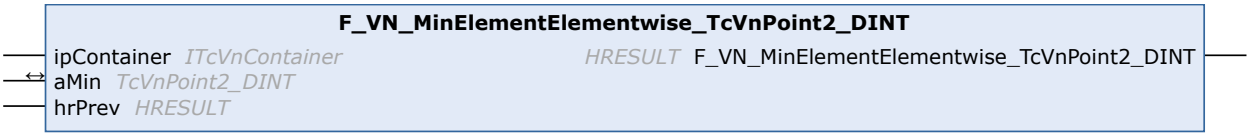
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_DINT [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

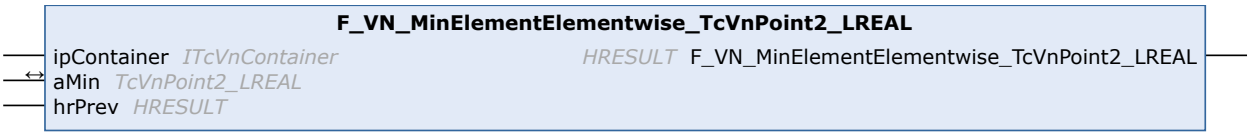
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

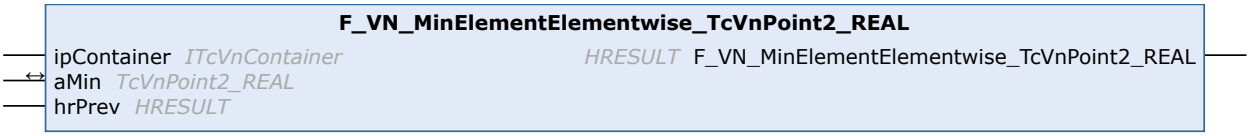
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

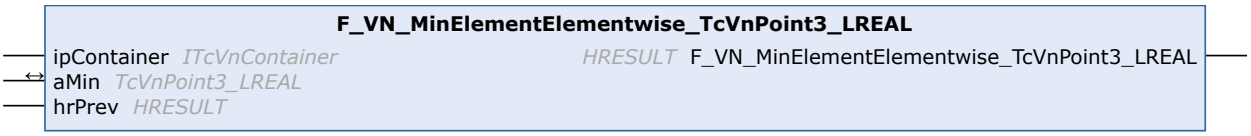
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint3_LREAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

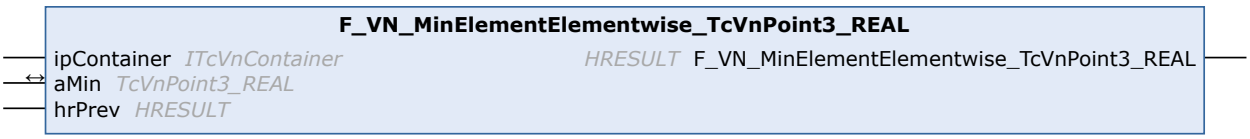
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint3_REAL [▶ 160]	Returns the requested element

Return value

HRESULT [▶ 141]

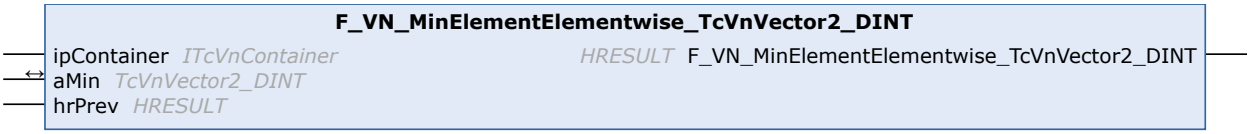
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

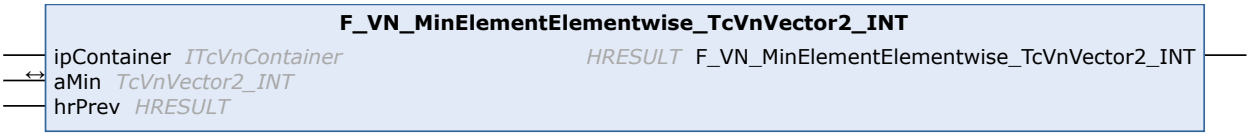
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

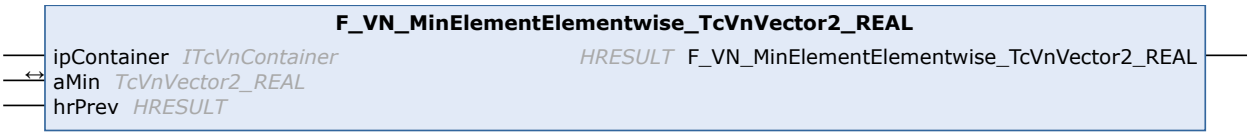
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

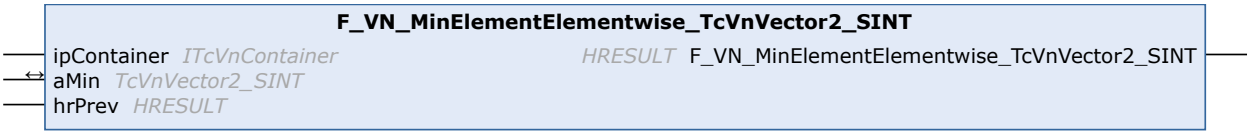
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

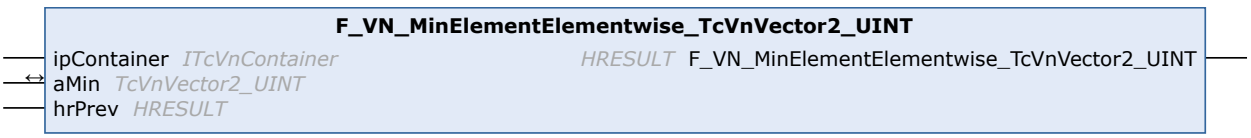
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

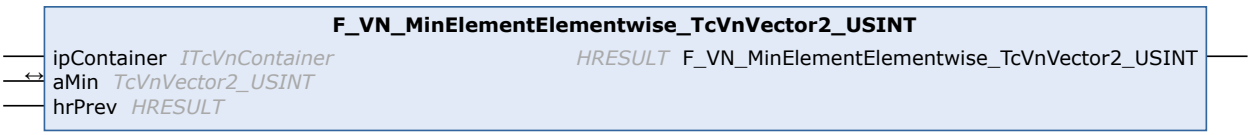
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

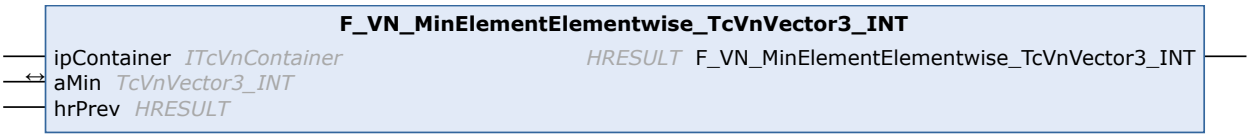
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

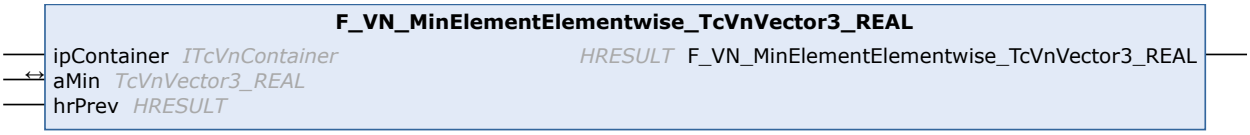
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

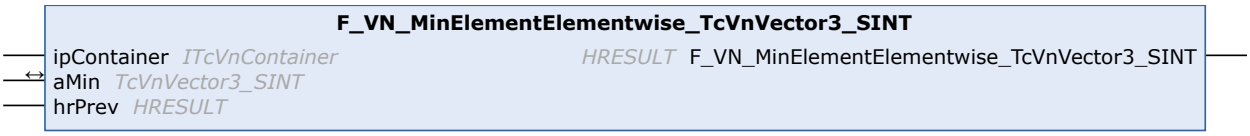
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

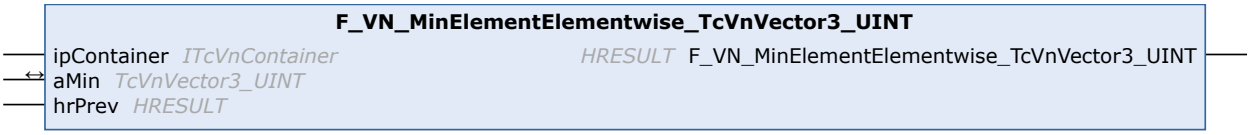
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

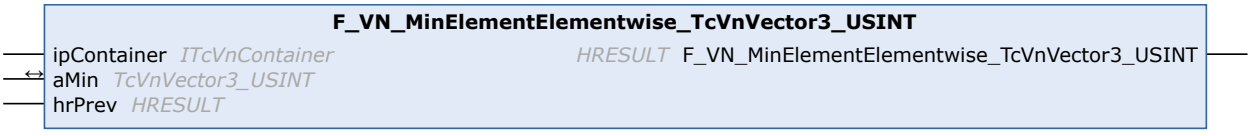
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

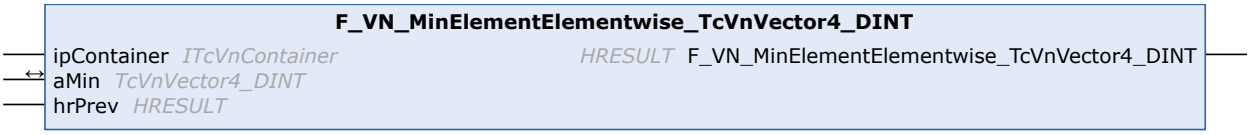
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_DINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

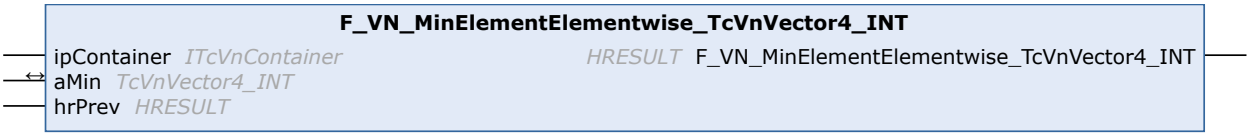
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_INT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

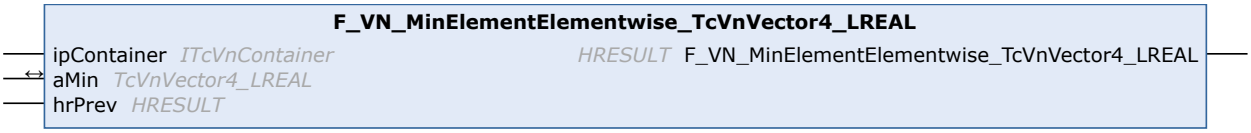
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_LREAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

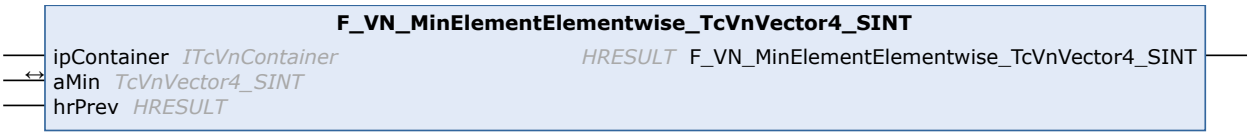
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_SINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

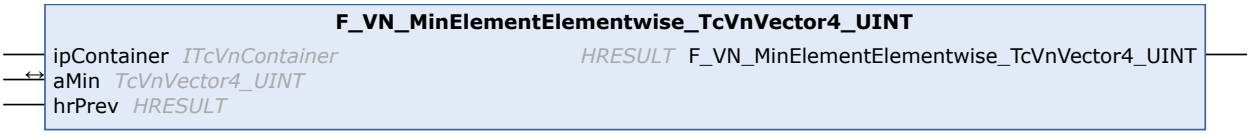
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_UINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

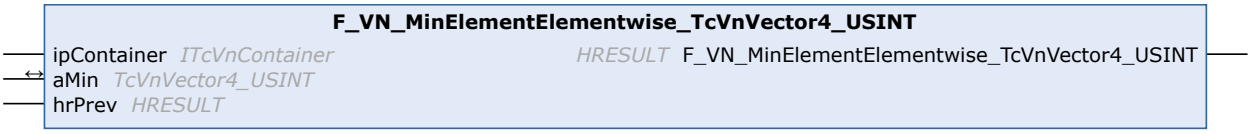
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_USINT [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

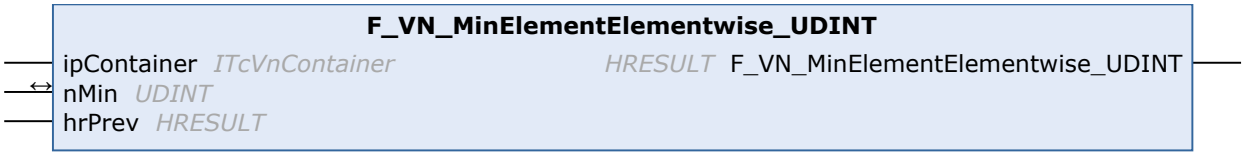
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

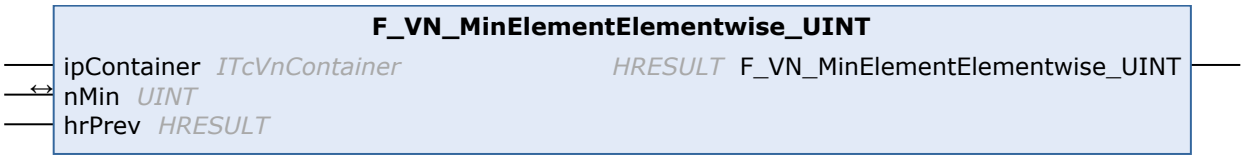
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

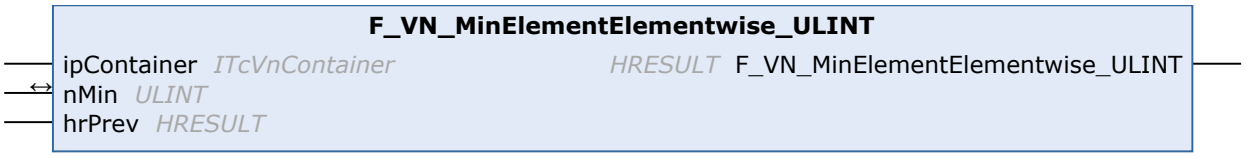
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

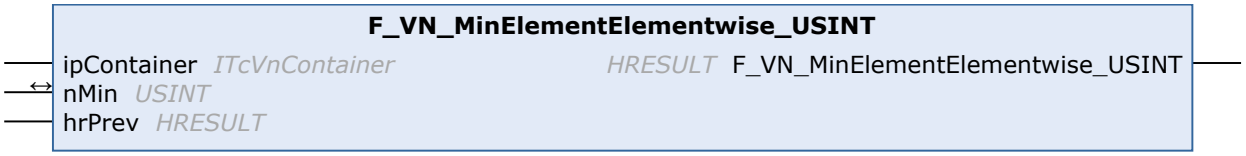
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

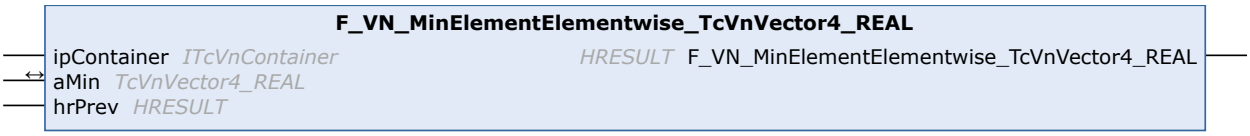
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_REAL [▶ 162]	Returns the requested element

Return value

HRESULT [▶ 141]

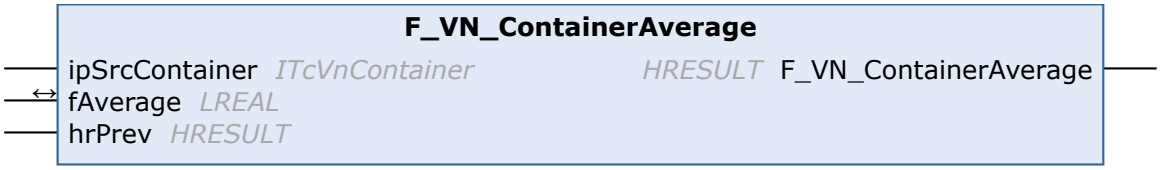
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

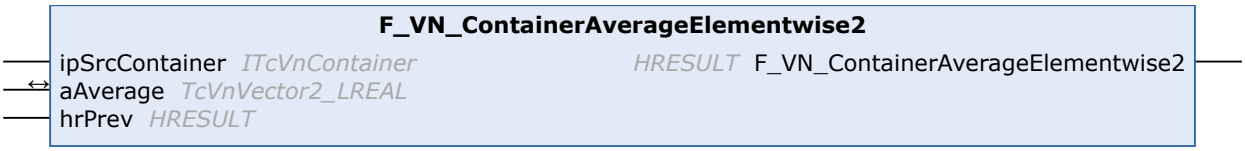
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 2D points or vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector2_LREAL [▶ 162]	Returns the calculated average values

Return value

HRESULT [▶ 141]

Further information

This function calculates the average values element by element for a container with two elementary sub-elements per container element.

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

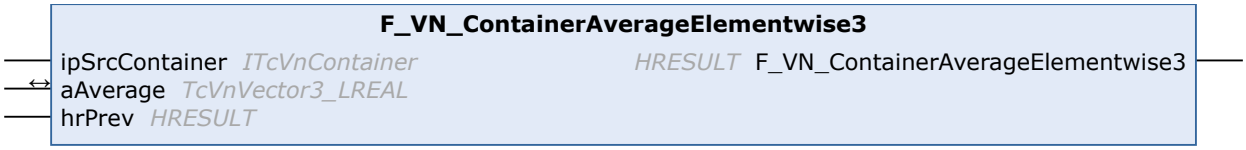
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 3D points or vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector3_LREAL [▶ 162]	Returns the calculated average values

Return value

HRESULT [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

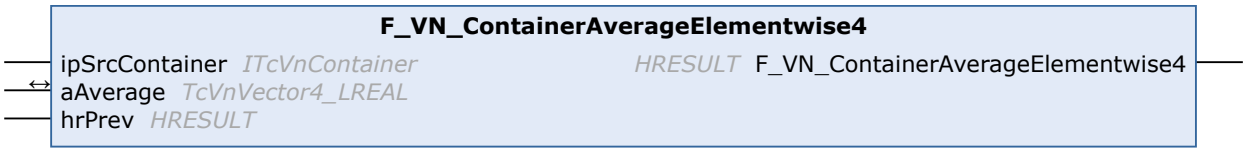
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 4D vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the calculated average values

Return value

HRESULT [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

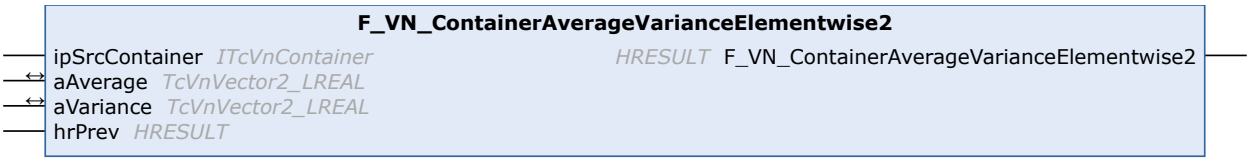
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 2D points or vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector2_LREAL [▶ 162]	Returns the calculated average values
aVariance	TcVnVector2_LREAL [▶ 162]	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 3D points or vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector3_LREAL [▶ 162]	Returns the calculated average values
aVariance	TcVnVector3_LREAL [▶ 162]	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 4D vectors
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the calculated average values
aVariance	TcVnVector4_LREAL [▶ 162]	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Further information

The table below shows which functions can be used for which container type:

Types of containers	Usable functions
Containers of elements	F_VN_ContainerAverage [▶ 943] F_VN_ContainerAverageVariance [▶ 947]
Containers of vectors, each with 2 elements	F_VN_ContainerAverageElementwise2 [▶ 944] F_VN_ContainerAverageVarianceElementwise2 [▶ 948]
Containers of vectors, each with 3 elements	F_VN_ContainerAverageElementwise3 [▶ 945] F_VN_ContainerAverageVarianceElementwise3 [▶ 950]
Containers of vectors, each with 4 elements	F_VN_ContainerAverageElementwise4 [▶ 946] F_VN_ContainerAverageVarianceElementwise4 [▶ 951]

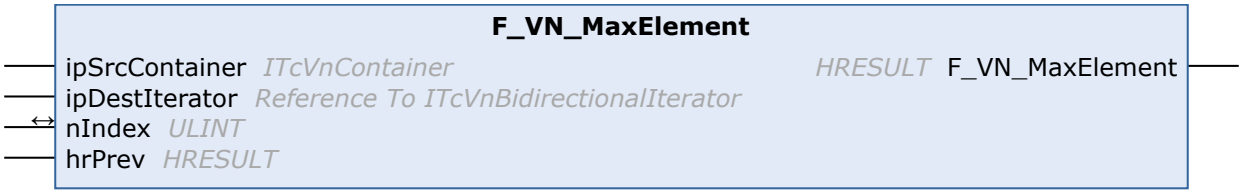
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 359]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

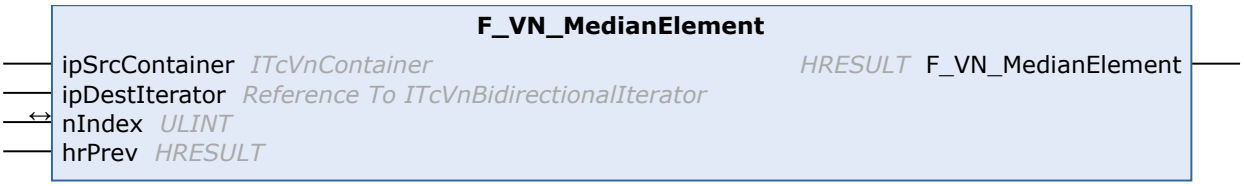
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 359]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

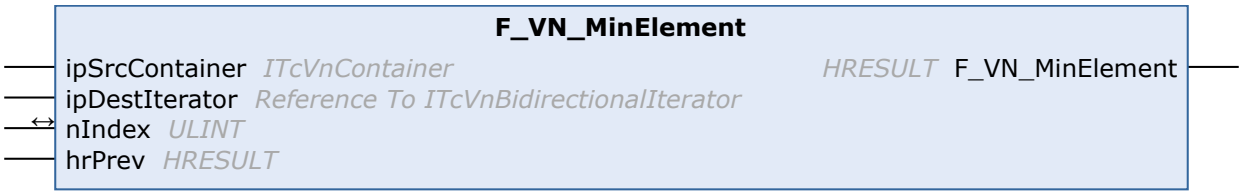
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 359]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9 Contour Analysis

The group contains functions for the analysis of contours [▶ 155]:

Functions

Shape approximation

- F_VN_ApproximatePolygon [▶ 956]
- F_VN_ConvexHullPoints(Exp) [▶ 971]
- F_VN_EnclosingCircle [▶ 976]
- F_VN_EnclosingRectangle [▶ 977]
- F_VN_EnclosingTriangle [▶ 978]
- F_VN_FitEllipse [▶ 979]
- F_VN_FitLine(Exp) [▶ 981]
- F_VN_UprightBoundingRectangle [▶ 988]

Points

- F_VN_CheckIfPointIsInsideContour [▶ 957]
- F_VN_ContourExtremePoint [▶ 964]
- F_VN_ConvexityDefects [▶ 973]

Geometrical features

- F_VN_ContourArea [▶ 958]
- F_VN_ContourCenterOfMass [▶ 959]
- F_VN_ContourCircularity [▶ 960]
- F_VN_ContourConvexity [▶ 961]
- F_VN_ContourEccentricity [▶ 962]
- F_VN_ContourElongation [▶ 963]
- F_VN_ContourInertiaRatio [▶ 965]
- F_VN_ContourMoments [▶ 966]
- F_VN_ContourOrientation(Exp) [▶ 967]
- F_VN_ContourPerimeter [▶ 969]
- F_VN_ContourRoundness [▶ 970]
- F_VN_FourierDescriptors [▶ 982]

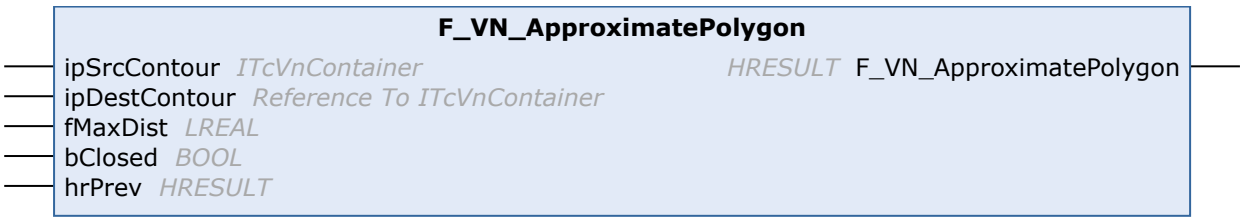
Matching

- F_VN_MatchContours(Exp) [▶ 987]
- F_VN_MatchContours1vsN(Exp) [▶ 986]

User-defined operations

- F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 974]
- F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 975]

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 [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 160]	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 [▶ 141]

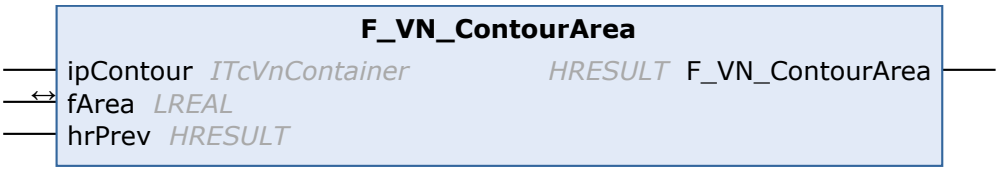
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

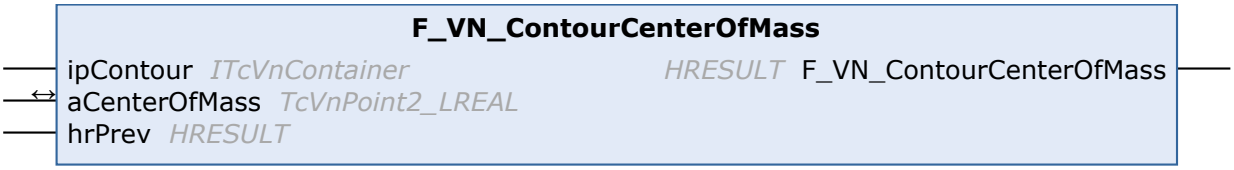
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 160]	Returns the center of mass of the contour

Return value

HRESULT [▶ 141]

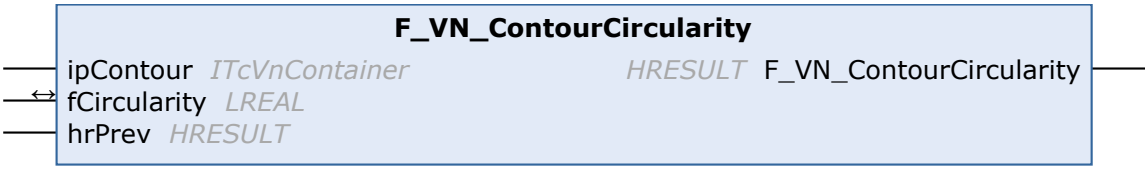
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

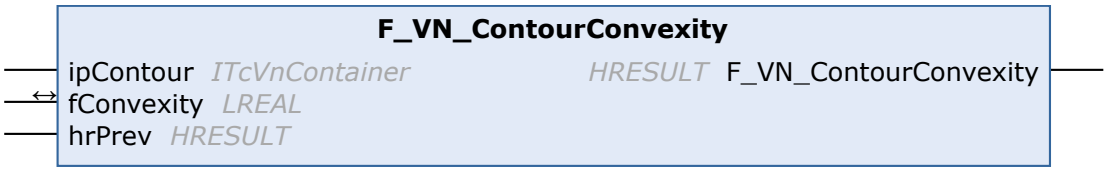
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

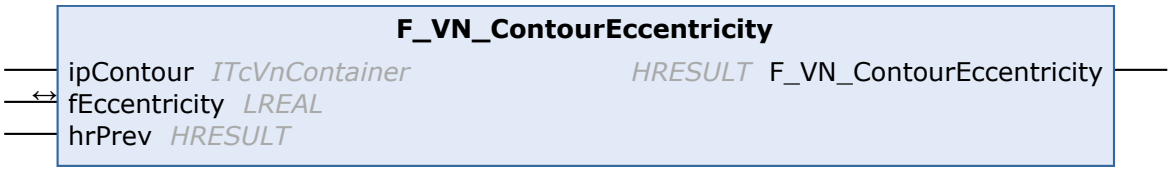
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

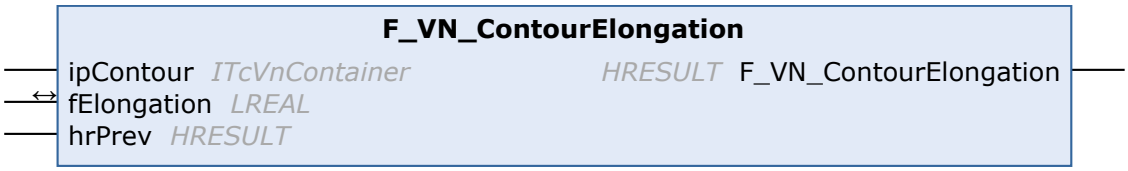
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

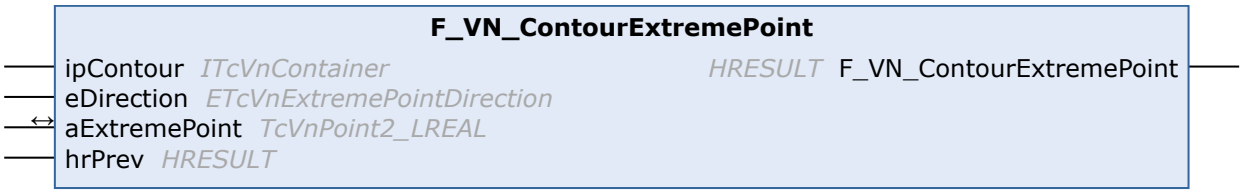
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source container with 2D points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
eDirection	ETcVnExtremePointDirection [▶ 200]	Selects the search direction for the extreme point
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aExtremePoint	TcVnPoint2_LREAL [▶ 160]	Returns the extreme point of the contour, according to eDirection

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

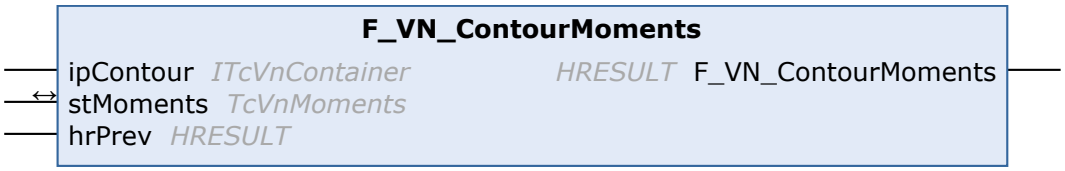
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 232]	Returns a struct containing the moments

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.

Return value

HRESULT [▶ 141]

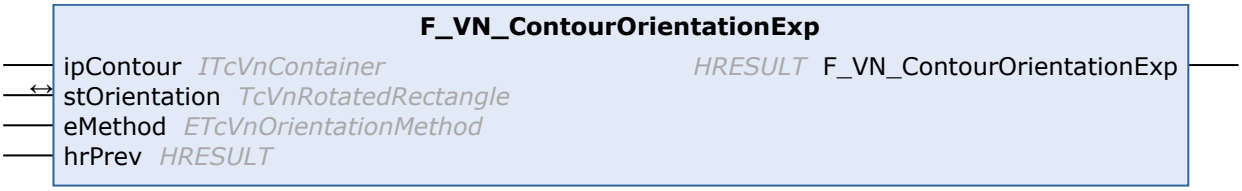
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 211]	Method for calculating the orientation.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	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 [▶ 141]

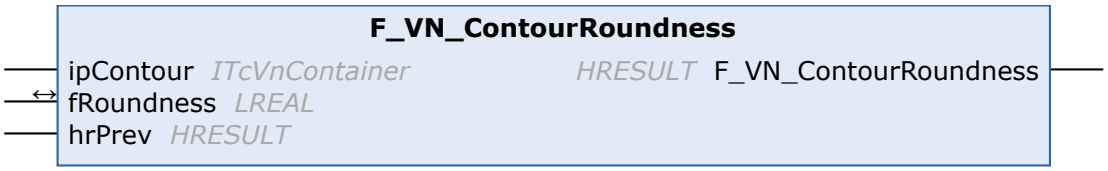
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

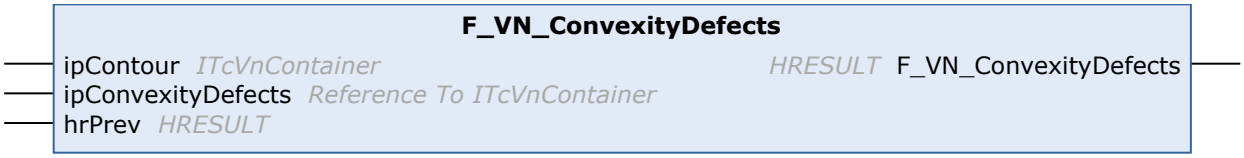
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source contour (ContainerType_Vector_TcVnPoint2_DINT; The elements of this container are the individual points.)
ipConvexityDefects	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

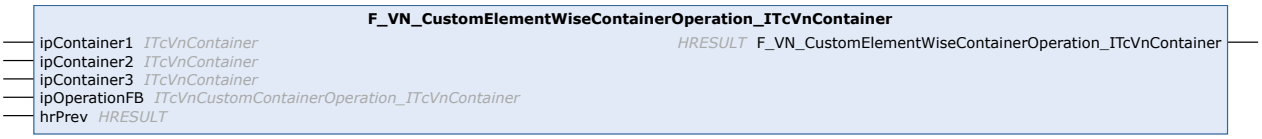
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container 1
ipContainer2	ITcVnContainer [▶ 370]	Container 2
ipContainer3	ITcVnContainer [▶ 370]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnContainer [▶ 246]	Custom operation on the container elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

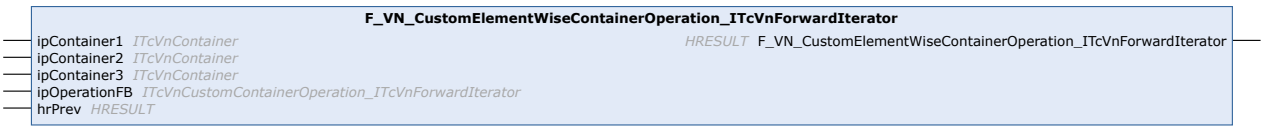
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container 1
ipContainer2	ITcVnContainer [▶ 370]	Container 2
ipContainer3	ITcVnContainer [▶ 370]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnForwardIterator [▶ 247]	Custom operation on the elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

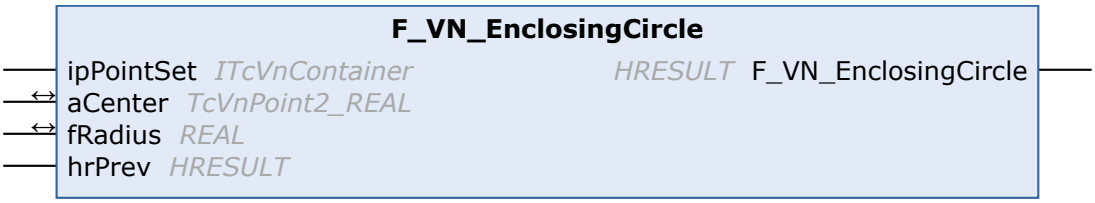
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Returns the center of the circle
fRadius	REAL	Returns the radius of the circle

Return value

HRESULT [▶ 141]

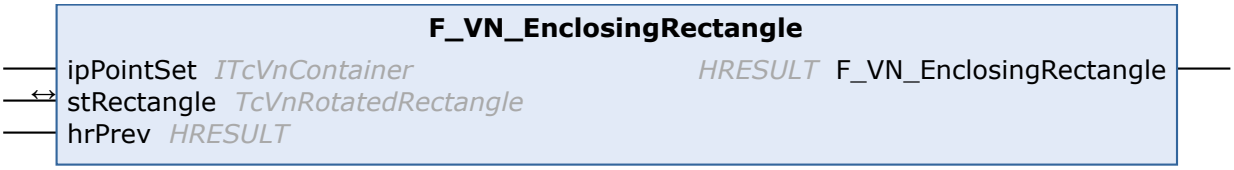
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 245]	Returns the determined rectangle

Return value

HRESULT [▶ 141]

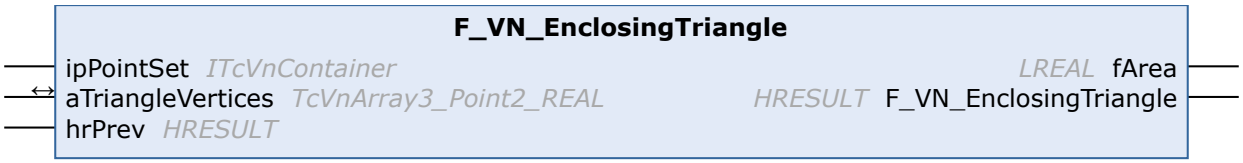
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 141]	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 [▶ 162]	Returns the 3 triangle vertices

Outputs

Name	Type	Description
fArea	LREAL	Returns the triangle area

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Resulting ellipse, described by a rotated rectangle.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFitLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 141]

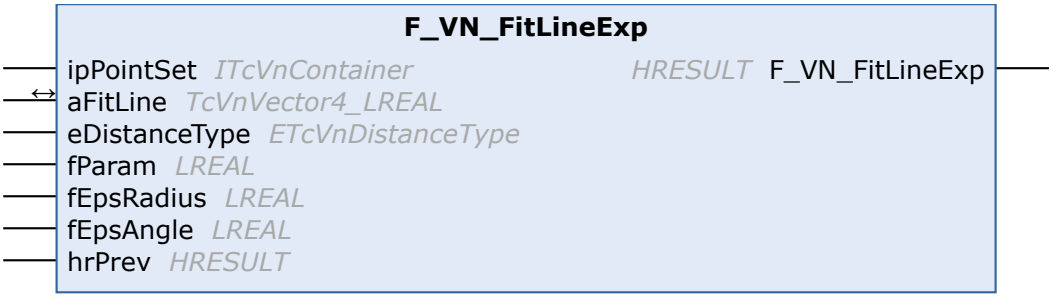
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
eDistanceType	ETcVnDistanceType [▶ 196]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFitLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 141]

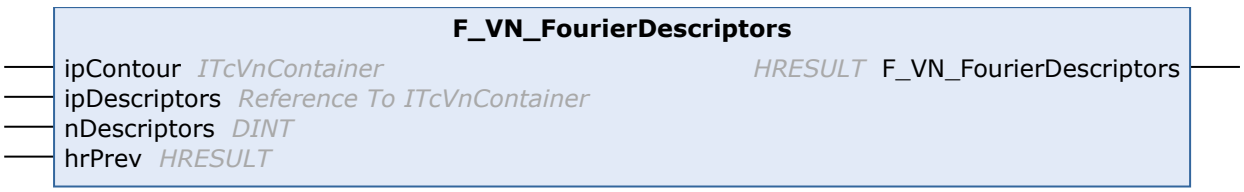
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

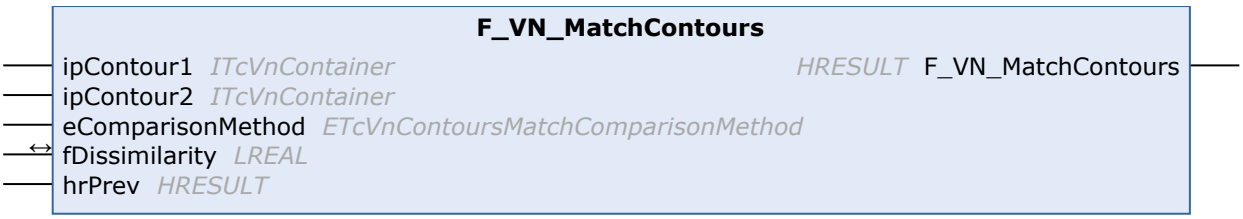
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 193]	Method used for comparing the Hu moment invariants of the contours
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Samples

- Match Contours 1vsN (manual shapes) [▶ 2771]
- Match Contours (extracted shapes) [▶ 2774]

Related functions

- F_VN_ContourMoments [▶ 966]
- F_VN_HuMomentInvariants [▶ 1538]
- F_VN_MatchContours [▶ 983]
- F_VN_MatchContours1vsN [▶ 984]

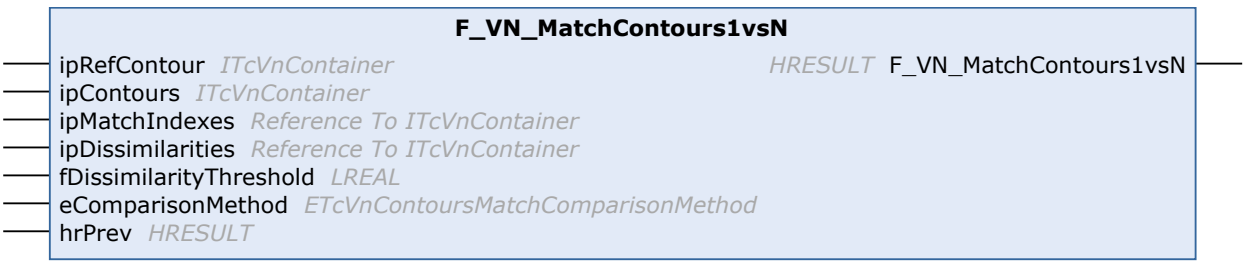
Required License

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 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 [▶ 370]	Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer [▶ 370]	Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 193]	Method used for comparing the Hu moment invariants of the contours
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

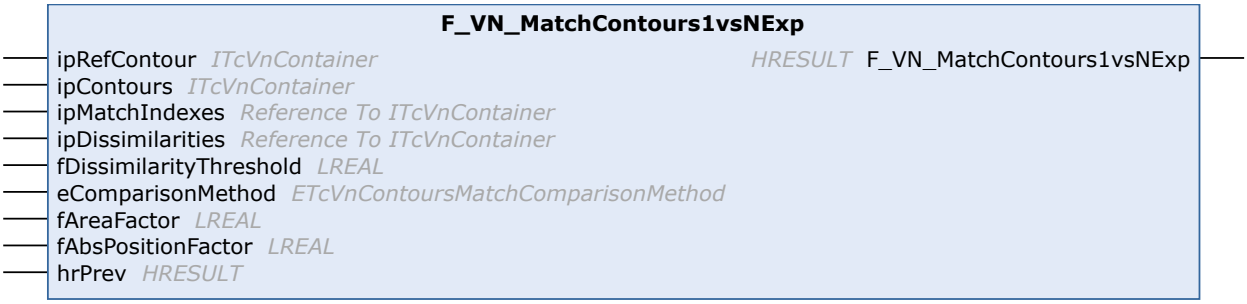
Required License

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 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 [▶ 370]	Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer [▶ 370]	Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 193]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

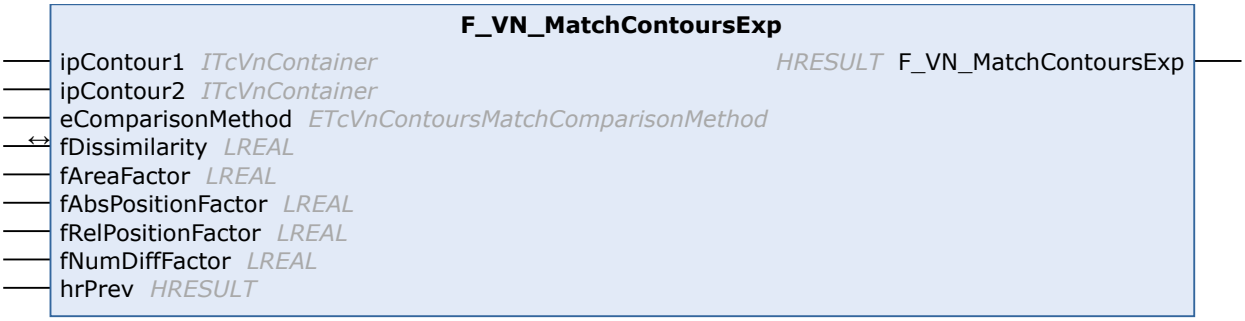
Required License

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 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 [▶ 370]	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 [▶ 370]	Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 193]	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 [▶ 141]	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 [▶ 141]

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 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 [▶ 370]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_UDINT [▶ 245]	Returns the determined rectangle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10 Control Functions

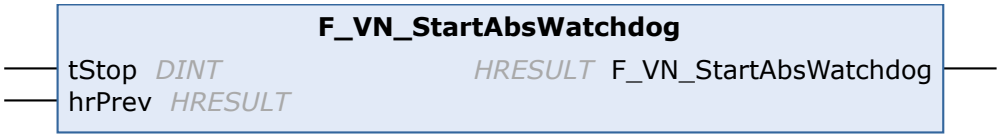
This group contains functions for sequential control, e.g. for the handling of watchdogs [▶ 146].

Functions

Watchdogs

- F_VN_StartAbsWatchdog(Exp) [▶ 990]
- F_VN_StartRelWatchdog(Exp) [▶ 992]
- F_VN_StopWatchdog [▶ 993]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

Further information can be found in the chapter Watchdogs [▶ 146].

tStop time exceeded

Note that the tStop time is slightly exceeded (by a few µs) when the watchdog is triggered.

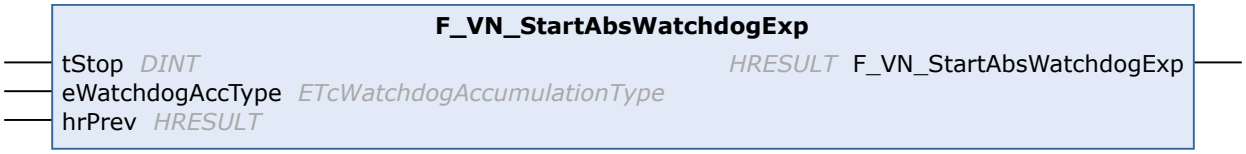
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 223]	Accumalation method used for combining the results of multiple functions enclosed by this watchdog
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

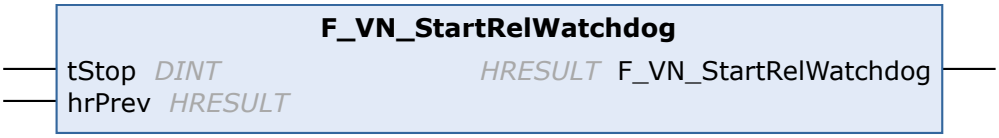
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

Further information can be found in the chapter Watchdogs [▶ 146].

tStop time exceeded

Note that the tStop time is slightly exceeded (by a few µs) when the watchdog is triggered.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 223]	Accumalation method used for combining the results of multiple functions enclosed by this watchdog
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

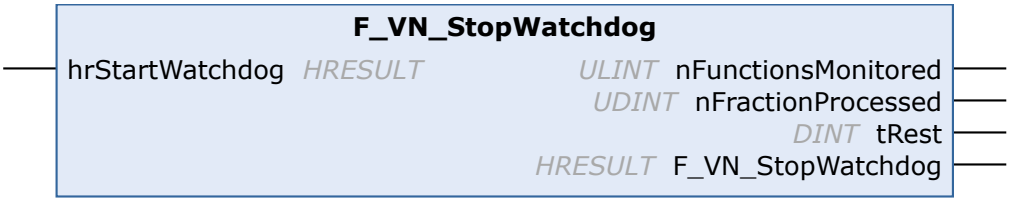
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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 [▶ 141]

Further information

The output nFractionProcessed must be interpreted differently depending on the parameterization of the watchdog. This is set with ETcWatchdogAccumulationType [▶ 223] when starting the watchdog.

Further information can be found under Watchdogs [▶ 146].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11 Drawing

This group contains functions for painting and describing images.

Handling 4-channel colors

4-channel colors are accepted directly into the result image. No alpha blending or similar takes place.

Functions

Arrow

- F_VN_DrawArrow(Exp) [▶ 995]
- F_VN_DrawArrow(Exp)_TcVnVector4_DINT [▶ 996]

Circles and Arcs

- F_VN_DrawCircle(Exp) [▶ 1000]
- F_VN_DrawCircles(Exp) [▶ 1002]
- F_VN_DrawCircularArc(Exp) [▶ 1004]
- F_VN_FillCircle [▶ 1043]

Components

- F_VN_DrawComponents(Exp) [▶ 1006]

Contours

- F_VN_DrawContours(Exp) [▶ 1009]
- F_VN_FillContours [▶ 1044]

Ellipse

- F_VN_DrawEllipse(Exp) [▶ 1012]
- F_VN_FillEllipse [▶ 1045]

Keypoints and Matches

- F_VN_DrawKeypoints(Exp) [▶ 1014]
- F_VN_DrawMatches(Exp) [▶ 1026]

Lines

- F_VN_DrawLine(Exp) [▶ 1016]
- F_VN_DrawLines(Exp) [▶ 1023]
- F_VN_DrawLine(Exp)_TcVnVector4_DINT [▶ 1017]
- F_VN_DrawLine(Exp)_TcVnVector4_LREAL [▶ 1018]

Orientation

- F_VN_DrawOrientation(Exp) [▶ 1028]

Points

- F_VN_DrawPoint(Exp) [▶ 1031]

- F_VN_DrawPoints(Exp) [▶ 1033]

Polygon

- F_VN_DrawPolygon(Exp) [▶ 1035]
- F_VN_FillPolygon [▶ 1046]

Rectangle

- F_VN_DrawRectangle [▶ 1037]
- F_VN_DrawRectangle_TcVnRectangle_DINT [▶ 1038]
- F_VN_DrawRectangle_TcVnRectangle_UDINT [▶ 1039]
- F_VN_DrawRotatedRectangle(Exp) [▶ 1040]
- F_VN_FillRectangle [▶ 1047]
- F_VN_FillRotatedRectangle [▶ 1048]

Intensity Profile

- F_VN_PlotIntensityProfile(Exp) [▶ 1049]

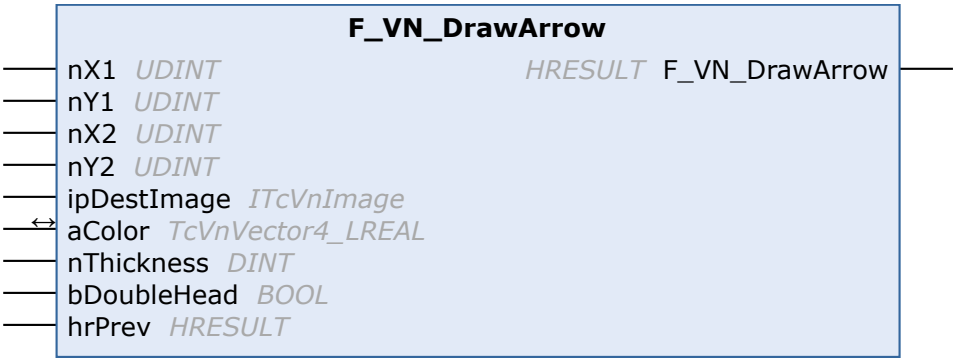
Text

- F_VN_PutLabel(Exp) [▶ 1051]
- F_VN_PutText(Exp) [▶ 1053]

Painting and filling

For many shapes there is an option to paint both the outline of the shape and the filled shape. Two functions are available for this: (`F_VN_Draw` and `F_VN_Fill` respectively). In addition, setting the outline thickness `nThickness` to `-1` leads to the shape being filled.

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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

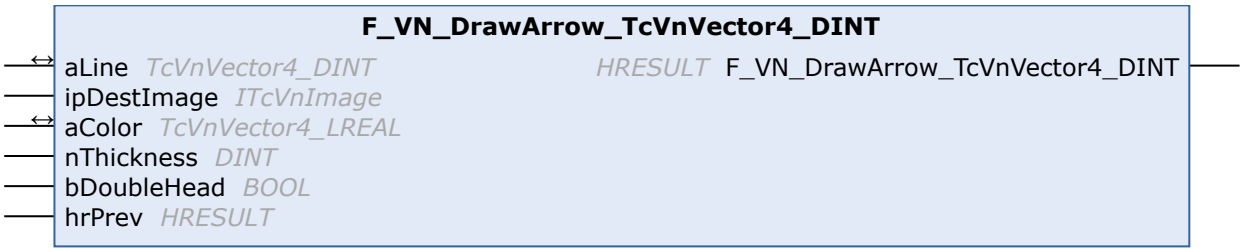
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 162]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

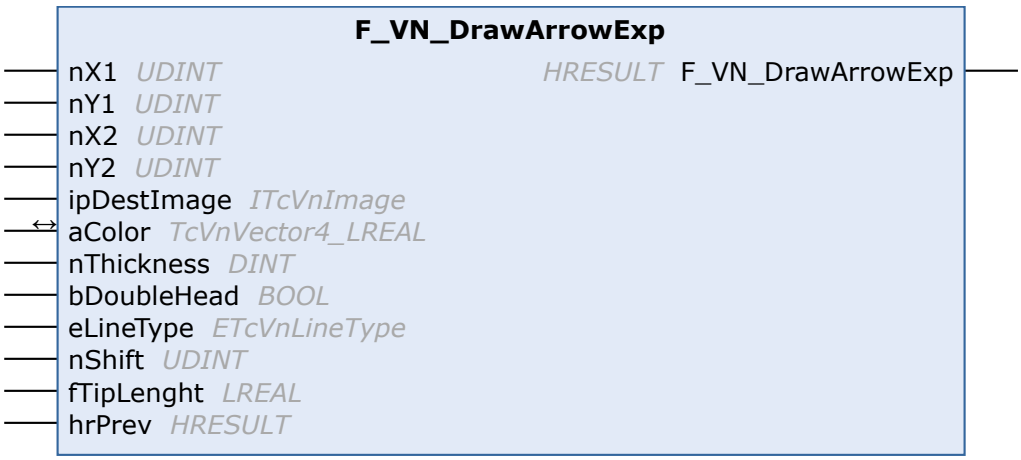
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 208]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

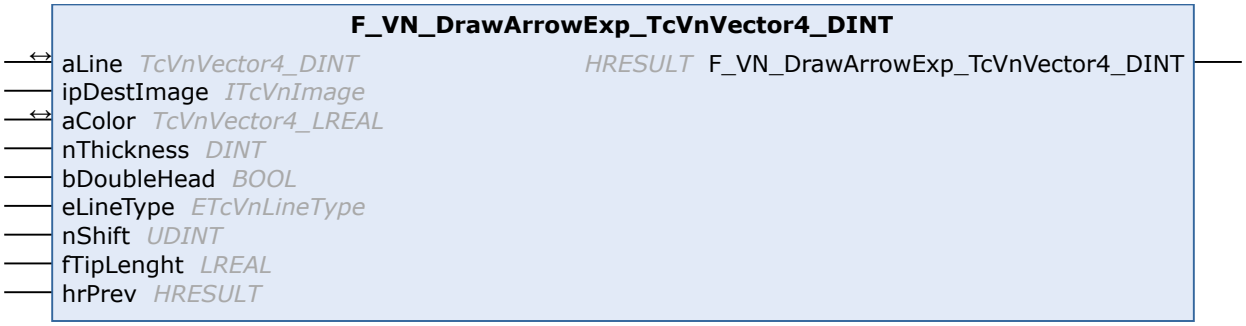
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 208]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 162]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

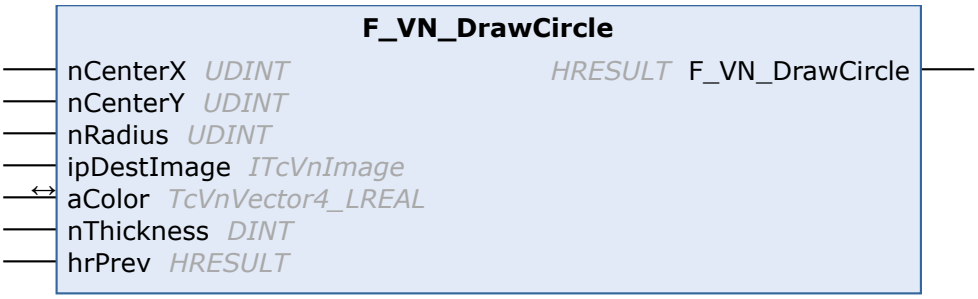
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

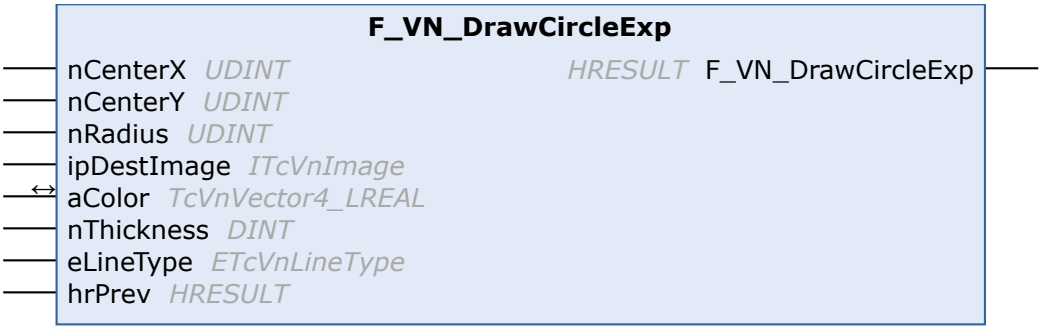
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

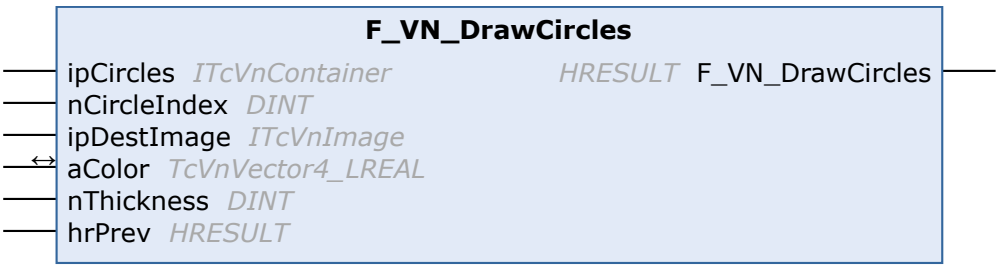
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

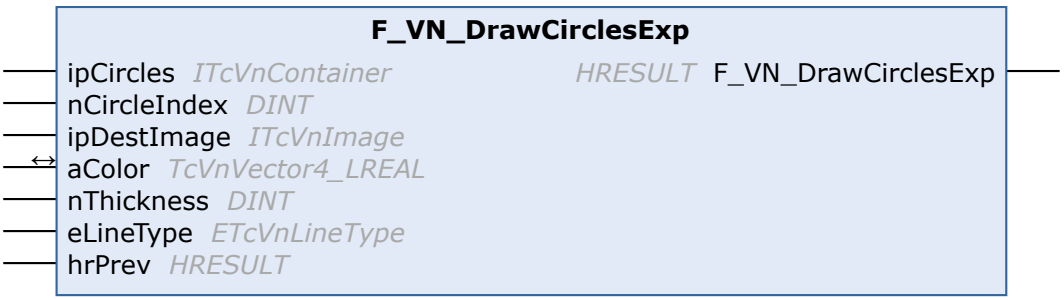
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

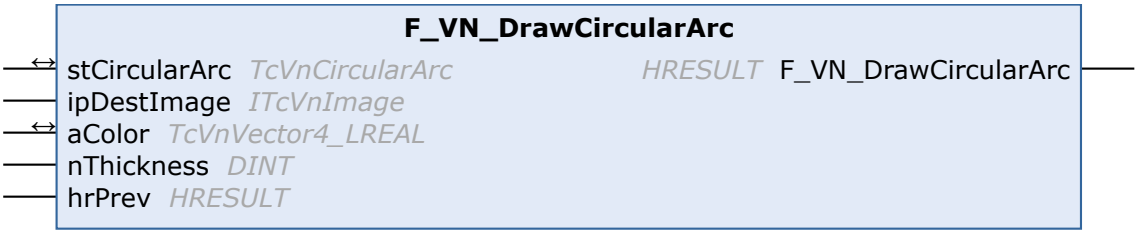
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the arc is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 227]	Circular arc definition
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

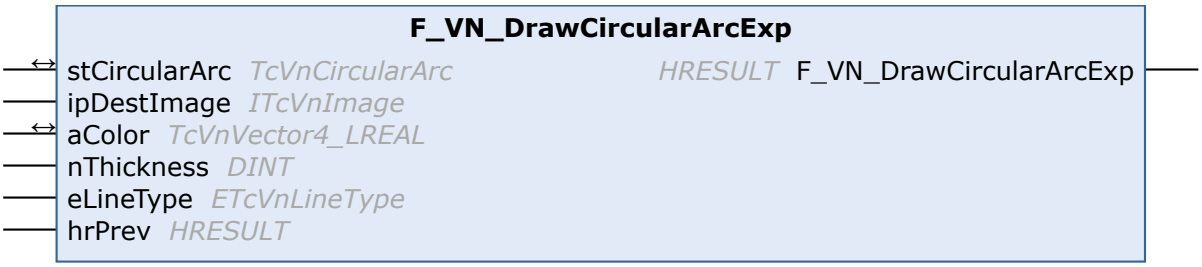
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the arc is filled)
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 227]	Circular arc definition
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

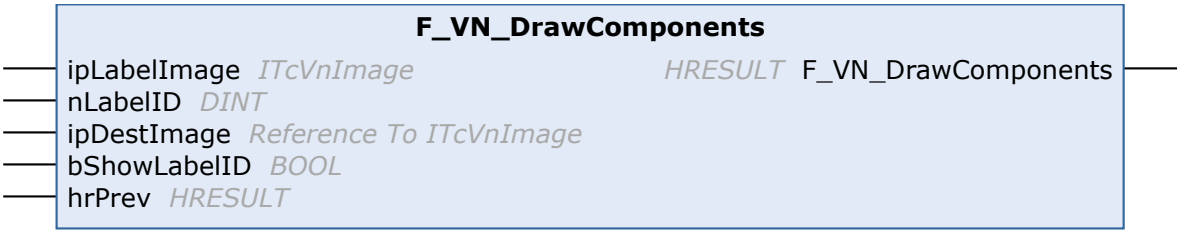
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

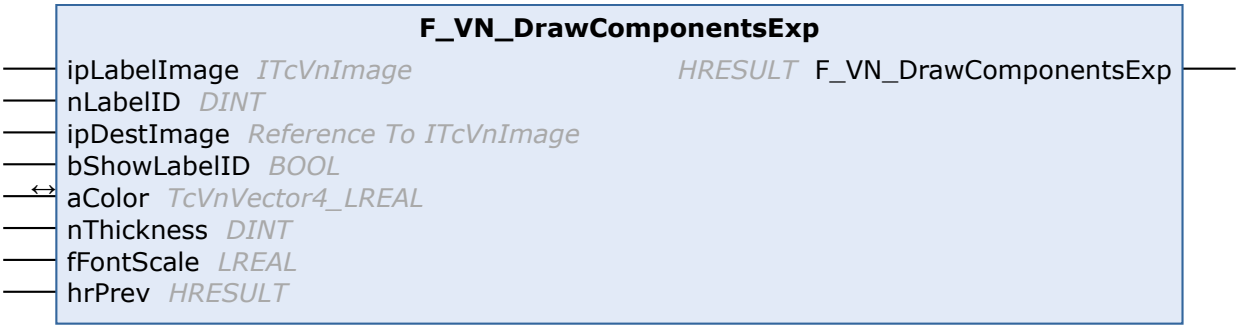
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color to draw the components (for {-1, -1, -1, -1}, a random color is chosen for each component)

Return value

HRESULT [▶ 141]

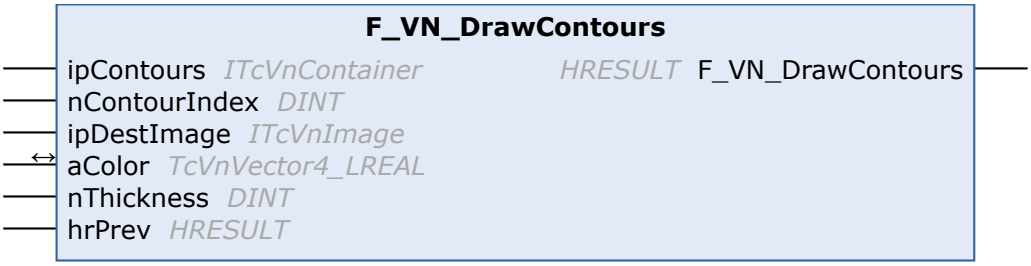
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the contours are filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_DrawContours` draws contours in the destination image. A contour is described based on a set of points that are stored in a container. Please note that only integer contour points can be drawn, so that the container must be of type *ContainerType_Vector_TcVnPoint2_DINT*. If the points are described using floating point numbers, F_VN_ConvertContainerType [▶ 748] can be used to convert the points.

In addition to a container with one contour, a container with several contours can be transferred to the function. This is of type *ContainerType_Vector_Vector_TcVnPoint2_DINT*. The input variable *nContourIndex* indicates which contour is to be drawn. If all contours are to be drawn, *nContourIndex* must be set to a negative value. It is more efficient to draw an entire container of contours at once (set *nContourIndex* to a negative value) than to draw them one after the other by repeatedly calling the function within a loop.

Expert parameters

The expert version F_VN_DrawContoursExp [▶ 1010] contains additional parameters.

Application

The painting of all contours in a contour list `ipContours` with a red line of 5 px in thickness looks like this, for example:

```
hr := F_VN_DrawContours(
    ipContours      :=  ipContours,
    nContourIndex   :=  -1,
    ipDestImage     :=  ipImageRes,
    aColor          :=  aColorRed,
    nThickness      :=  5,
    hrPrev          :=  hr
);
```

Typical functions that return contours of type ContainerType_Vector_Vector_TcVnPoint2_DINT

- F_VN_DetectBlobs [▶ 1179]
- F_VN_FindContours [▶ 1188] or F_VN_FindContoursExp [▶ 1191]
- F_VN_FindContourHierarchyExp [▶ 1184]

Samples

- Blob Detection with watchdog monitoring [▶ 2780]

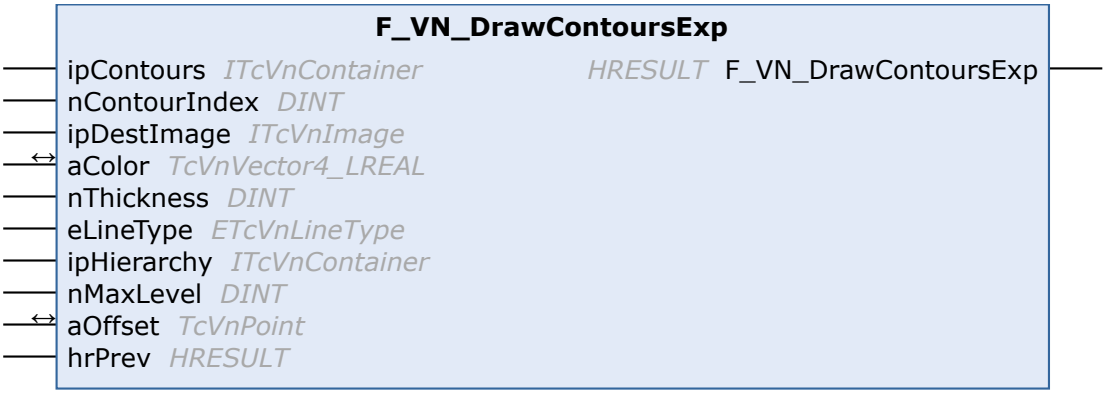
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the contours are filled)
eLineType	ETcVnLineType [▶ 208]	Line type
ipHierarchy	ITcVnContainer [▶ 370]	Contour hierarchy (ContainerType_Vector_TcVnVector4_DINT)
nMaxLevel	DINT	Maximum level of contours to be drawn
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color
aOffset	TcVnPoint [▶ 160]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Further information

The function F_VN_DrawContoursExp corresponds to the function F_VN_DrawContours [▶ 1009], extended by additional input variables.

Thus, the hierarchy description, which is returned by the function F_VN_FindContourHierarchyExp [▶ 1183] in addition to the found contours, can also be taken into account. The hierarchy level up to which the contours are to be drawn is specified via `ipHierarchy` and via the input variable `nMaxLevel`. If 0 is transferred for `ipHierarchy`, the hierarchy is ignored.

`aOffset` can be used to draw all contour points with a uniform offset in x and y direction. This is of interest, for example, if the contours were searched in a ROI but are to be drawn into the original 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 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the ellipse is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Ellipse to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the ellipse is filled)
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Ellipse to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

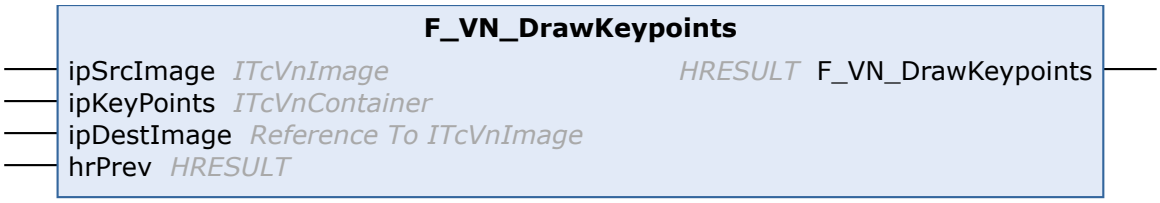
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer [▶ 370]	Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate color image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

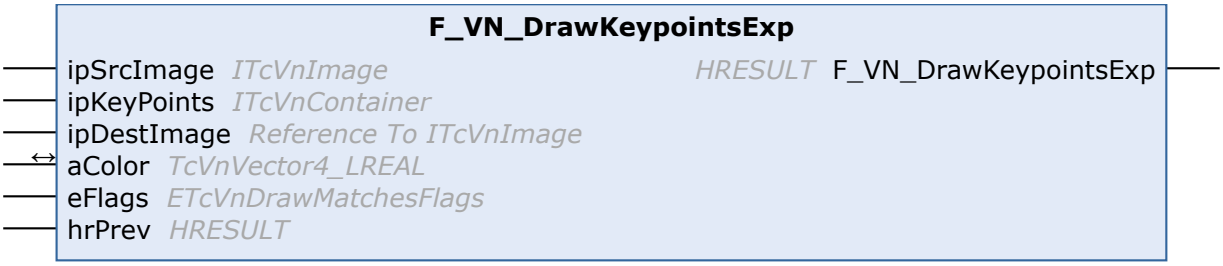
Required License

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 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 [▶ 411]	Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer [▶ 370]	Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate color image will be created if required.)
eFlags	ETcVnDrawMatchesFlags [▶ 196]	A combination of flags to support overdrawing an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color to draw the keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 141]

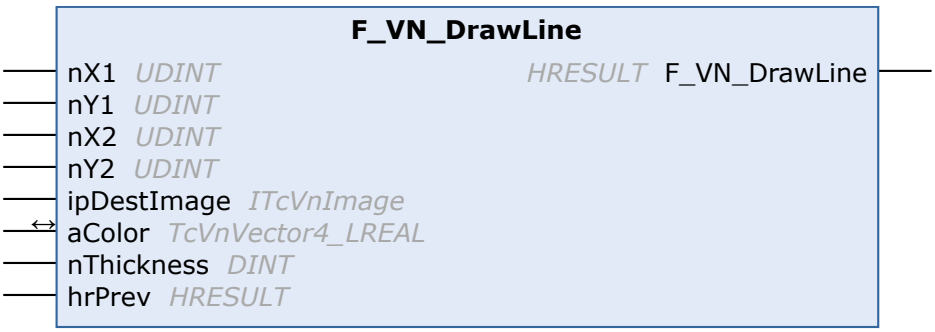
Required License

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 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

HRESULT

16#734	OUTOFRANGE	Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531] to ensure that the specified line fits in the image area.

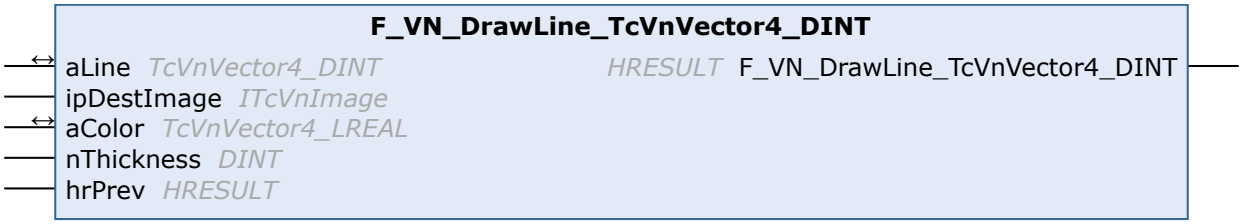
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 162]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

HRESULT

16#734	OUTOFRANGE	Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531] to ensure that the specified line fits in the image area.

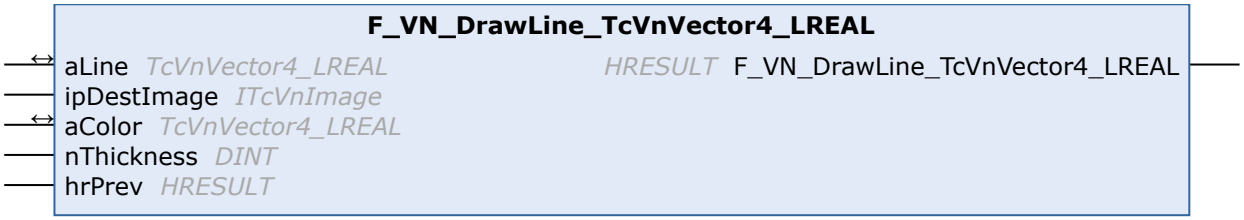
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 162]	Color

Return value

HRESULT [▶ 141]

NOTICE

Floating point exceptions

This function can cause errors if the option **Floating point exceptions** of the executing PLC task [▶ 61] is active. Therefore, deactivate this option.

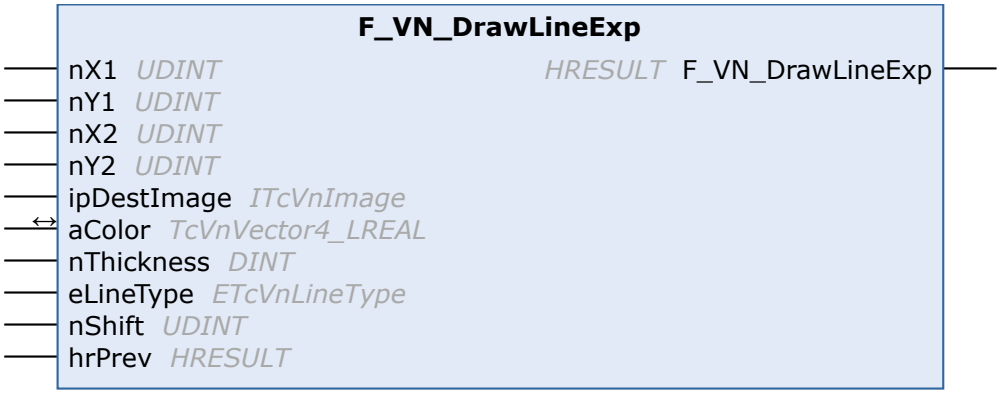
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 208]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

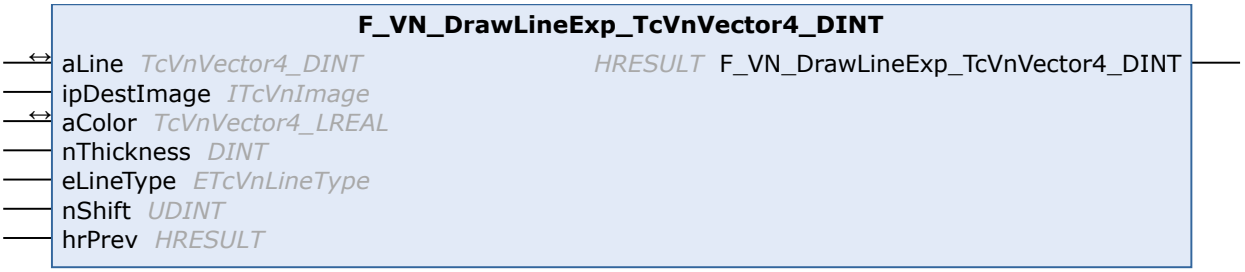
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 208]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 162]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

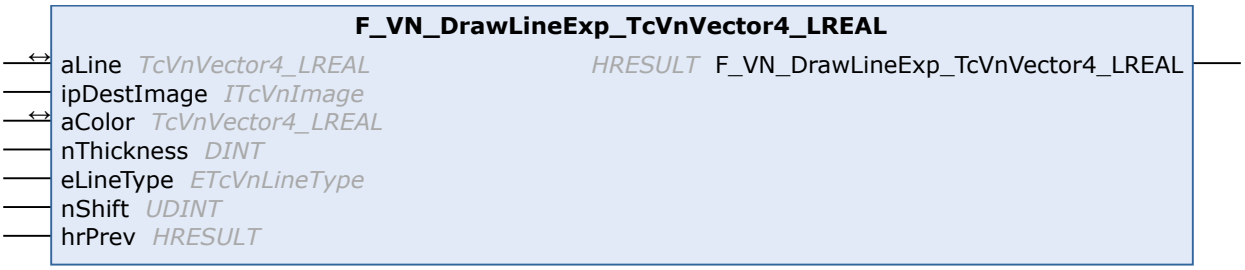
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 208]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 162]	Color

Return value

HRESULT [▶ 141]

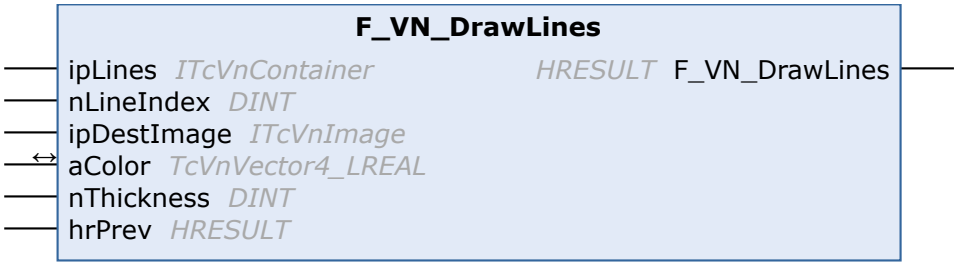
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

HRESULT

16#734	OUTOFRANGE	Part of the line lies outside of the image area. Use the function F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531] to ensure that the specified line fits in the image area.

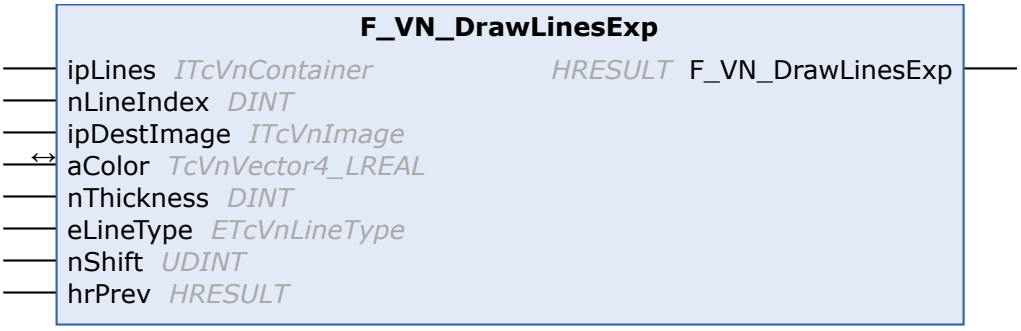
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 208]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

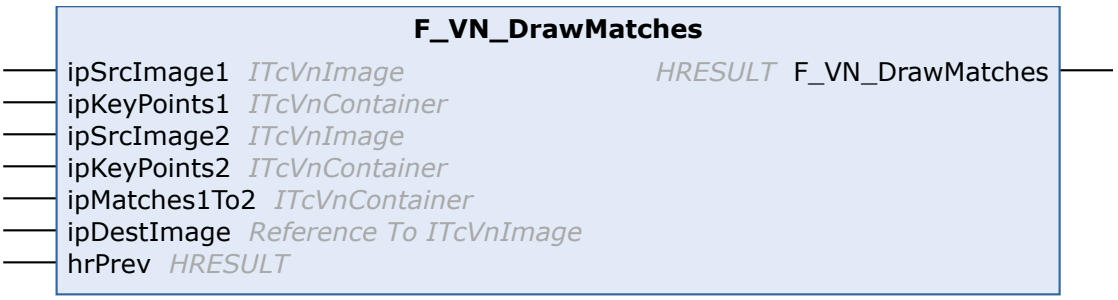
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer [▶ 370]	Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage [▶ 411]	Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer [▶ 370]	Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer [▶ 370]	Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate color image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

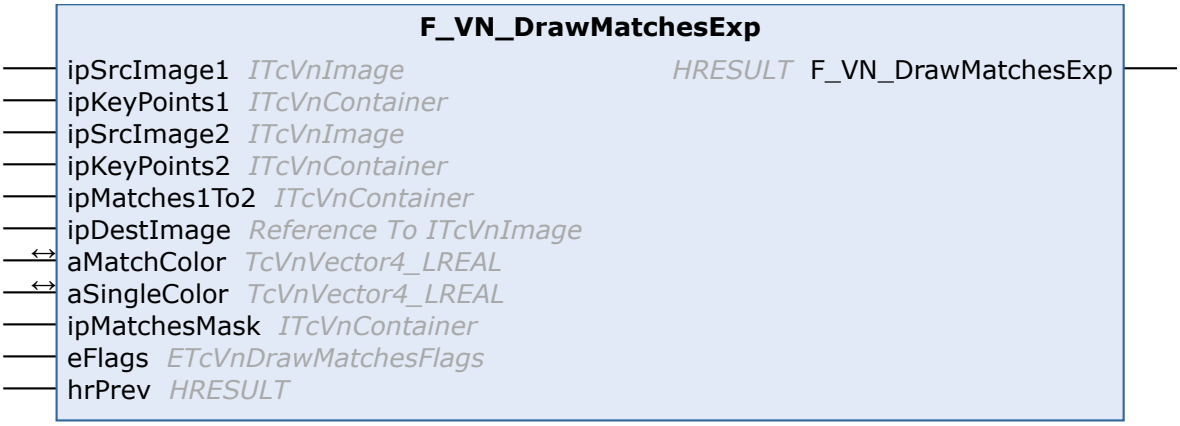
Required License

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 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 [▶ 411]	Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer [▶ 370]	Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage [▶ 411]	Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer [▶ 370]	Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer [▶ 370]	Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate color image will be created if required.)
ipMatchesMask	ITcVnContainer [▶ 370]	Mask to select the matches to be drawn (ContainerType_Vector_SINT; Set to 0 if all matches should be drawn.)
eFlags	ETcVnDrawMatchesFlags [▶ 196]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMatchColor	TcVnVector4_LREAL [▶ 162]	Color to draw the matches (for {-1, -1, -1, -1}, a random color is chosen for each point and line)
aSingleColor	TcVnVector4_LREAL [▶ 162]	Color to draw the single keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 141]

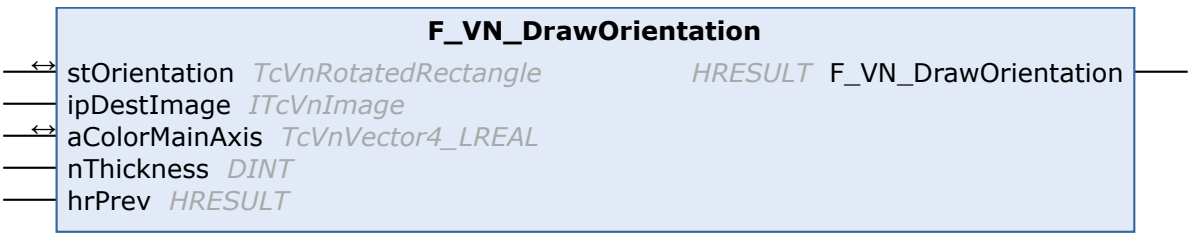
Required License

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 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
aColorMainAxis	TcVnVector4_LREAL [▶ 162]	Color of the major axis

Return value

HRESULT [▶ 141]

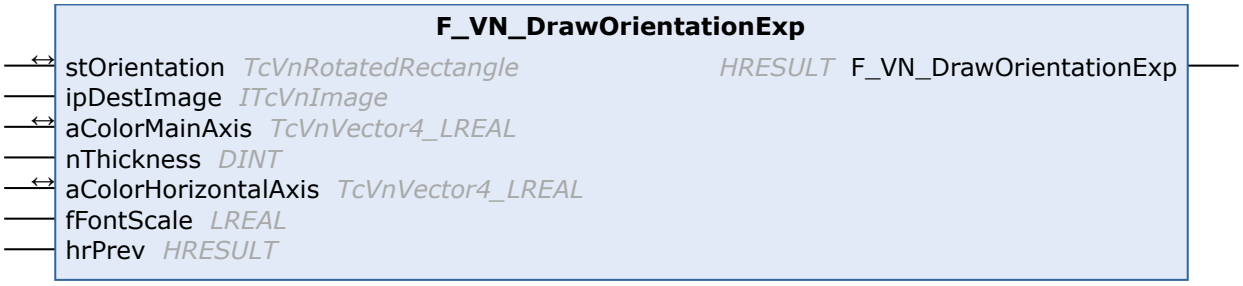
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness
fFontScale	LREAL	Scaling factor for the text
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
aColorMainAxis	TcVnVector4_LREAL [▶ 162]	Color of the major axis
aColorHorizontalAxis	TcVnVector4_LREAL [▶ 162]	Color of the horizontal axis

Return value

HRESULT [▶ 141]

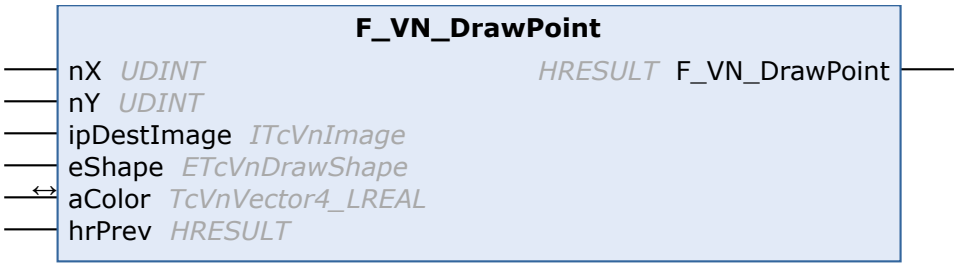
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
eShape	ETcVnDrawShape [▶ 197]	Point shape
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color (for {-1, -1, -1, -1}, a random color is chosen)

Return value

HRESULT [▶ 141]

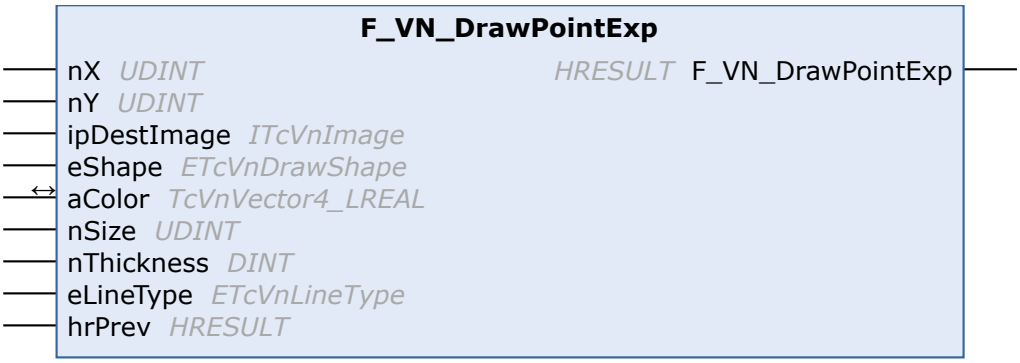
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
eShape	ETcVnDrawShape [▶ 197]	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 [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color (for {-1, -1, -1, -1}, a random color is chosen)

Return value

HRESULT [▶ 141]

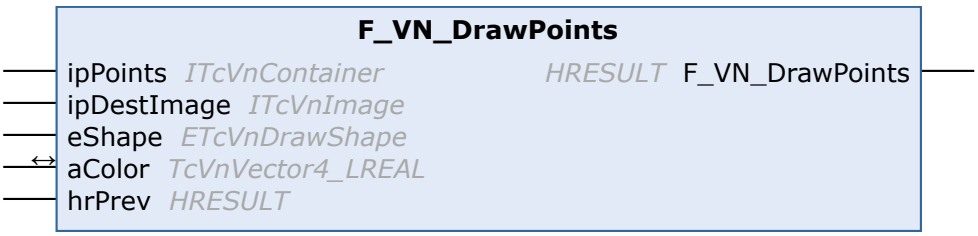
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage [▶ 411]	Destination image
eShape	ETcVnDrawShape [▶ 197]	Point shape
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 141]

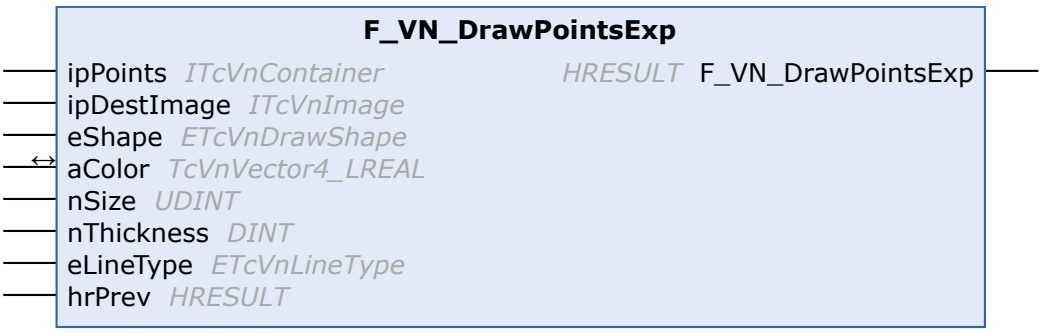
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage [▶ 411]	Destination image
eShape	ETcVnDrawShape [▶ 197]	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 [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 141]

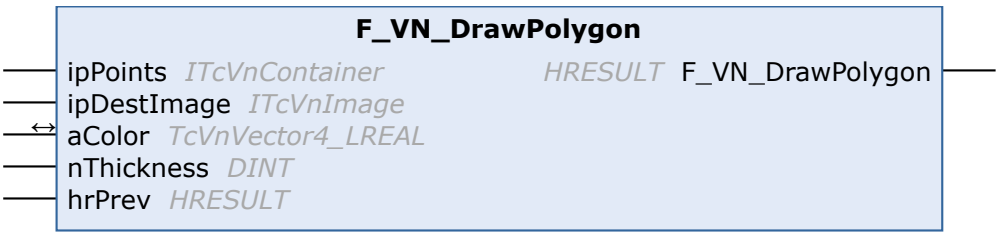
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the polygon is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

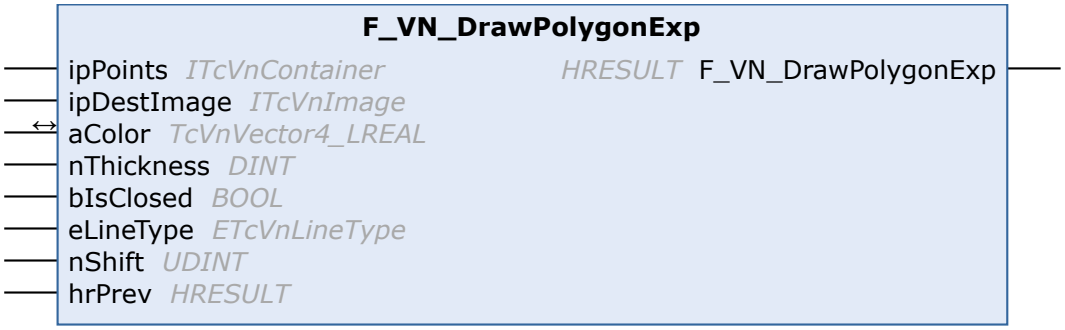
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 411]	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 [▶ 208]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

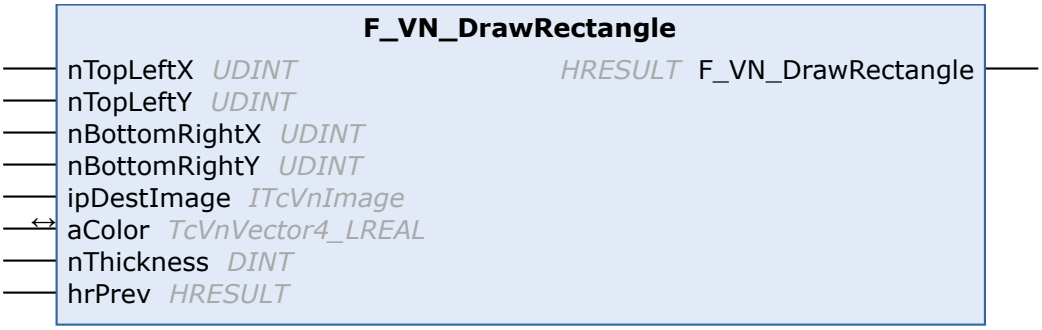
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

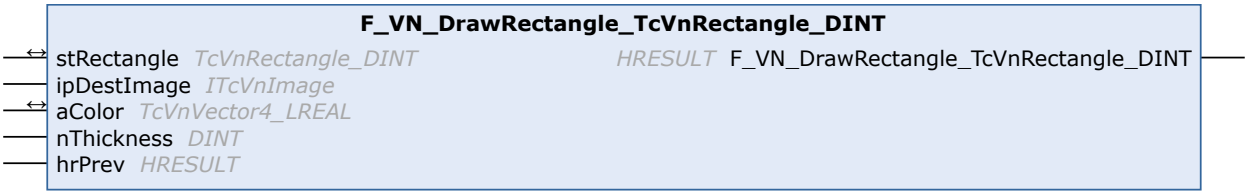
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 244]	Rectangle to be drawn
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_UDINT [▶ 245]	Rectangle to be drawn
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

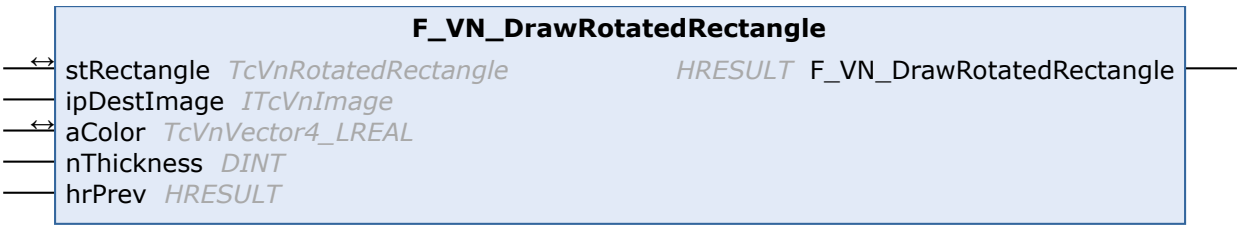
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 245]	Rectangle to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

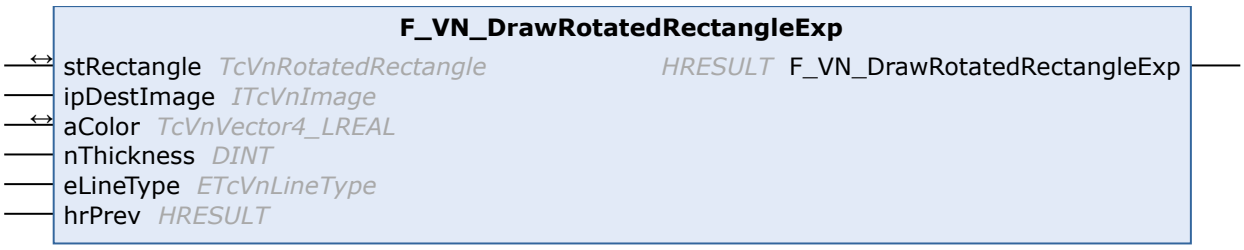
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 245]	Rectangle to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

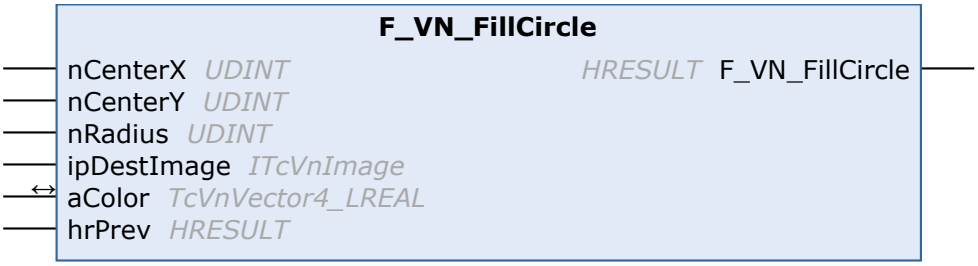
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Ellipse to be painted (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

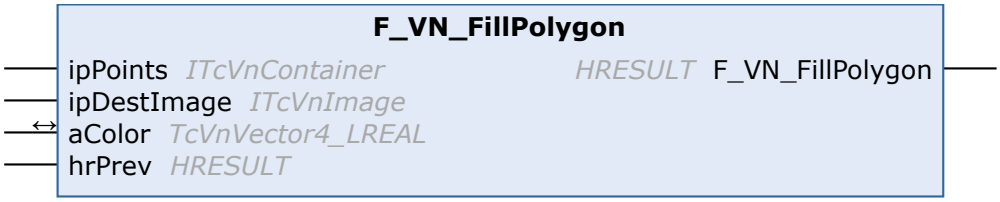
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

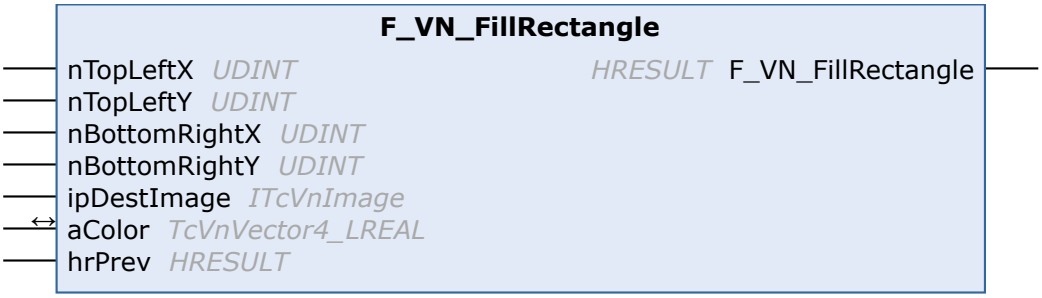
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 245]	Rectangle to be painted (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 162]	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (USINT, 1 channel. x: position, y: intensity, origin: bottom left. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 160]	End point of the line segment

Return value

HRESULT [▶ 141]

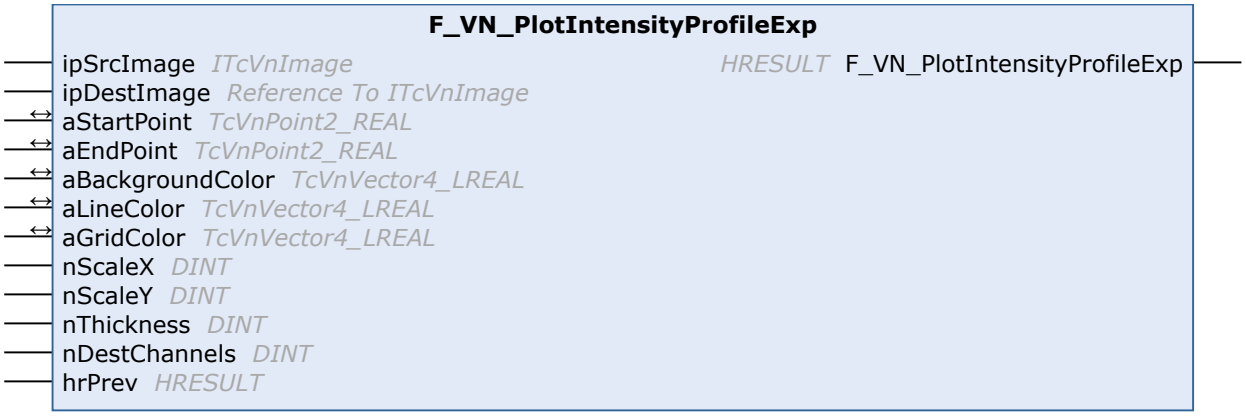
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 160]	End point of the line segment
aBackgroundColor	TcVnVector4_LREAL [▶ 162]	Background color
aLineColor	TcVnVector4_LREAL [▶ 162]	Line color
aGridColor	TcVnVector4_LREAL [▶ 162]	Grid color

Return value

HRESULT [▶ 141]

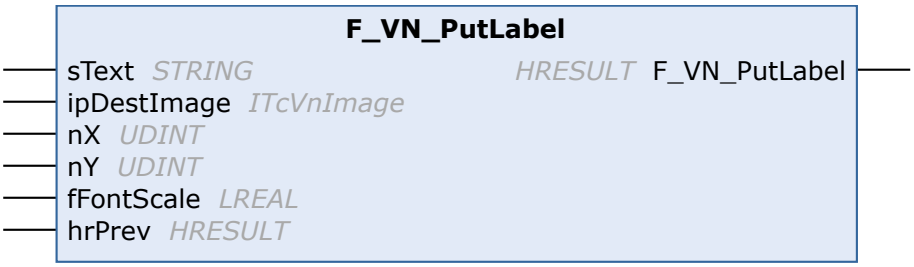
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
fFontScale	LREAL	Scaling factor
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

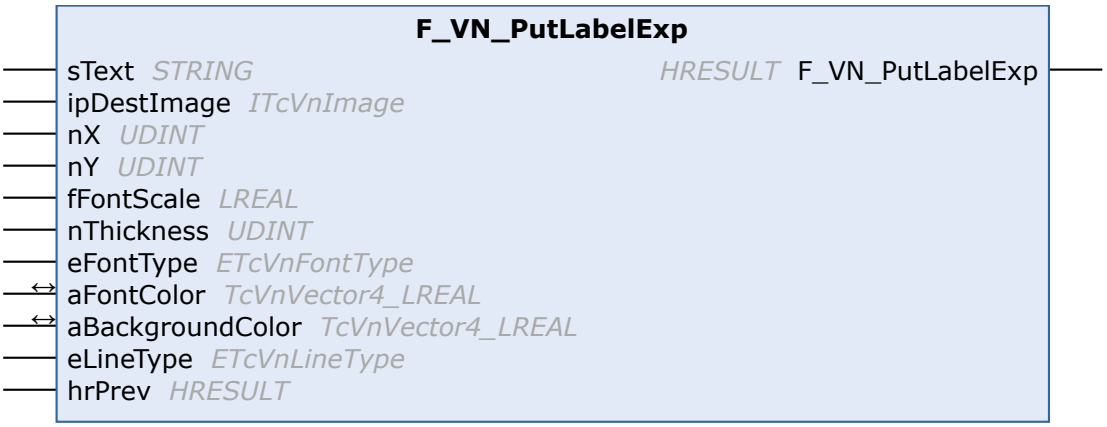
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
fFontScale	LREAL	Scaling factor
nThickness	UDINT	Line thickness
eFontType	ETcVnFontType [▶ 203]	Font type
eLineType	ETcVnLineType [▶ 208]	Line type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFontColor	TcVnVector4_LREAL [▶ 162]	Font color
aBackgroundColor	TcVnVector4_LREAL [▶ 162]	Background color

Return value

HRESULT [▶ 141]

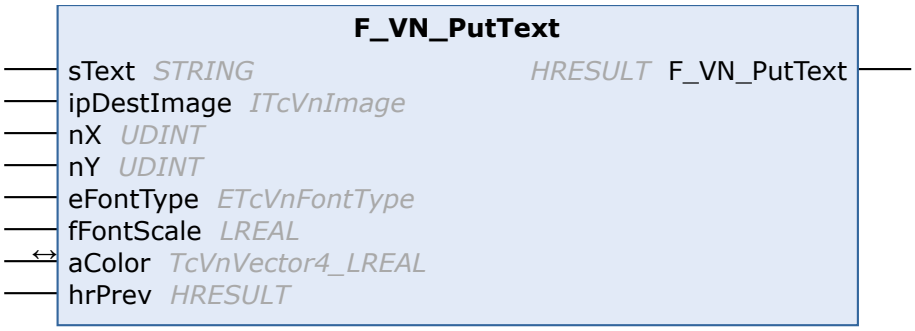
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 203]	Font type
fFontScale	LREAL	Scaling factor
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Text color

Return value

HRESULT [▶ 141]

Further information

The following fonts are available:



Related functions

- F_VN_PutLabelExp [▶ 1052]
- F_VN_PutText(Exp) [▶ 1053]

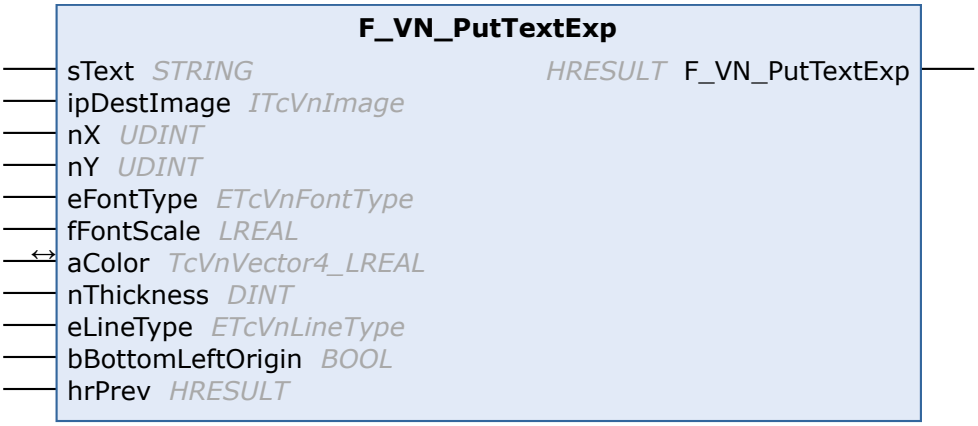
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 203]	Font type
fFontScale	LREAL	Scaling factor
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 208]	Line type
bBottomLeftOrigin	BOOL	Sets the image origin to the bottom left corner, if true
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 162]	Text color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12 Fourier Analysis

This group contains functions for frequency analysis and frequency filtering in images.

Functions

Butterworth Filter

- F_VN_CreateBandpassButterworthFilter [▶ 1057]
- F_VN_CreateBandrejectButterworthFilter [▶ 1059]
- F_VN_CreateHighpassButterworthFilter [▶ 1062]
- F_VN_CreateLowpassButterworthFilter [▶ 1064]

Gaussian filter

- F_VN_CreateBandpassGaussianFilter [▶ 1058]
- F_VN_CreateBandrejectGaussianFilter [▶ 1060]
- F_VN_CreateHighpassGaussianFilter [▶ 1063]
- F_VN_CreateLowpassGaussianFilter [▶ 1065]

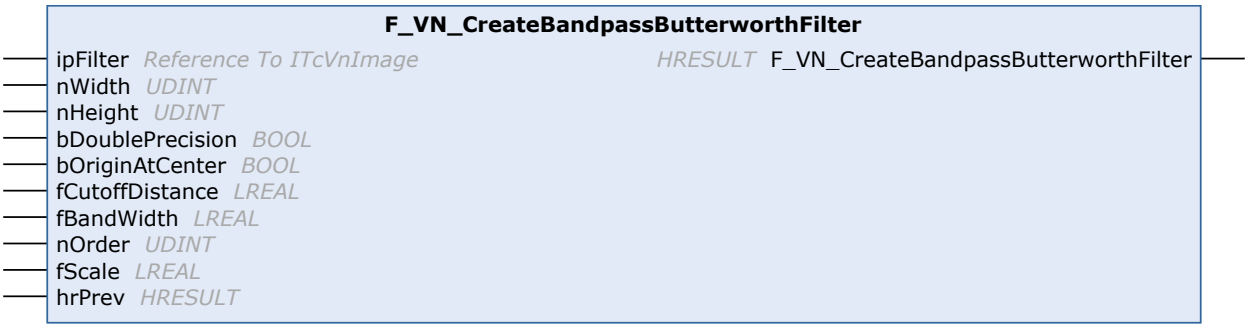
Discrete Fourier Transformation

- F_VN_Dft [▶ 1066]
- F_VN_InverseDft [▶ 1067]
- F_VN_OptimalDftSize [▶ 1068]

Miscellaneous

- F_VN_PadImageBorder(Exp) [▶ 1069]

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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

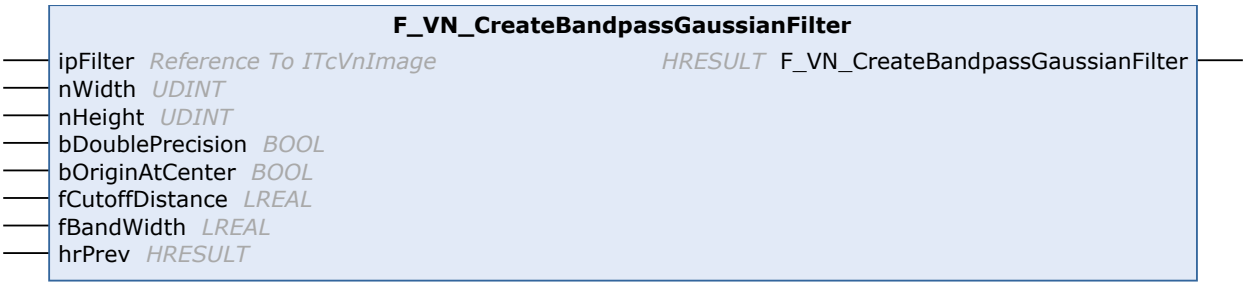
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

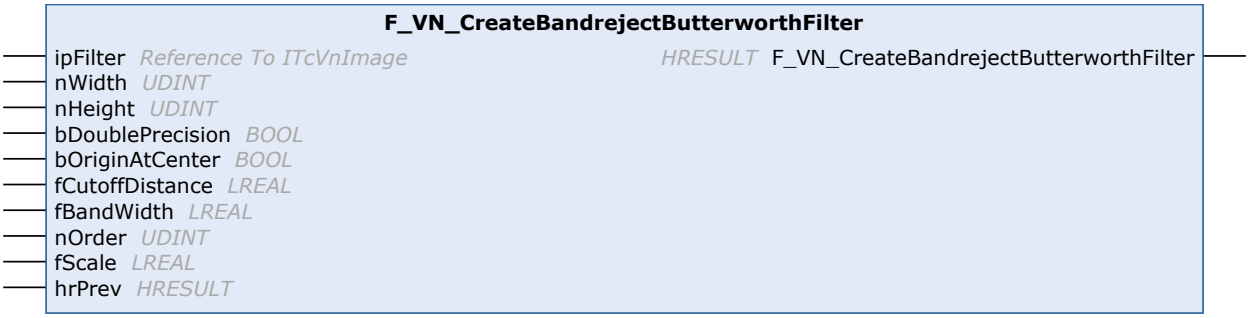
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

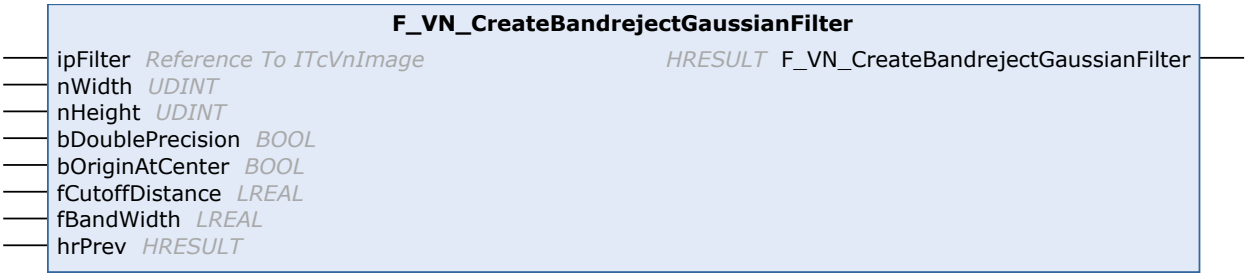
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

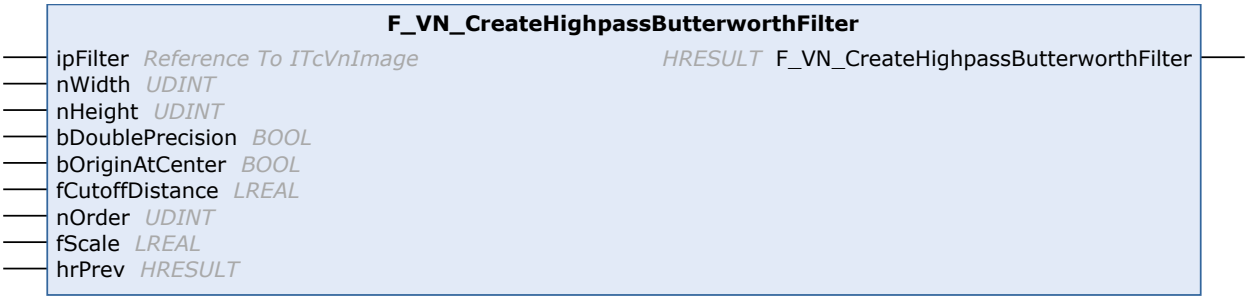
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

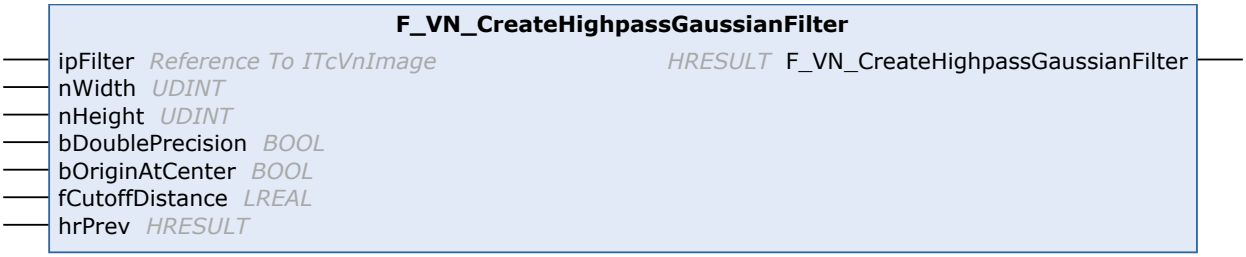
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

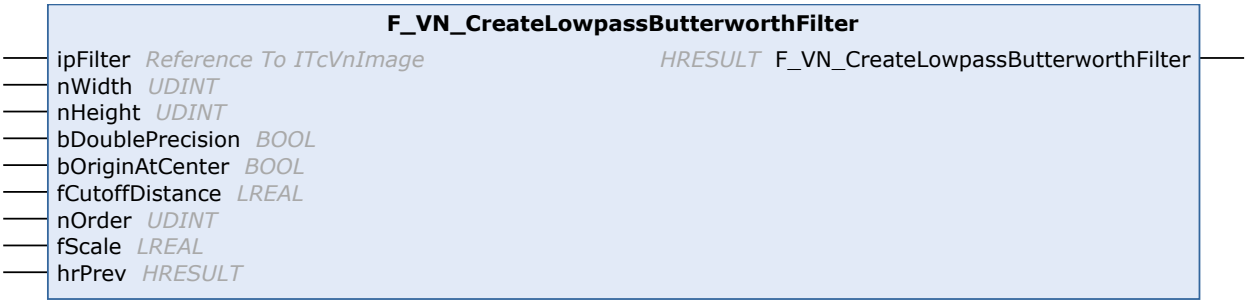
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

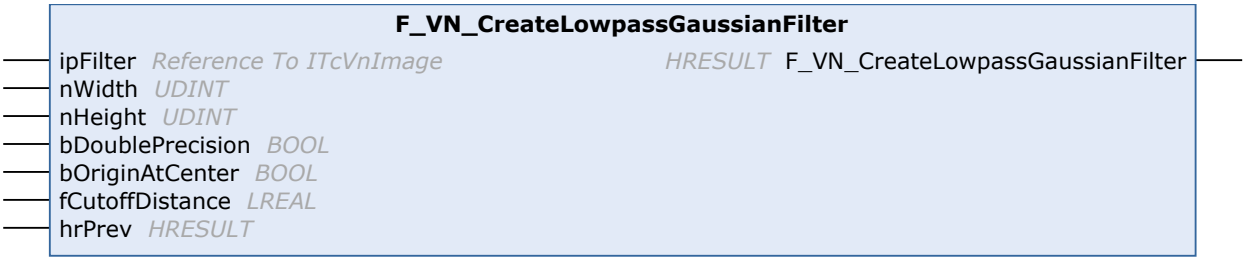
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

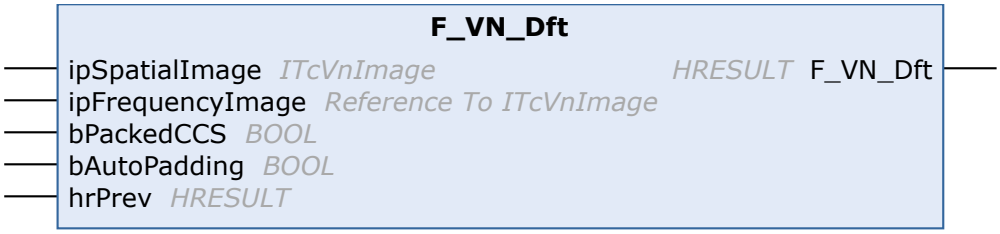
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (elements of type REAL or LREAL, 1 (Re) or 2 (Re + Im) channels)
ipFrequencyImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source frequency image (elements of type REAL or LREAL, 1 (packed CCS) or 2 (Re + Im) channels)
ipSpatialImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

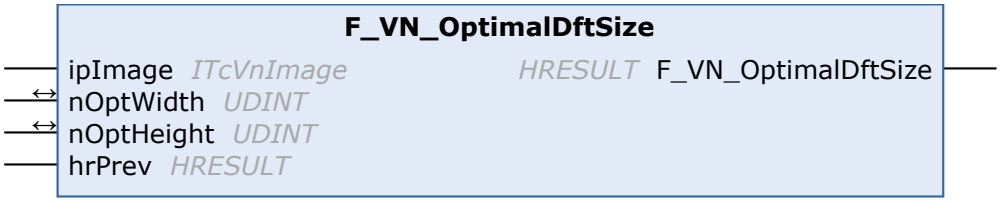
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image for which to compute the optimal width and height
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.12 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 [▶ 411]	Source image
ipPaddedImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

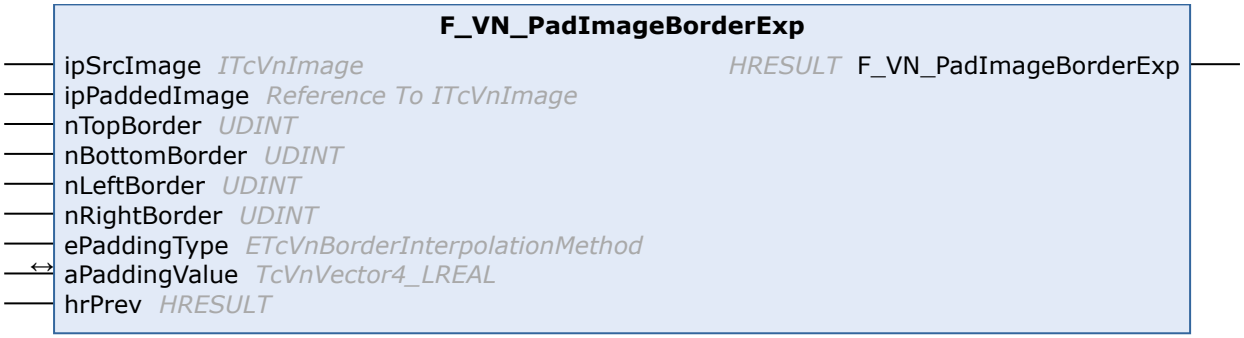
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.13 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 [▶ 411]	Source image
ipPaddedImage	Reference To ITcVnImage [▶ 411]	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 [▶ 166]	Specifies how the pixel values of the padding area are determined
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPaddingValue	TcVnVector4_LREAL [▶ 162]	Specifies the padding value if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13 Geometric and Coordinate Transformations

This group contains functions for transforming images and pixels.

Functions

Calibration [▶ 1072]

- Intrinsic and extrinsic calibration
- Compensation of optical distortions
- Transformation between image and world coordinates

Affine Transformation

- F_VN_ApplyRotationToAffineTransformation(Exp) [▶ 1115]
- F_VN_ApplyScalingToAffineTransformation(Exp) [▶ 1117]
- F_VN_ApplyTranslationToAffineTransformation(Exp) [▶ 1119]
- F_VN_ApplyYAxisInversionToAffineTransformation(Exp) [▶ 1121]
- F_VN_DecomposeAffineTransformation(Exp) [▶ 1136]
- F_VN_GenerateAffineTransformationUnitMatrix2D [▶ 1142]
- F_VN_GetAffineTransformation [▶ 1143]
- F_VN_GetAffineTransformation2D(Exp) [▶ 1144]
- F_VN_InvertAffineTransformation [▶ 1149]
- F_VN_WarpAffine(Exp) [▶ 1165]
- F_VN_WarpAffine_Container [▶ 1166]
- F_VN_WarpAffine_Point [▶ 1167]
- F_VN_WarpAffine_Rectangle [▶ 1168]

Transformation between cartesian and polar coordinates

- F_VN_ConvertCartesianToPolarAngleImage(Exp) [▶ 1123]
- F_VN_ConvertCartesianToPolarAngles(Exp) [▶ 1124]
- F_VN_ConvertCartesianToPolarImages(Exp) [▶ 1126]
- F_VN_ConvertCartesianToPolarMagnitudeImage [▶ 1128]
- F_VN_ConvertCartesianToPolarMagnitudes [▶ 1129]
- F_VN_ConvertCartesianToPolarPoints(Exp) [▶ 1130]
- F_VN_ConvertPolarToCartesianImages(Exp) [▶ 1132]
- F_VN_ConvertPolarToCartesianPoints(Exp) [▶ 1134]

- F_VN_RemapImageToLogPolarSpace(Exp) [▶ 1153]
- F_VN_RemapImageToPolarSpace(Exp) [▶ 1156]

Simple image operations

- F_VN_FlipImage [▶ 1141]
- F_VN_PyramidDown [▶ 1151]
- F_VN_PyramidUp [▶ 1152]
- F_VN_ResizeImage [▶ 1159]

Image alignment and rotation

- F_VN_AlignRotatedImageRegion(Exp) [▶ 1113]
- F_VN_RotateImage(Exp) [▶ 1162]

Perspective transformation

- F_VN_DecomposeHomography(Exp) [▶ 1138]
- F_VN_GetPerspectiveTransformation [▶ 1146]
- F_VN_Homography(Exp) [▶ 1147]
- F_VN_PerspectiveTransformation [▶ 1150]
- F_VN_WarpPerspective(Exp) [▶ 1170]
- F_VN_WarpPerspective_Container [▶ 1171]
- F_VN_WarpPerspective_Point [▶ 1172]
- F_VN_WarpPerspective_Rectangle [▶ 1173]

6.1.4.13.1 Calibration

This subgroup contains functions for the correlation between pixels and real world pixels. They are usually used together with the measurement functions [▶ 1481].

Functions

Calibration

- F_VN_CalibrateCamera(Exp) [▶ 1073]
- F_VN_CalibrateCameraManually(Exp) [▶ 1083]
- F_VN_CalibrateCameraPlanar(Exp) [▶ 1086]
- F_VN_CalibrateLinescanCamera(Exp) [▶ 1088]
- F_VN_DetectPatternPoints(Exp) [▶ 1098]
- F_VN_GenerateCalibrationPatternReferencePoints [▶ 1101]
- F_VN_SortDetectedPatternPoints [▶ 1106]
- F_VN_SolvePnP(Exp) [▶ 1103]

Compensation of lens distortion

- F_VN_CompensateLensDistortion(Exp) [▶ 1092]
- F_VN_CompensateLensDistortionForPoints(Exp) [▶ 1095]

Transformation between image and world coordinates

- F_VN_TransformCoordinatesImageToWorld [▶ 1108]
- F_VN_TransformCoordinatesWorldToImage [▶ 1112]

- F_VN_TransformCoordinatesPlanar [▶ 1110]
- F_VN_ImagePointsWorldDistance [▶ 1102]

Requirements for the input images

The input images should not be used in a packed or filtered format, such as Bayer format. In the case of the calibration functions, this has the consequence that the recognition of the patterns is disturbed, which can lead to poorer calibration results or no calibration results at all. Since an interpolation takes place during the compensation functions, in which the original arrangement of the pixels is changed, the image can no longer be converted correctly afterwards. Therefore, it is recommended to always perform a conversion first before using a function from this subgroup.

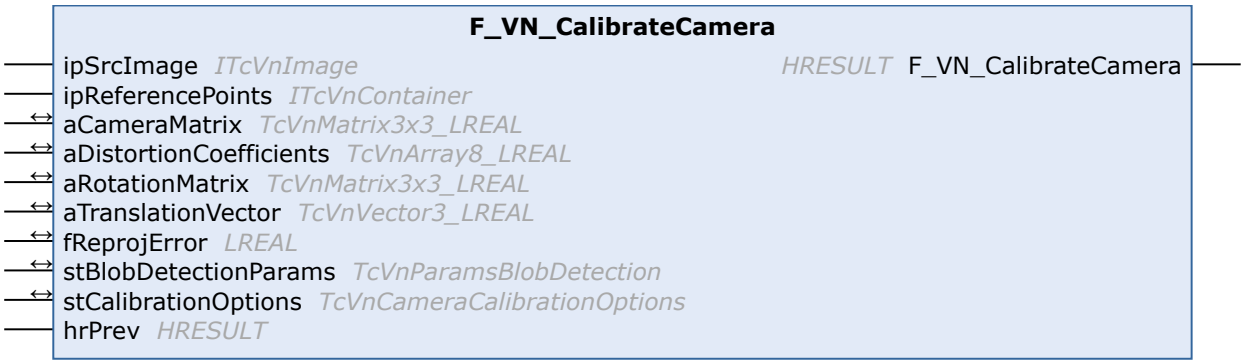
Requirements for the camera matrix

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

The camera matrix is made up of the focal lengths fx and fy and the principal point (cx,cy), which are expressed in pixel units. The principal point, which is the intersection of the optical axis with the image, is normally located near the center of the image. The following conditions must be met:

- 0 < fx and fy
- cx >= 0 and <= image width -1
- cy >= 0 and <= image height -1
- The other positions in the matrix must be filled with 0 or 1, as shown in the figure. Other values are not permissible.

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 [▶ 411]	Source image containing a calibration pattern (1 channel, USINT elements)
ipReferencePoints	ITcVnContainer [▶ 370]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CalibrateCamera` calculates the intrinsic and extrinsic camera parameters on the basis of an input image containing a calibration pattern. The elements of the calibration pattern must be detectable as a blob.

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image with a bit depth of 8 bits. This image must contain a calibration pattern with circles that matches the reference points `ipReferencePoints`. The function expects black filled circles on a white background. To invert the image, the `ETcVnThresholdType = TCVN_TT_BINARY_INV` of the Blob function can be used.

The expert version F_VN_CalibrateCameraExp [▶ 1076] is required for a calibration with several input images.

Reference points

The container `ipReferencePoints` must contain the reference points that are illustrated on the input image `ipSrcImage` as the calibration pattern. The points must be specified as 3-dimensional points of the type `REAL` so that the container is of the type `ContainerType_Vector_TcVnPoint3_REAL`.

The reference points can be generated as follows:

- Manual creation of an XML file and reading with the function block FB_VN_ReadCalibrationPattern [▶ 1592].
- Configuration in the calibration assistants [▶ 107], saving in the TcCOM object by clicking **Write Results** and reading with the method GetCalibPatternRef [▶ 1629] of the function block FB_VN_GevCameraControl [▶ 1621]. The set center point is taken into account.
- Inserting reference points individually into a container.

Matrices of the camera parameters

These matrices return the calculated parameters of the camera model. They can then be used for the compensation of the objective distortions and for the conversion between pixel coordinates and world coordinates.

- Camera matrix
- Distortion coefficients
- Rotation matrix
- Translation vector

Rounding errors

On account of rounding errors, the calculated parameter values can deviate slightly from the values calculated in the calibration assistants.

Reprojection errors

Parameters for blob detection

No chessboard pattern

As the reference points are detected as a blob structure, the use of chessboard patterns is not possible.

Options for calibration

Default settings

To get the default settings from the calibration wizard [▶ 107], the bits `bFixAspectRatio`, `bFixPrincipalPoint`, `bFixK5`, `bFixK6` and `bRationalModel` must be set to TRUE.

bFixPrincipalPoint = „Fix Center Point“

The bit `bFixPrincipalPoint` corresponds to the option "Fix Center Point" in the camera calibration assistant.

Samples

- Calibration Assistant [▶ 2838]

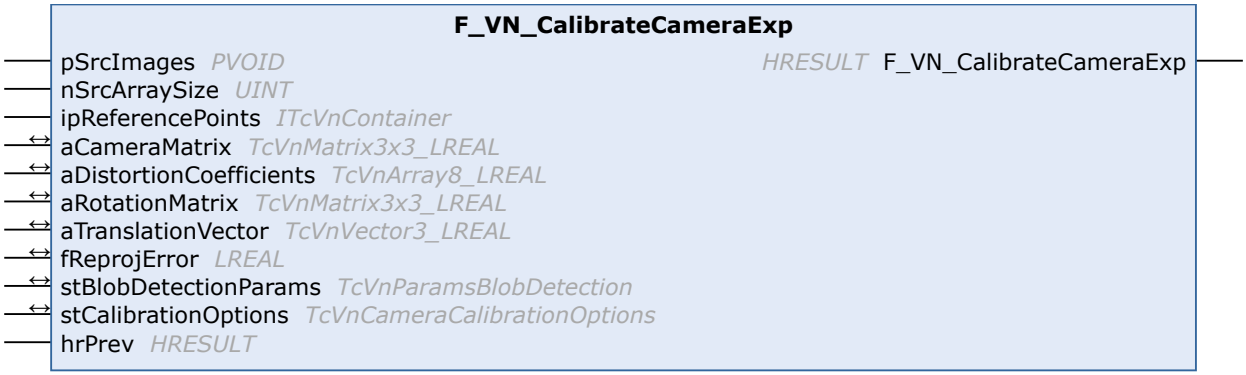
Required License

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 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 [▶ 370]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CalibrateCameraExp` is the Expert version of F_VN_CalibrateCamera [▶ 1073]. It enables the use of several input images for the calibration of the camera.

Parameter

Input images

The input images for the camera calibration are transferred to the function with the help of the pointer `pSrcImages` and the number of images `nSrcArraySize`. The individual images must otherwise meet the same requirements as for the standard function.

NOTICE

Pointer handling

Make sure that the specification of the pointer and the array size match the actual data. Otherwise an impermissible memory access may occur, resulting in a system crash.

Reference points

The container `ipReferencePoints` must contain the reference points that are illustrated on the input image `ipSrcImage` as the calibration pattern. The points must be specified as 3-dimensional points of the type `REAL` so that the container is of the type `ContainerType_Vector_TcVnPoint3_REAL`.

The reference points can be generated as follows:

- Manual creation of an XML file and reading with the function block FB_VN_ReadCalibrationPattern [▶ 1592].
- Configuration in the calibration assistants [▶ 107], saving in the TcCOM object by clicking **Write Results** and reading with the method GetCalibPatternRef [▶ 1629] of the function block FB_VN_GevCameraControl [▶ 1621]. The set center point is taken into account.
- Inserting reference points individually into a container.

Matrices of the camera parameters

These matrices return the calculated parameters of the camera model. They can then be used for the compensation of the objective distortions and for the conversion between pixel coordinates and world coordinates.

- Camera matrix
- Distortion coefficients
- Rotation matrix
- Translation vector

Rounding errors

On account of rounding errors, the calculated parameter values can deviate slightly from the values calculated in the calibration assistants.

Reprojection errors

Parameters for blob detection

No chessboard pattern

As the reference points are detected as a blob structure, the use of chessboard patterns is not possible.

Options for calibration

Default settings

To get the default settings from the calibration wizard [▶ 107], the bits `bFixAspectRatio`, `bFixPrincipalPoint`, `bFixK5`, `bFixK6` and `bRationalModel` must be set to TRUE.

bFixPrincipalPoint = „Fix Center Point“

The bit `bFixPrincipalPoint` corresponds to the option "Fix Center Point" in the camera calibration assistant.

Application

Samples

- Calibration Assistant [▶ 2838]

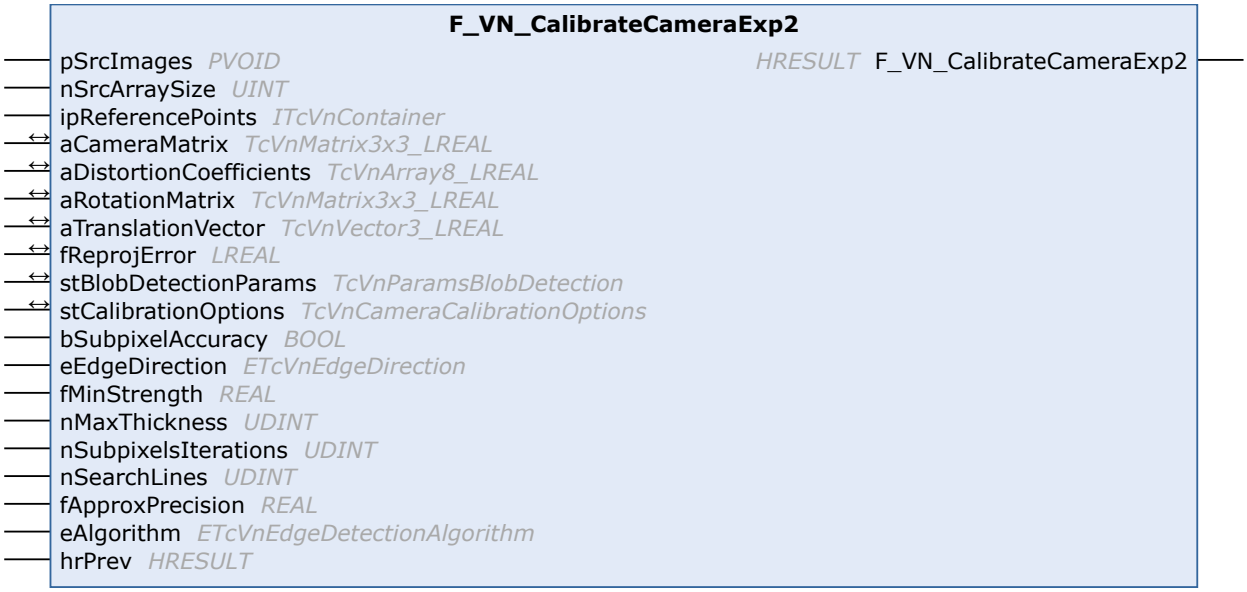
Required License

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 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 [▶ 370]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
bSubpixelAccuracy	BOOL	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

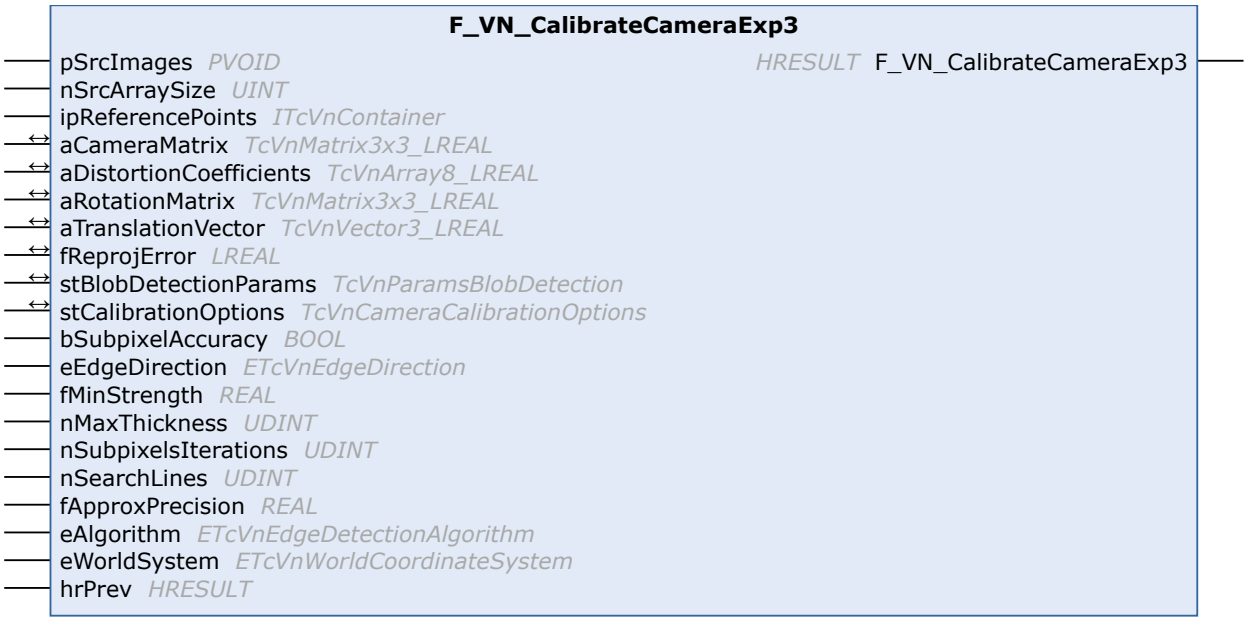
Required License

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 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 [▶ 370]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
bSubpixelAccuracy	BOOL	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 222]	World coordinate system orientation
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

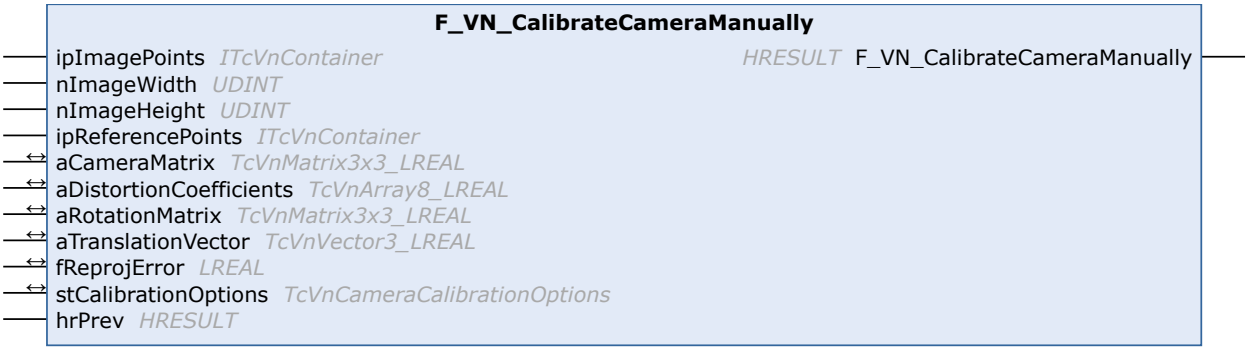
Required License

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 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 [▶ 370]	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 [▶ 370]	Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector (for the first inner container)
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

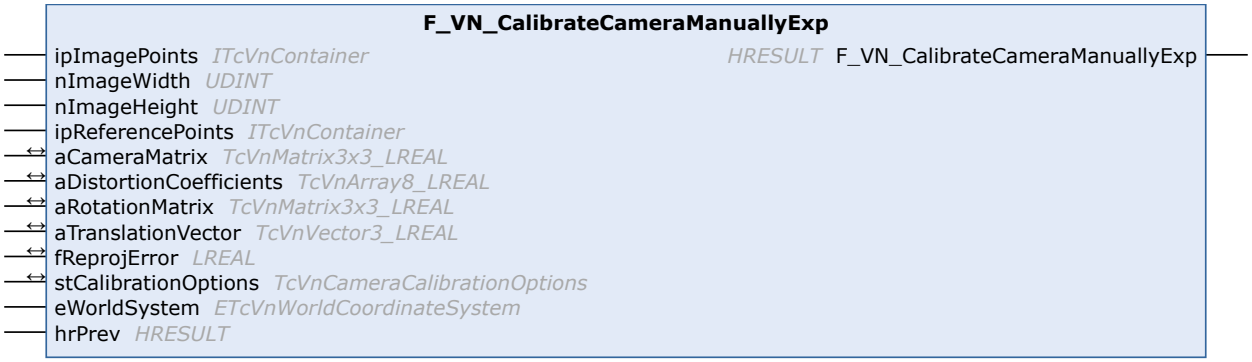
Required License

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 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 [▶ 370]	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 [▶ 370]	Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 222]	World coordinate system orientation
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector (for the first inner container)
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options

Return value

HRESULT [▶ 141]

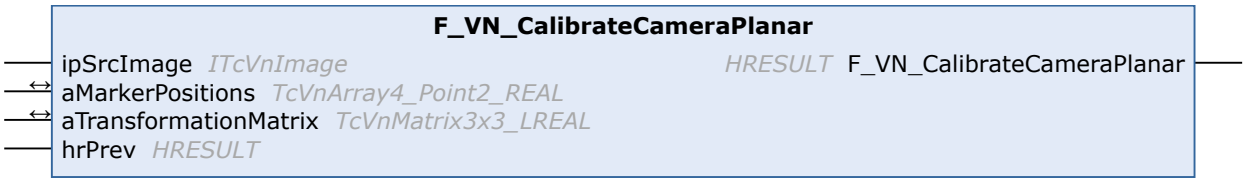
Required License

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 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 [▶ 411]	Binary source image (background zero, markers non-zero)
hrPrev	HRESULT [▶ 141]	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 [▶ 162]	Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system

Return value

HRESULT [▶ 141]

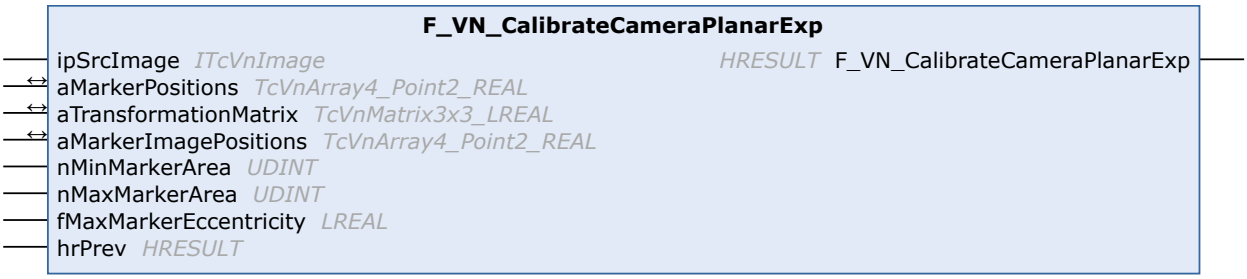
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	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 [▶ 162]	Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system
aMarkerImagePositions	TcVnArray4_Point2_REAL [▶ 162]	Returns the marker positions in image coordinates (left-handed coordinate system)

Return value

HRESULT [▶ 141]

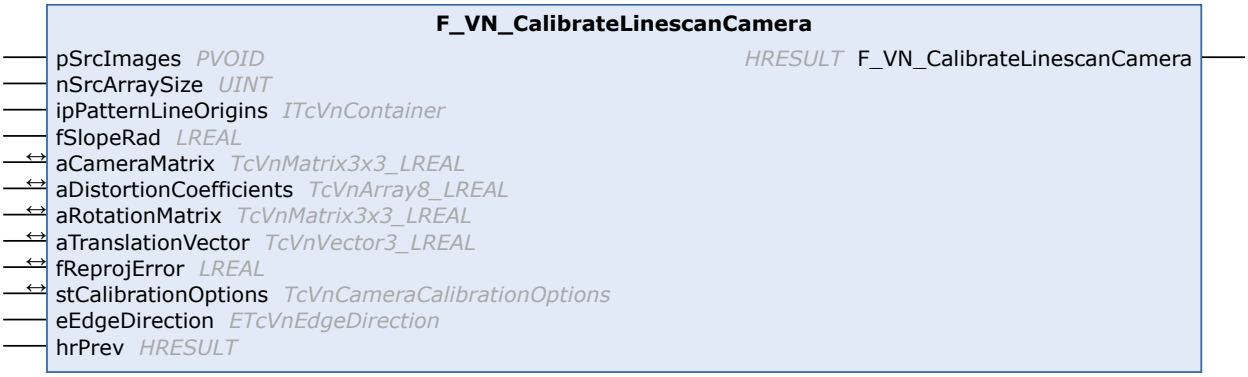
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 199]	Specification of the edge direction to search for
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)

Return value

HRESULT [▶ 141]

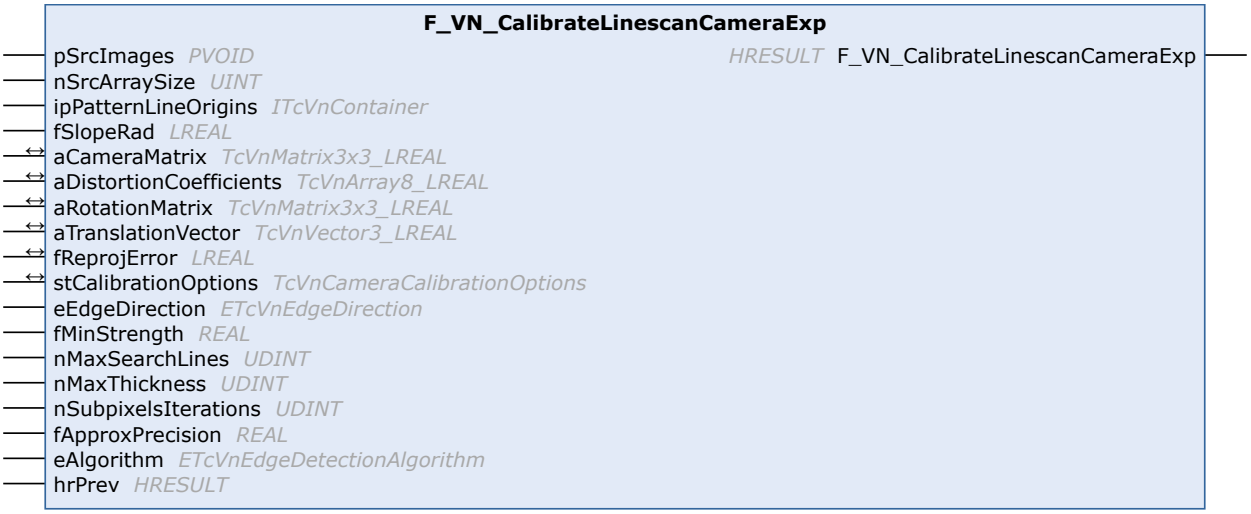
Required License

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 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 [▶ 370]	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 [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 226]	Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)

Return value

HRESULT [▶ 141]

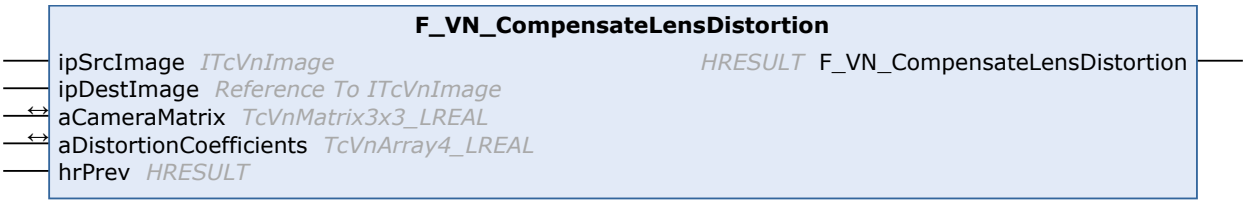
Required License

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Allows additional scaling and shifting of the result image

Return value

HRESULT [▶ 141]

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 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 [▶ 370]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 141]

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 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 [▶ 370]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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 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 [▶ 370]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Allows additional shifting of the result point coordinates

Return value

HRESULT [▶ 141]

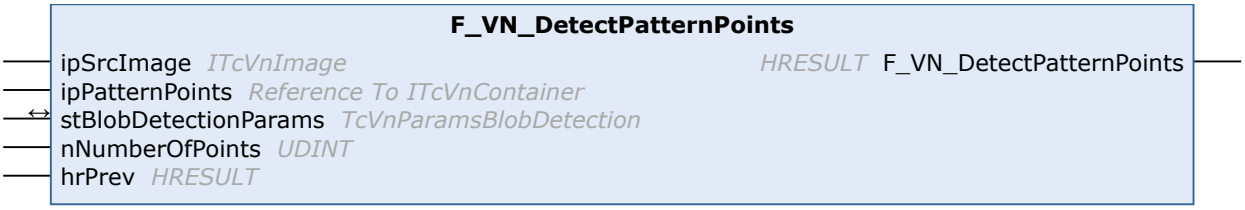
Required License

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 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 [▶ 411]	Source image (USINT, 1 channel)
ipPatternPoints	Reference To ITcVnContainer [▶ 370]	Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
nNumberOfPoints	UDINT	Expected number of pattern points
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters to detect and filter contours.

Return value

HRESULT [▶ 141]

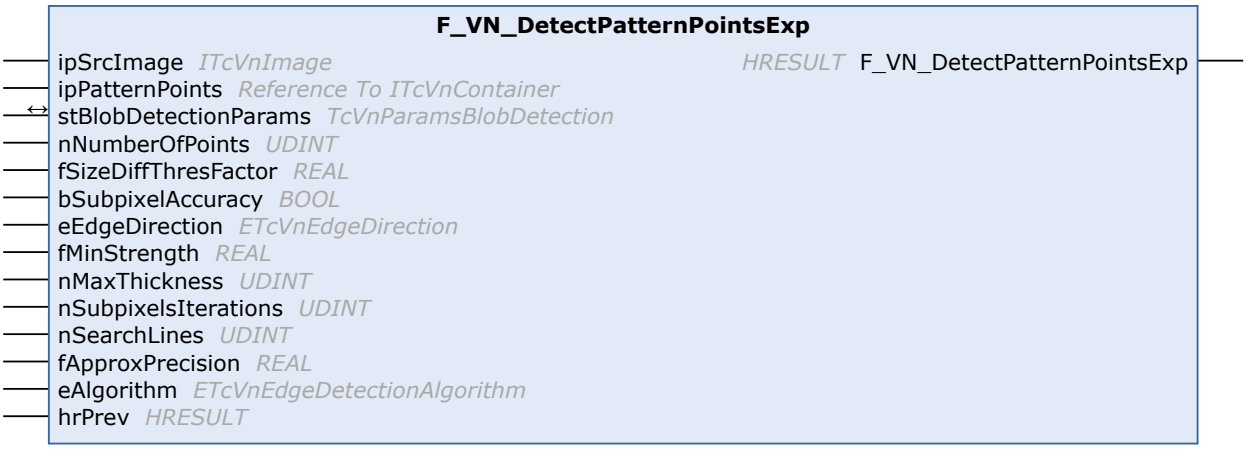
Required License

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 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 [▶ 411]	Source image (USINT, 1 channel)
ipPatternPoints	Reference To ITcVnContainer [▶ 370]	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 [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 234]	Parameters to detect and filter contours.

Return value

HRESULT [▶ 141]

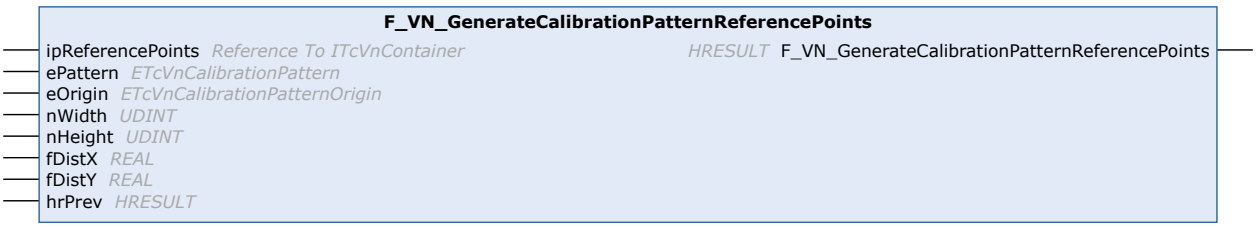
Required License

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 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 [▶ 370]	Returns the generated reference points (ContainerType_Vector_TcVnPoint3_REAL)
ePattern	ETcVnCalibrationPattern [▶ 167]	Calibration pattern type
eOrigin	ETcVnCalibrationPatternOrigin [▶ 168]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

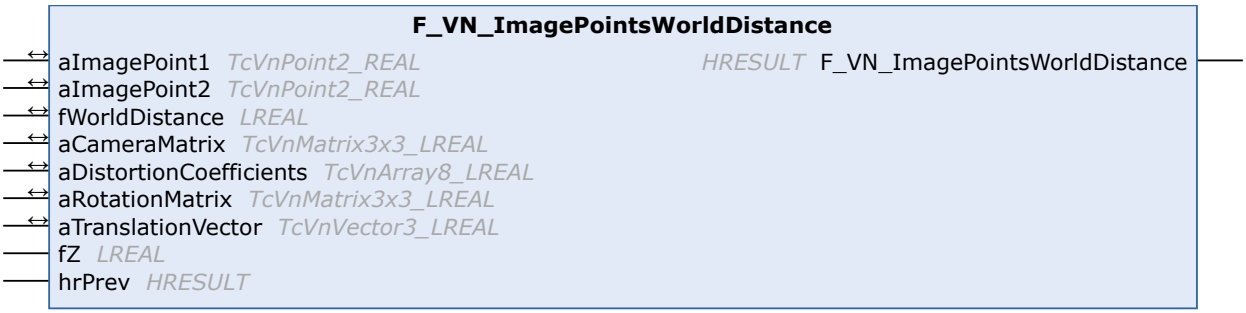
Required License

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aImagePoint1	TcVnPoint2_REAL [▶ 160]	First point in the image coordinate system
aImagePoint2	TcVnPoint2_REAL [▶ 160]	Second point in the image coordinate system
fWorldDistance	LREAL	Returns the distance between aImagePoint1 and aImagePoint2 in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

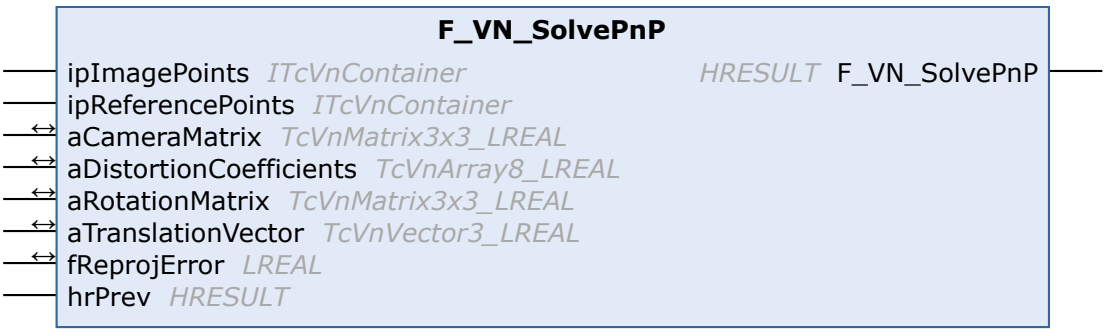
Required License

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 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 [▶ 370]	Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer [▶ 370]	Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error

Return value

HRESULT [▶ 141]

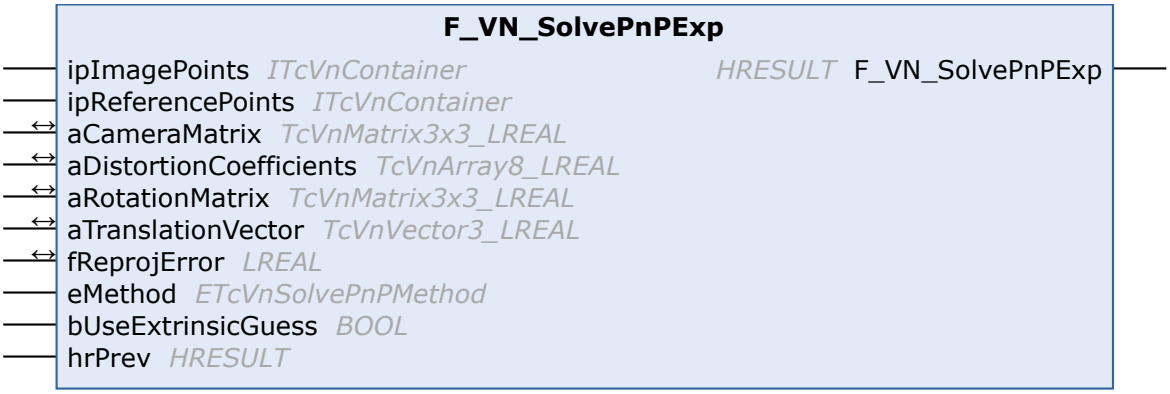
Required License

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 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 [▶ 370]	Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer [▶ 370]	Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
eMethod	ETcVnSolvePnPMethod [▶ 216]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error

Return value

HRESULT [▶ 141]

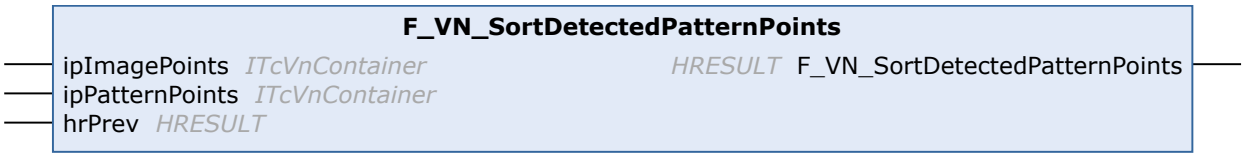
Required License

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 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 [▶ 370]	Detected pattern point positions in the image coordinate system (will be sorted by this function, ContainerType_Vector_TcVnPoint2_REAL)
ipPatternPoints	ITcVnContainer [▶ 370]	The pattern points in the calibration pattern coordinate system, used as a reference to sort ipImagePoints (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

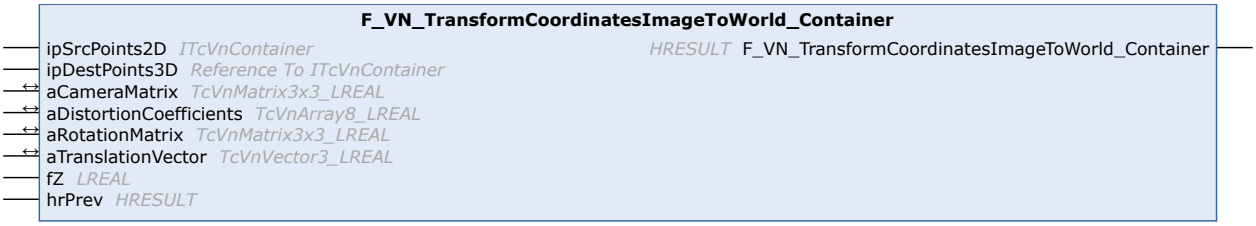
Required License

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 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 [▶ 370]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints3D	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

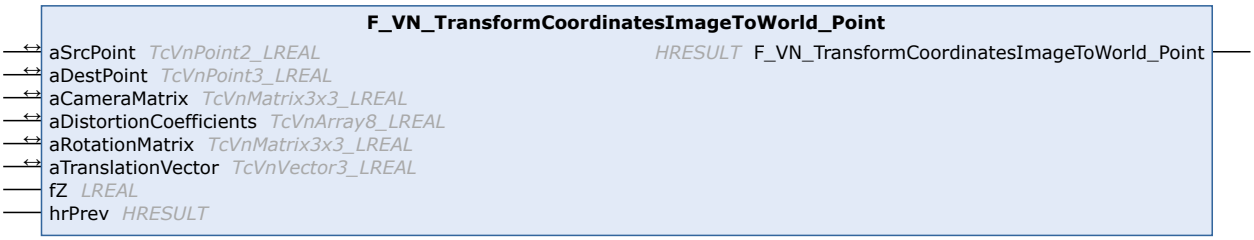
Required License

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 160]	Point in the image coordinate system
aDestPoint	TcVnPoint3_LREAL [▶ 160]	Returns the point in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

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 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 [▶ 370]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns the transformed 2D points (TcVnPoint2_LREAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 141]

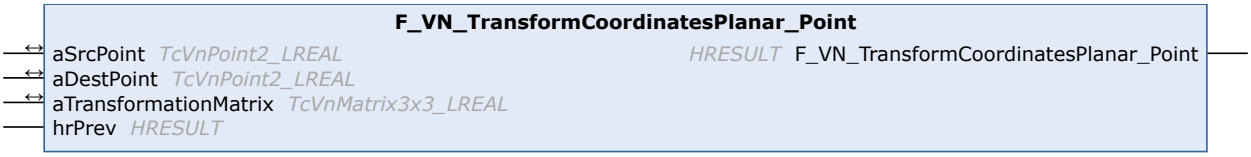
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 160]	Point in the source coordinate system
aDestPoint	TcVnPoint2_LREAL [▶ 160]	Returns point in the destination coordinate system
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 141]

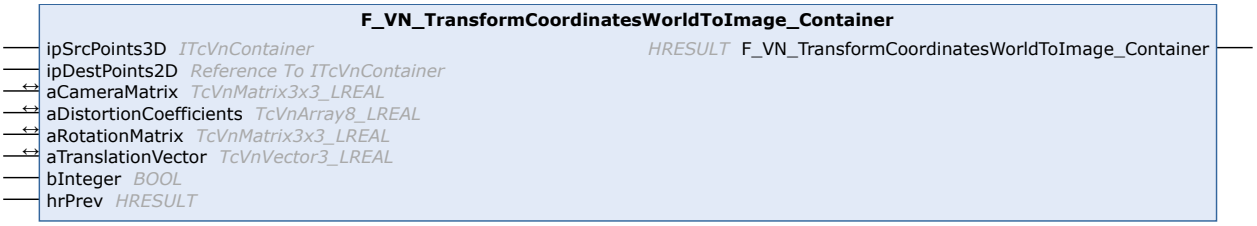
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 3D source points (TcVnPoint3_LREAL)
ipDestPoints2D	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

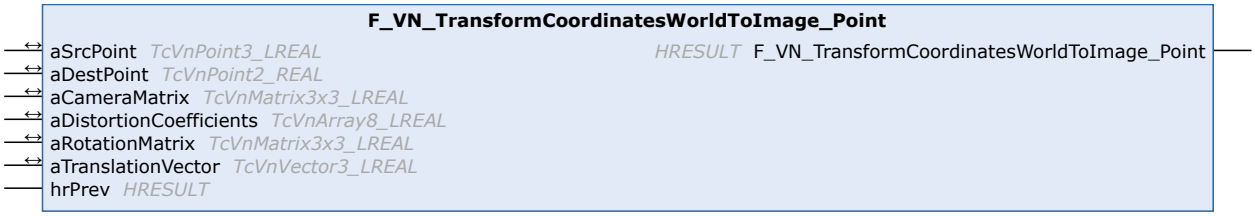
Required License

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint3_LREAL [▶ 160]	3D source point (world coordinate system)
aDestPoint	TcVnPoint2_REAL [▶ 160]	Returns the transformed 2D point (image coordinate system)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

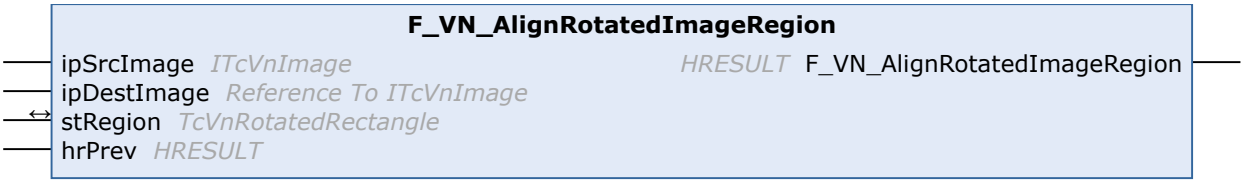
Required License

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRegion	TcVnRotatedRectangle [▶ 245]	Image region to be aligned

Return value

HRESULT [▶ 141]

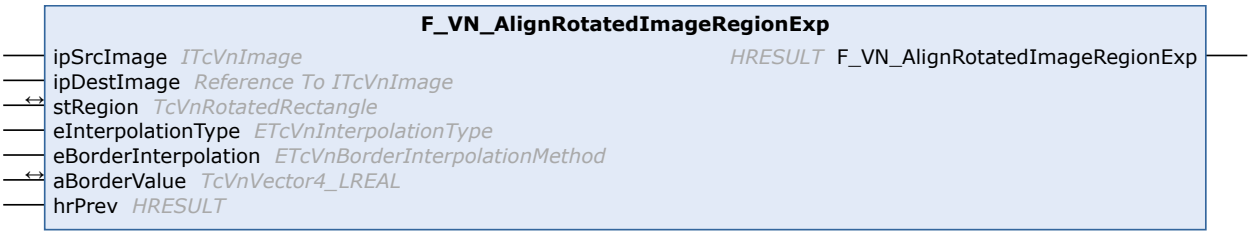
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
eInterpolationType	ETcVnInterpolationType [▶ 206]	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 166]	Border interpolation method (ISOLATED not supported)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRegion	TcVnRotatedRectangle [▶ 245]	Image region to be aligned
aBorderValue	TcVnVector4_LREAL [▶ 162]	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the rotation is applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the rotation is applied

Return value

HRESULT [▶ 141]

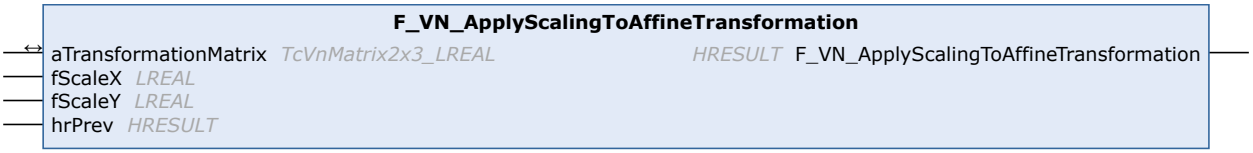
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the scaling is applied

Return value

HRESULT [▶ 141]

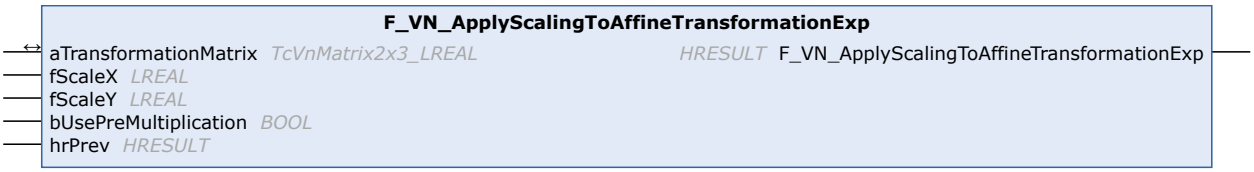
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the scaling is applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the translation is applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the translation is applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the inversion of the y-axis direction is applied

Return value

HRESULT [▶ 141]

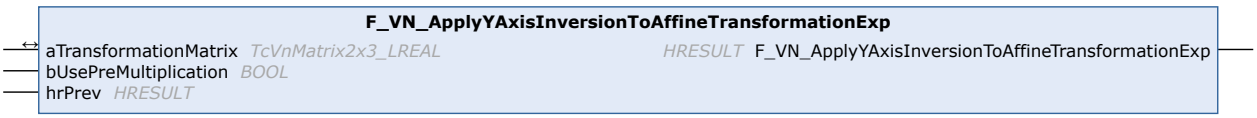
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix to which the inversion of the y-axis direction is applied

Return value

HRESULT [▶ 141]

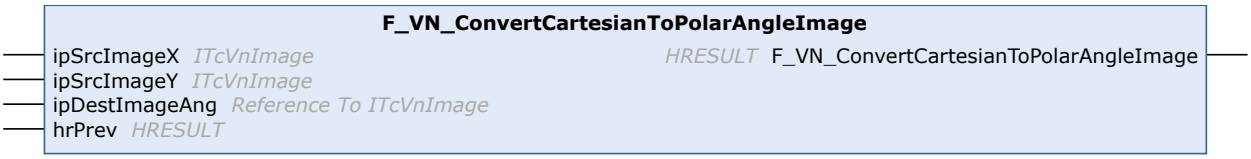
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 411]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	Reference To ITcVnImage [▶ 411]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 411]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

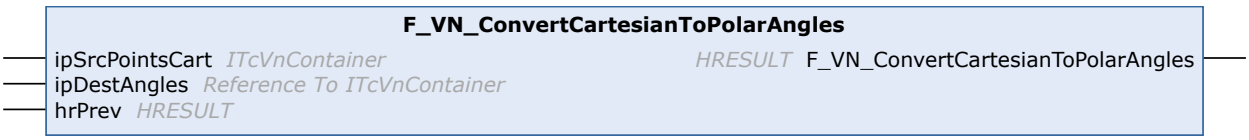
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

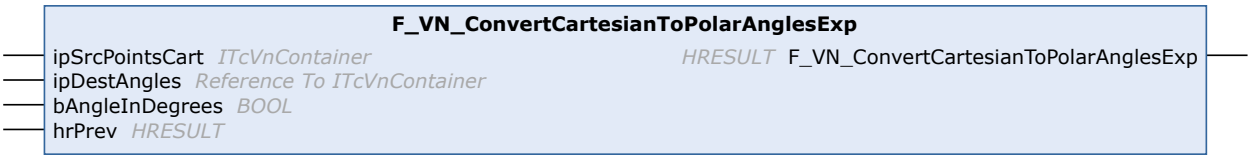
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

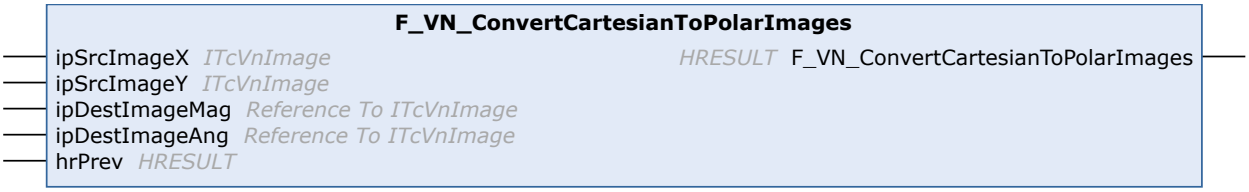
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 411]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 411]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	Reference To ITcVnImage [▶ 411]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 411]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 411]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

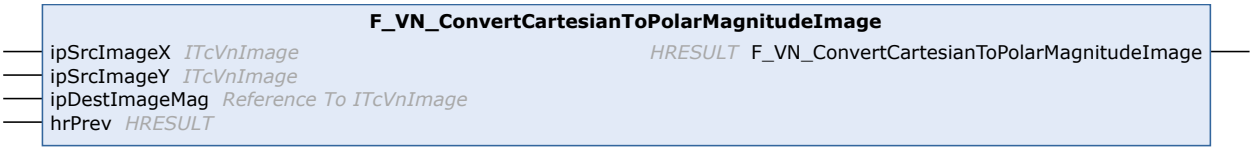
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 411]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 411]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

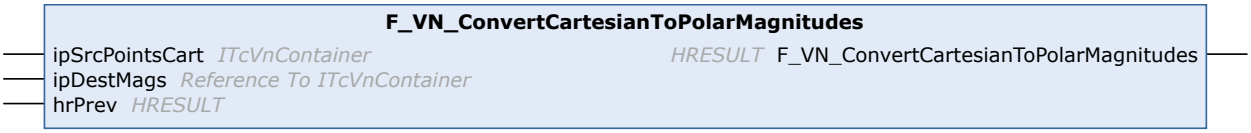
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestMags	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

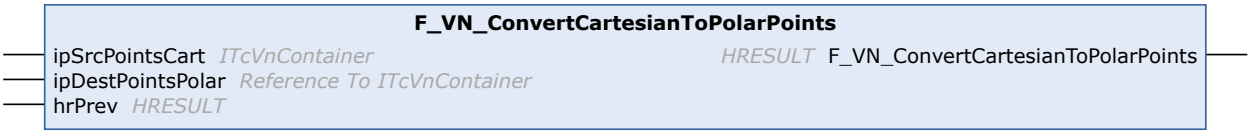
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

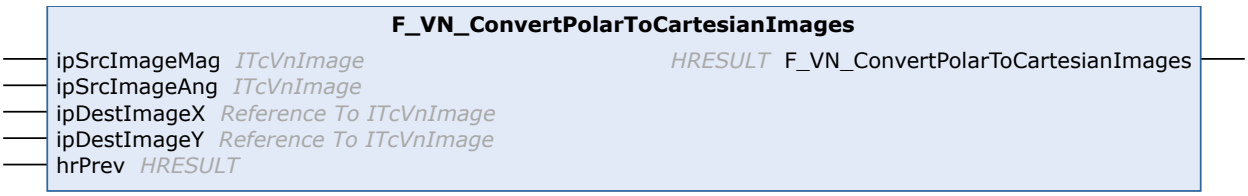
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage [▶ 411]	Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	Reference To ITcVnImage [▶ 411]	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 [▶ 411]	Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

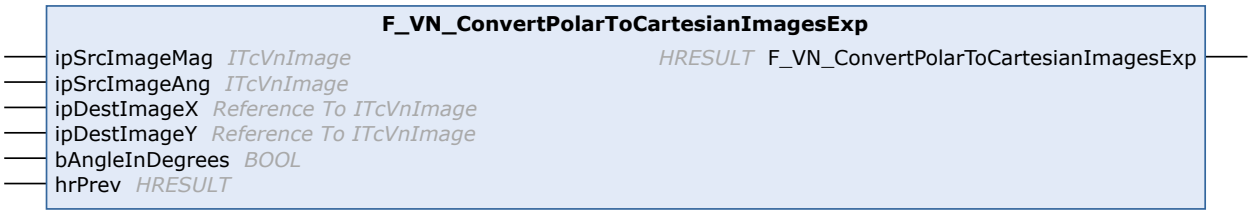
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage [▶ 411]	Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	Reference To ITcVnImage [▶ 411]	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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

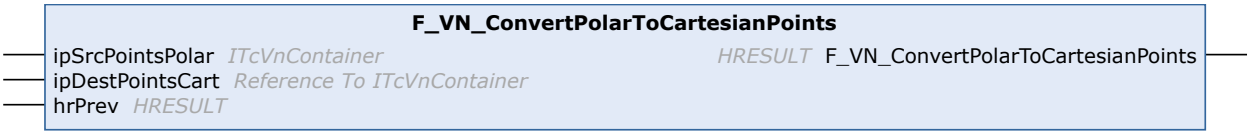
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix
fRotationAngle	LREAL	Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)

Return value

HRESULT [▶ 141]

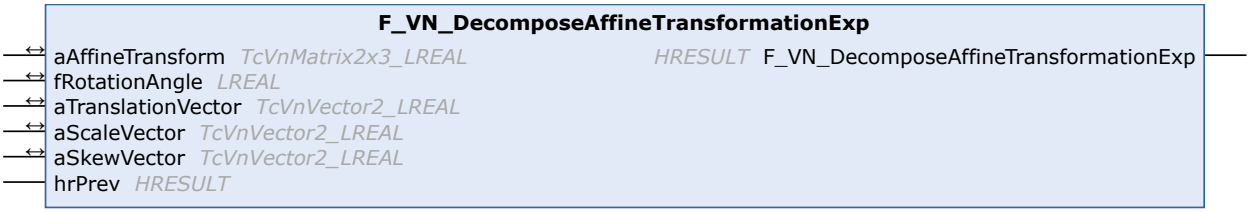
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix
fRotationAngle	LREAL	Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)
aTranslationVector	TcVnVector2_LREAL [▶ 162]	2D translation vector. The values are in pixels
aScaleVector	TcVnVector2_LREAL [▶ 162]	2D scale vector
aSkewVector	TcVnVector2_LREAL [▶ 162]	2D skew vector. The values are the skew angles in x and y directions in radian

Return value

HRESULT [▶ 141]

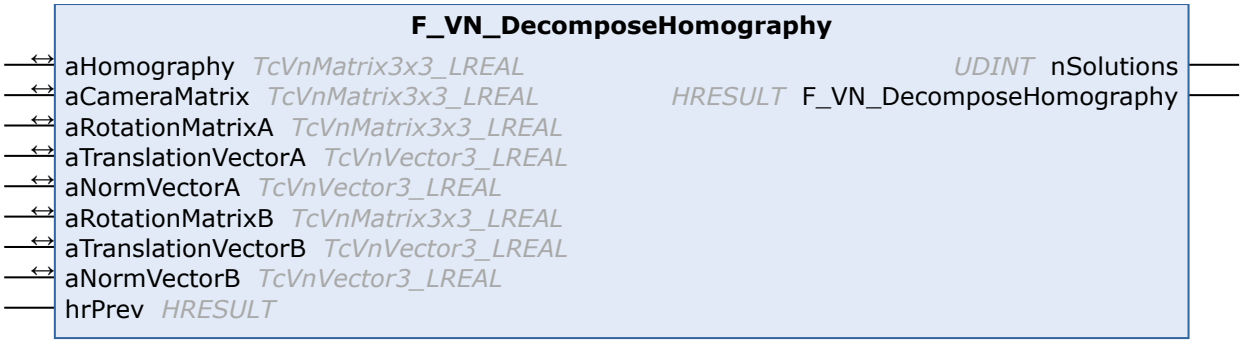
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aHomography	TcVnMatrix3x3_LREAL [▶ 162]	Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 162]	Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 162]	Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 162]	Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 162]	Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 162]	Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 162]	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 [▶ 141]

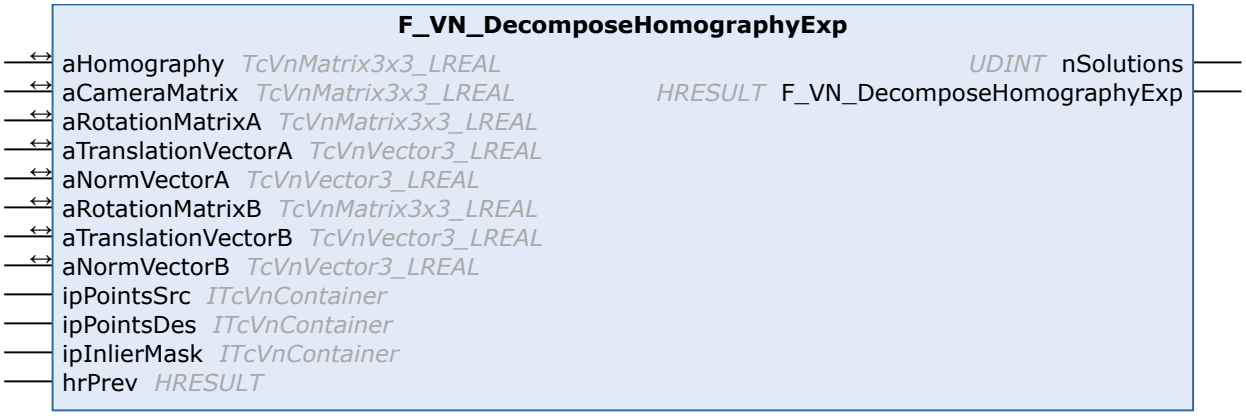
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with source points (ContainerType_Vector_TcVnPoint2_REAL)
ipPointsDes	ITcVnContainer [▶ 370]	Container with destination points (same number as ipPoints1, ContainerType_Vector_TcVnPoint2_REAL)
ipInlierMask	ITcVnContainer [▶ 370]	A mask marking the inliers (optional, set to 0 if not available; ContainerType_Vector_SINT or ContainerType_Vector_USINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aHomography	TcVnMatrix3x3_LREAL [▶ 162]	Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 162]	Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 162]	Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 162]	Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 162]	Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 162]	Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 162]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eFlipAxis	ETcVnFlipAxis [▶ 203]	Selects the axis around which to flip the image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

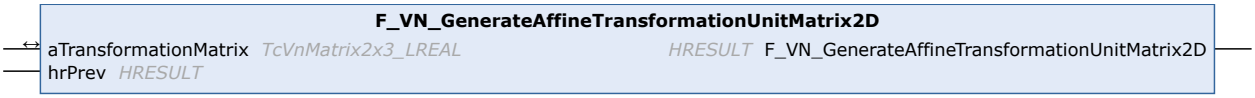
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Resulting affine transformation unit matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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 [▶ 162]	Source points
aDestPoints	TcVnArray3_Point2_REAL [▶ 162]	Destination points
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Resulting transformation matrix

Return value

HRESULT [▶ 141]

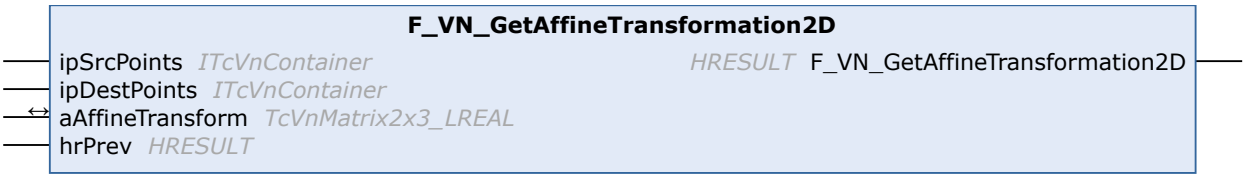
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 370]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 162]	Returns the affine transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 370]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 200]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 162]	Returns the affine transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 141]

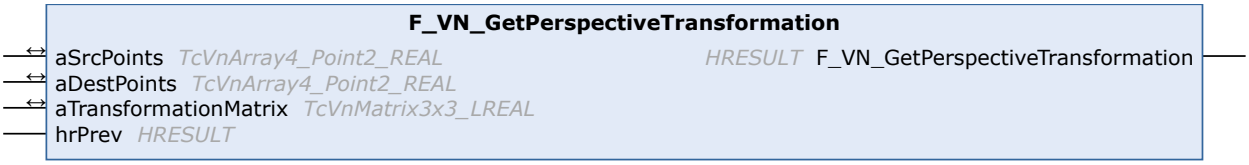
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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 [▶ 162]	Source points
aDestPoints	TcVnArray4_Point2_REAL [▶ 162]	Destination points
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Resulting transformation matrix

Return value

HRESULT [▶ 141]

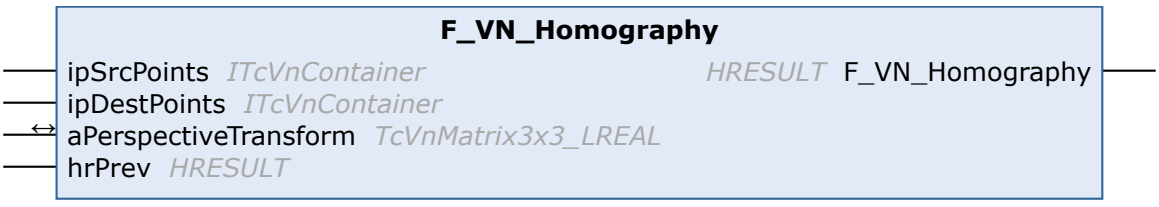
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 370]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	Returns the perspective transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 141]

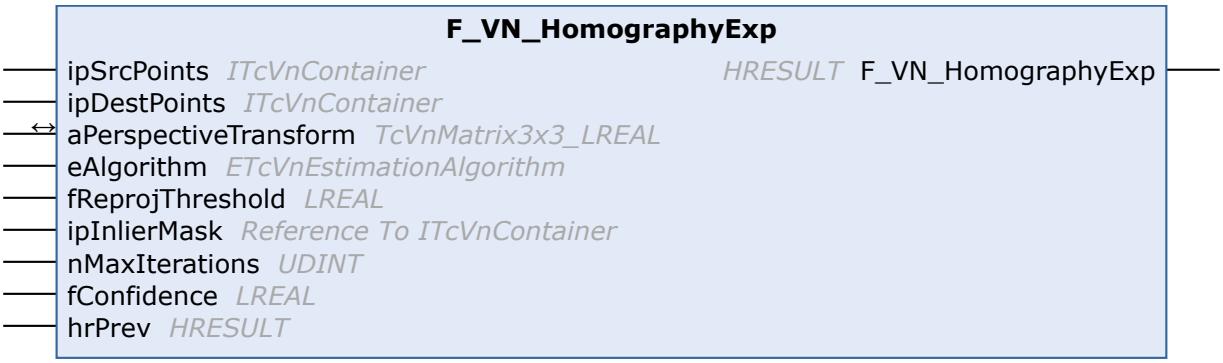
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 370]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 200]	Estimation algorithm
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only for RANSAC, RHO)
ipInlierMask	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	Returns the perspective transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 141]

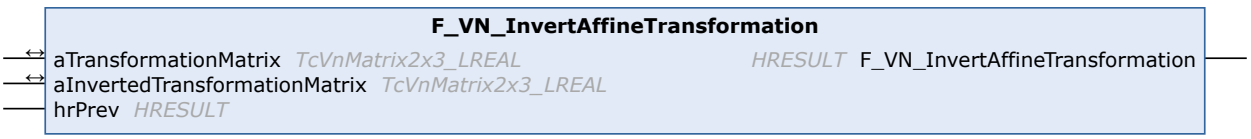
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Source affine transformation matrix
aInvertedTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Resulting inverted affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with 3D source points (TcVnPoint3_REAL or TcVnPoint3_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns the transformed points (same type as ipSrcPoints)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Translation vector

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	Interpolation type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

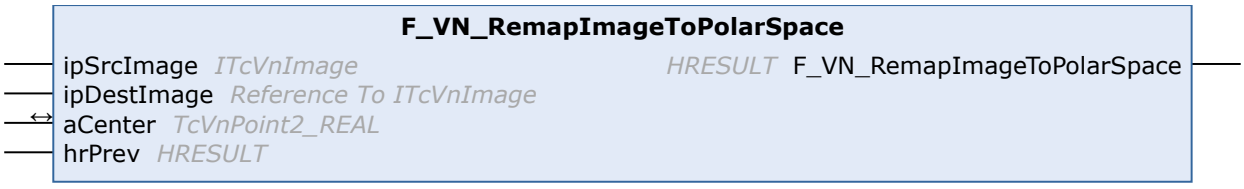
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

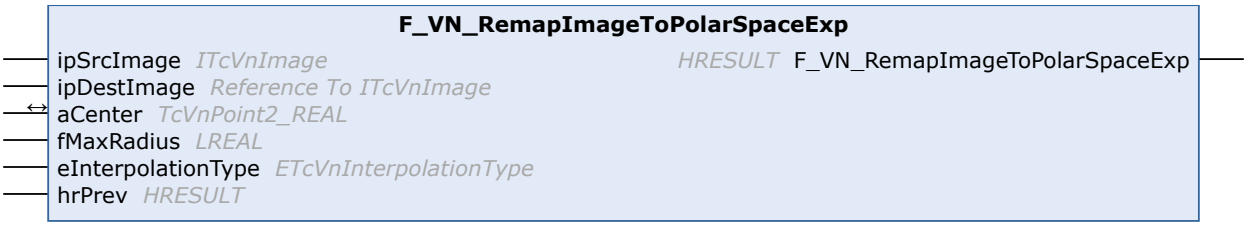
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	Interpolation type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

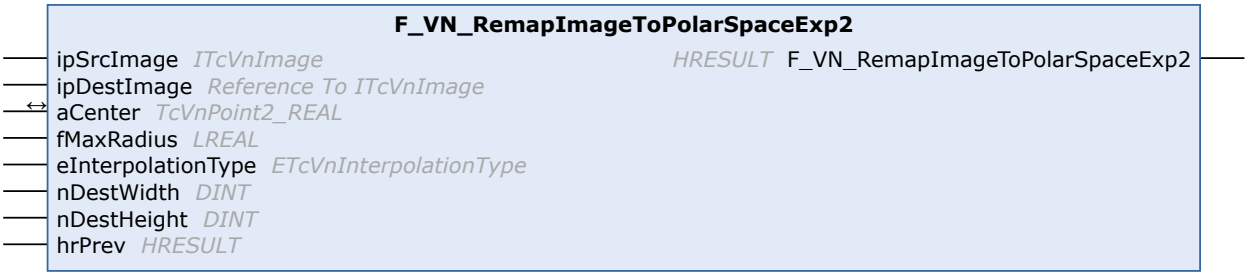
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 160]	Center point for the transformation

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
nWidth	UDINT	New width
nHeight	UDINT	New height
eInterpolationType	ETcVnInterpolationType [▶ 206]	Interpolation type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

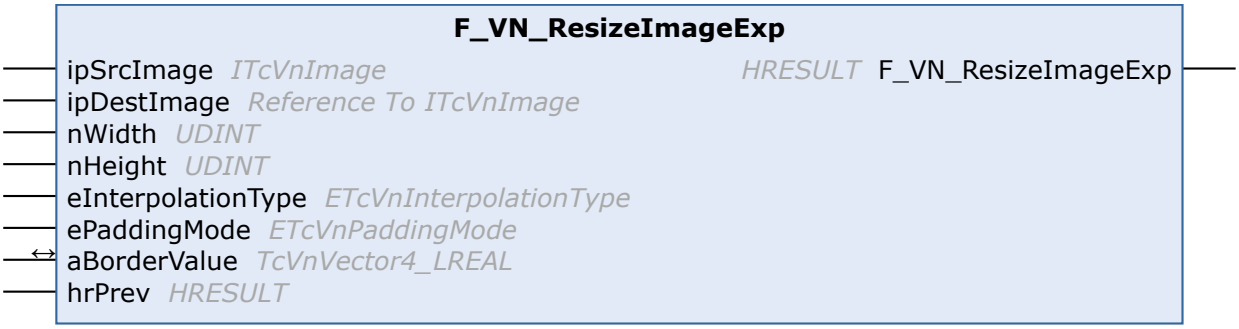
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
nWidth	UDINT	New width
nHeight	UDINT	New height
eInterpolationType	ETcVnInterpolationType [▶ 206]	Interpolation type
ePaddingMode	ETcVnPaddingMode [▶ 212]	Image padding mode
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aBorderValue	TcVnVector4_LREAL [▶ 162]	Border value, if TCVN_PM_LETTERBOX is used

Return value

HRESULT [▶ 141]

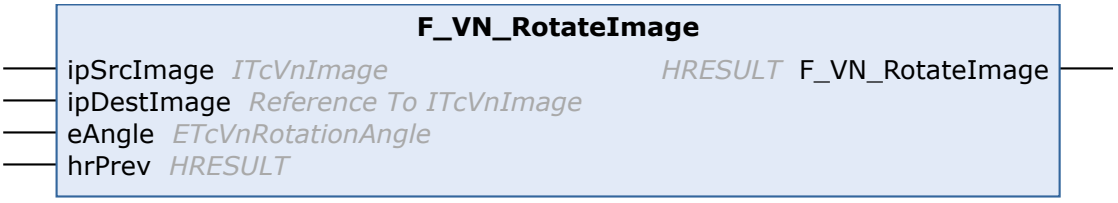
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
eAngle	ETcVnRotationAngle [▶ 215]	Rotation angle
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

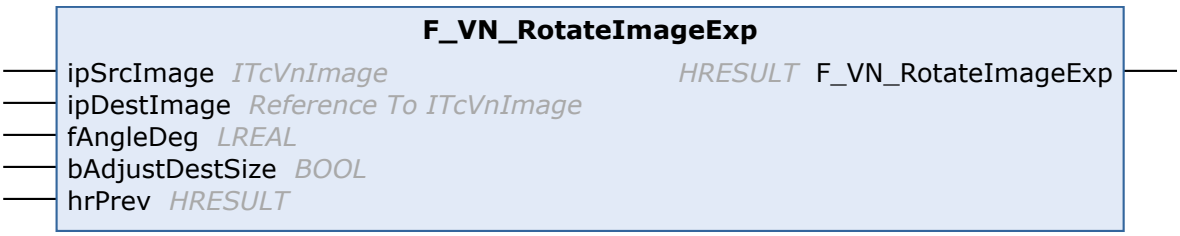
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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.

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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

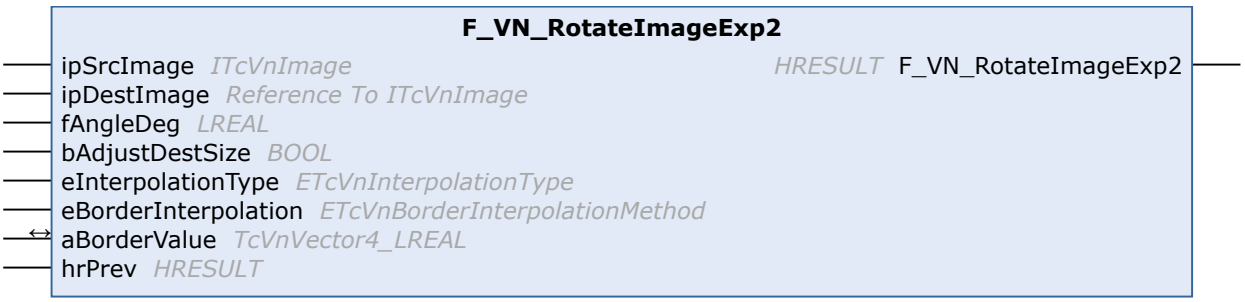
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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.

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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 166]	Border interpolation method (ISOLATED not supported)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aBorderValue	TcVnVector4_LREAL [▶ 162]	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix

Return value

HRESULT [▶ 141]

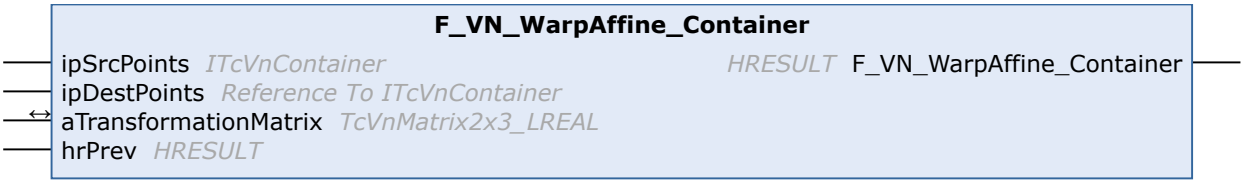
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix

Return value

HRESULT [▶ 141]

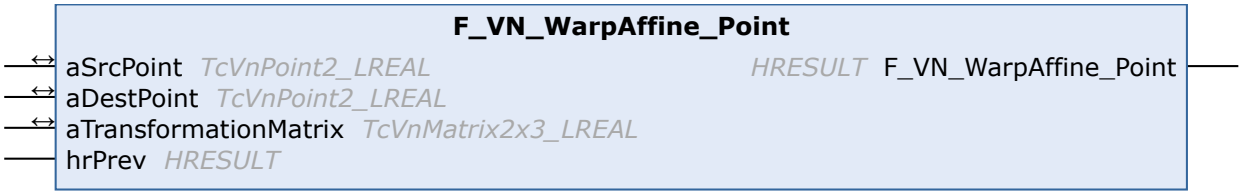
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 160]	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 160]	Destination point
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix

Return value

HRESULT [▶ 141]

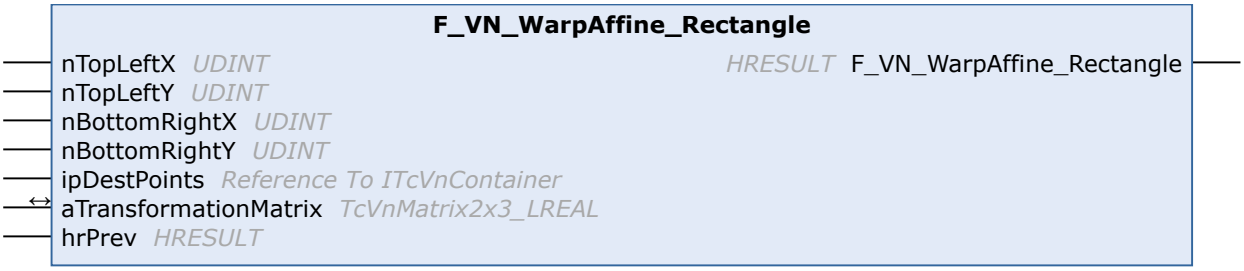
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix

Return value

HRESULT [▶ 141]

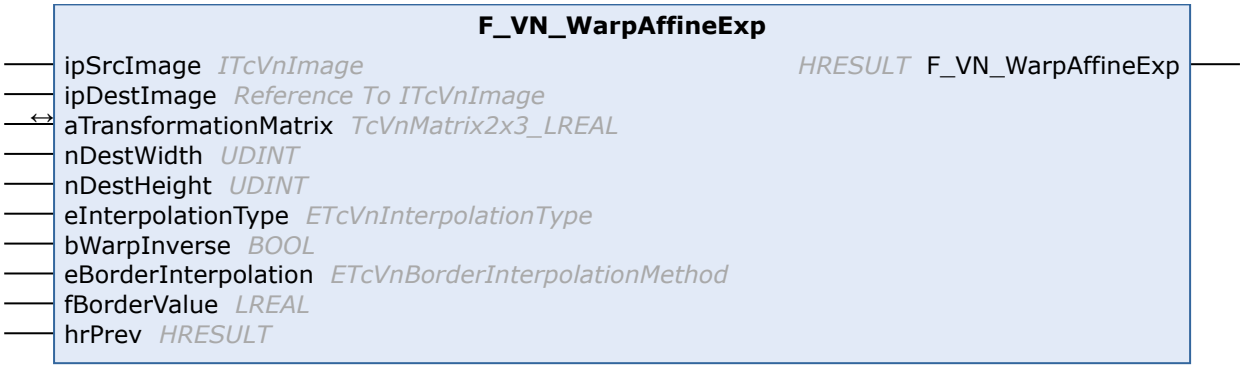
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	Interpolation method
bWarpInverse	BOOL	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 166]	Border interpolation method (ISOLATED not supported)
fBorderValue	LREAL	Border value, if CONSTANT is used
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 162]	Affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 370]	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 160]	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 160]	Destination point
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Perspective transformation matrix

Return value

HRESULT [▶ 141]

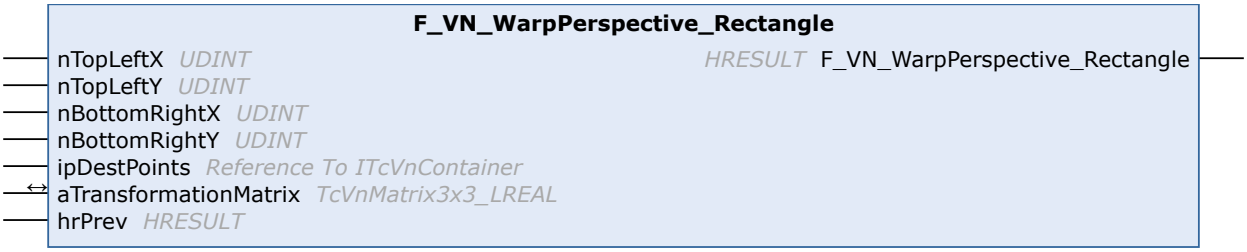
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Perspective transformation matrix

Return value

HRESULT [▶ 141]

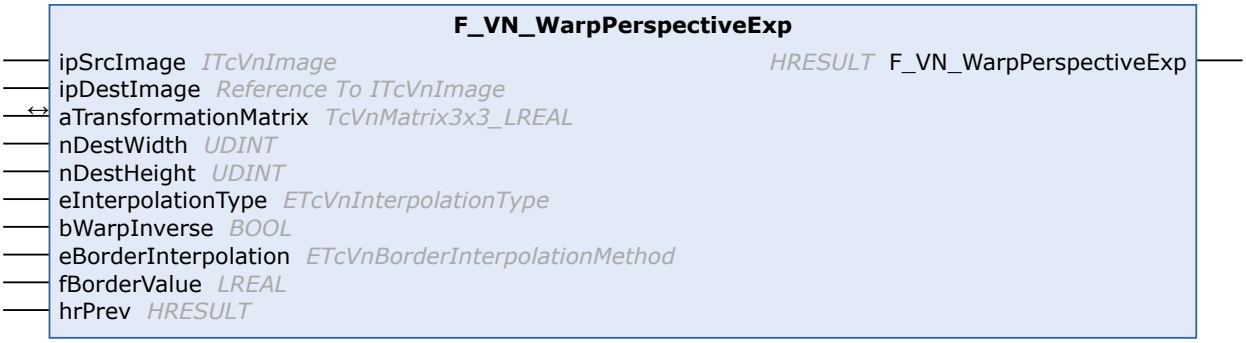
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 206]	Interpolation method
bWarpInverse	BOOL	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 166]	Border interpolation method (ISOLATED not supported)
fBorderValue	LREAL	Border value, if CONSTANT is used
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14 Image Analysis

This group contains functions for extracting information from images.

Functions

Object detection [▶ 1176]

- Contour search
- Object finding by Hough transformation
- Matching by image comparison

Statistical image features

- F_VN_ImageAverage(Exp) [▶ 1218]
- F_VN_ImageAverageStdDev(Exp) [▶ 1219]
- F_VN_ImageAbsoluteAverage [▶ 1233]
- F_VN_ImageMedian(Exp) [▶ 1224]
- F_VN_MaxPixelValue(Exp) [▶ 1227]
- F_VN_MinPixelValue(Exp) [▶ 1229]

Edge detection

- F_VN_CannyEdgeDetection(Exp) [▶ 1204]

Finding contiguous image regions

- F_VN_ConnectedComponents(Exp) [▶ 1207]
- F_VN_ConnectedComponentsWithStats (Exp) [▶ 1210]
- F_VN_GetConnectedComponent(Exp) [▶ 1216]

Alignment of a region

- F_VN_RegionOrientation(Exp) [▶ 1231]
- F_VN_UprightInnerRectangle(Exp) [▶ 1233]

Miscellaneous

- F_VN_CountNonZeroPixels [▶ 1212]
- F_VN_DistanceTransformation(Exp) [▶ 1213]
- F_VN_ImageCenterOfMass(Exp) [▶ 1221]
- F_VN_ImageMoments [▶ 1226]

Samples

Image analysis [▶ 2780]

6.1.4.14.1 Object Detection

This subgroup contains functions for object recognition.

Functions

Object detection

- F_VN_DetectBlobs(Exp) [▶ 1178]
- F_VN_FindContours(Exp) [▶ 1190]
- F_VN_FindContourHierarchyExp [▶ 1183]
- F_VN_HoughCircles(Exp) [▶ 1193]

- F_VN_HoughLines(Exp) [▶ 1195]
- F_VN_HoughLinesP(Exp) [▶ 1197]

Matching

- F_VN_MatchImageHuMoments [▶ 1198]
- F_VN_MatchTemplate(Exp) [▶ 1203]
- F_VN_MatchTemplateAndEvaluate(Exp) [▶ 1201]

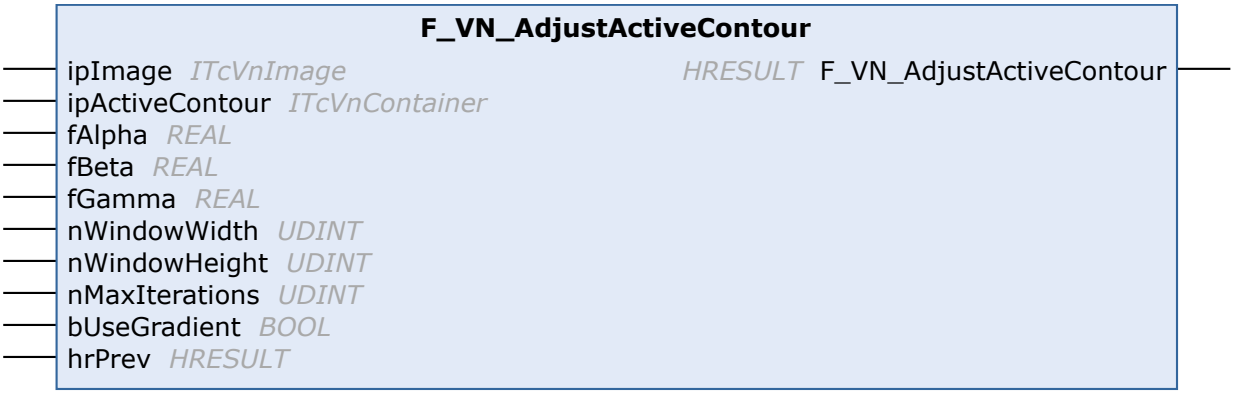
Miscellaneous

- F_VN_AdjustActiveContour [▶ 1177]

Samples

Object Detection [▶ 2780]

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 [▶ 411]	Source image (1 channel, USINT elements)
ipActiveContour	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

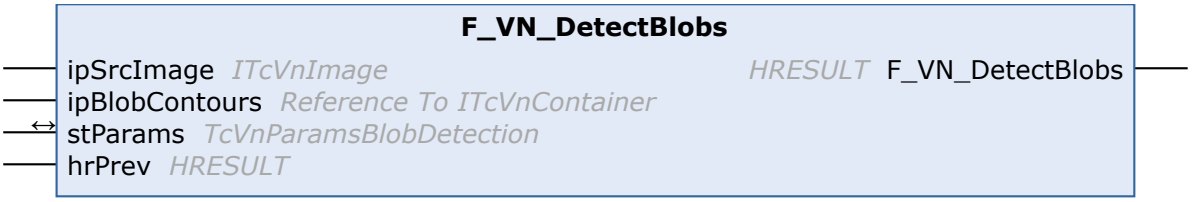
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBlobDetection [▶ 234]	Parameters to filter the detected contours.

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_DetectBlobs` detects blob contours on the basis of definable features in the input image. Internally, this takes place through contour finding analogous to F_VN_FindContours [▶ 1187] and subsequent filtering of the contours found on the basis of contour features.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image of the type `USINT` or `REAL`. 3-channel images are interpreted as RGB and converted internally into grayscale images.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipBlobContours`.

Parameters for blob detection

The parameters `stParams` of the structure TcVnParamsBlobDetection [▶ 234] define which features a contour in the input image must have in order to be found and returned.

Expert parameters

The expert version F_VN_DetectBlobsExp [▶ 1181] contains additional parameters.

Blobs parameters

Threshold value

In the first step, a contour search is carried out internally using one or more binary images. For this purpose, the input image is converted into binary images using one or more threshold values and the threshold value type.

Setting a single threshold value corresponds to calling F_VN_Threshold [▶ 1334]. To do this, *fThresholdStep* must be set to 0. Only the value of *fMinThreshold* is used.

Alternatively, several threshold values can be taken into account. The first threshold value is *fMinThreshold*. Starting from this value, *fThresholdStep* is added for the next threshold value until the threshold value is greater than *fMaxThreshold*. If, for example, the three threshold values 100, 150 and 200 are to be taken into account, the parameterization is as follows:

```
stBlobParams.fMinThreshold     := 100;
stBlobParams.fMaxThreshold     := 200;
stBlobParams.fThresholdStep    := 50;
```

Same contours with different threshold values

It may happen that the same contour is found at different threshold values. Identical contours are identified by means of a center point comparison. In this case *fMinBlobDistance* specifies the minimum center distance of different contours. If the distance is not reached, the contour is selected according to *eBlobCombination* (ETcVnBlobCombination [▶ 165]).

In addition, multiple finding of a contour at different threshold values can be a filter criterion. How often a contour has to be found in order to be considered is specified via *nMinRepeatability*.

Filter

In the second step, the contours found are filtered according to the defined parameters. The individual filter criteria can be activated via a Boolean variable.

Area in pixels

- Parameter: bFilterByArea, fMinArea, fMaxArea
- Alternatively, the area of a contour can be determined via F_VN_ContourArea [▶ 958].

Circularity

- Parameter: bFilterByCircularity, fMinCicularity, fMaxCircularity
- Value range from 0 to 1; a perfect circle has a circularity of 1.
- Alternatively, the circularity of a contour can be determined via F_VN_ContourCircularity [▶ 960].

Convexity

- Parameter: bFilterByConvexity, fMinConvexity, fMaxConvexity
- The value range is from 0 to 1. A perfectly convex shape has a convexity of 1.

Eccentricity

- Parameter: bFilterByEccentricity, fMinEccentricity, fMaxEccentricity
- Value range from 0 to 1; a perfect circle has an eccentricity of 0.
- Alternatively, the eccentricity of a contour can be determined via F_VN_ContourEccentricity [▶ 962].

Inertia ratio

- Parameter: bFilterByInertiaRatio, fMinInertiaRatio, fMaxInertiaRatio
- Value range from 0 to 1; shapes with the same height and width have an inertia ratio of 1.

Application

```
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
);
```

Sample

- Blob Detection with watchdog monitoring [▶ 2780]

Related functions

- F_VN_FindContours(Exp) [▶ 1187] for general contour finding
- F_VN_FindContourHierarchyExp [▶ 1183] with return of the hierarchy
- F_VN_DetectBlobs [▶ 1178] with integrated filtering of the contours

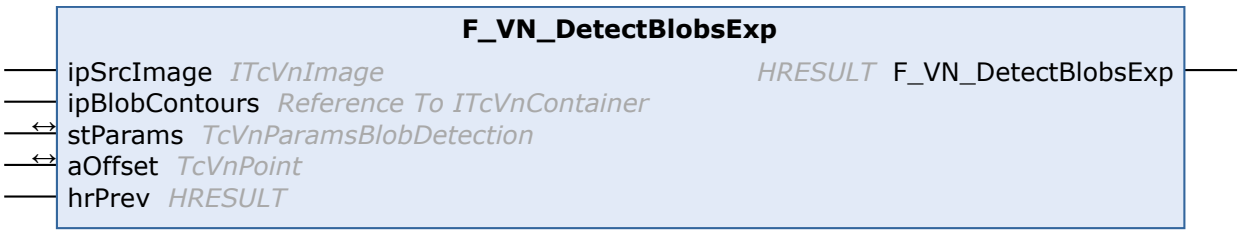
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBlobDetection [▶ 234]	Parameters to filter the detected contours.
aOffset	TcVnPoint [▶ 160]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_DetectBlobsExp` is the expert version of F_VN_DetectBlobs [▶ 1178]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel or 3-channel image of the type `USINT` or `REAL`. 3-channel images are interpreted as RGB and converted internally into grayscale images.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipBlobContours`.

Parameters for blob detection

The parameters `stParams` of the structure TcVnParamsBlobDetection [▶ 234] define which features a contour in the input image must have in order to be found and returned.

Offset (Expert)

The offset `aOffset` defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Related functions

- F_VN_FindContours(Exp) [▶ 1187] for general contour finding
- F_VN_FindContourHierarchyExp [▶ 1183] with return of the hierarchy
- F_VN_DetectBlobs [▶ 1178] with integrated filtering of the contours

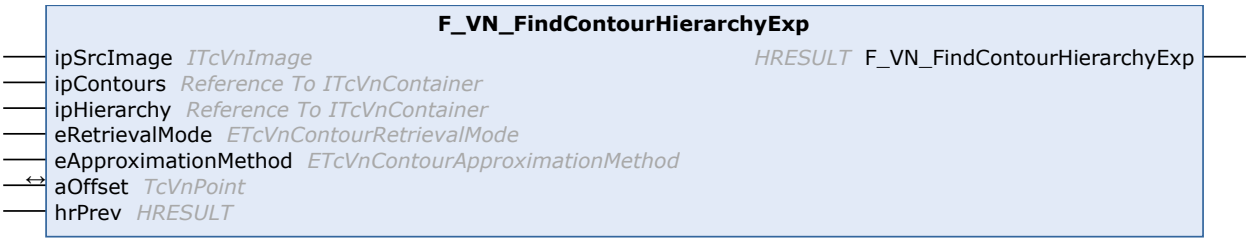
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 191]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 188]	Contour encoding
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aOffset	TcVnPoint [▶ 160]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FindContourHierarchyExp` is an extension of the function F_VN_FindContoursExp [▶ 1190] by the return of the contour hierarchy.

Parameter

Input image

The input image `ipSrcImage` must have precisely 1 channel and should be a binary image.

If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipContours`. The container is a two-dimensional array of points and therefore has the type `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

The contours that are found are returned in `ipContours`. Each contour consists of a sum of points which are combined into a container [▶ 152]. Another higher-level container then consolidates all contours.

Hierarchy (Expert, return value)

The parameter `ipHierarchy` returns the hierarchy of the contours found in a container [▶ 152] of the type TcVnVector4_DINT. The vectors of the container have the following elements:

`[0]` – NEXT - Index of the next contour on the same level
`[1]` – PREVIOUS - Index of the previous contour on the same level
`[2]` – CHILD - Index of the first child contour
`[3]` – PARENT - Index of parent contour

Search mode (Expert)

The search mode `eRetrievalMode` of the type ETcVnContourRetrievalMode [▶ 191] defines which contours are to be sought:

- `TCVN_CRM_LIST`: All contours
- `TCVN_CRM_EXTERNAL`: Only outer contours (as with F_VN_FindContours [▶ 1187])
- `TCVN_CRM_CONNECTED_COMPONENTS`: Contours with a two-layer hierarchy
- `TCVN_CRM_TREE`: All contours; used in interaction with the function F_VN_FindContourHierarchyExp [▶ 1183] in order to determine the hierarchy of the contours
- `TCVN_CRM_FLOODFILL`: Returns the contours found with the help of a floodfill algorithm (available only for DINT images)

Approximation method (Expert)

The approximation method `eApproximationMethod` of the type ETcVnContourApproximationMethod [▶ 188] specifies whether the contours are to be described on the basis of all surrounding points (TCVN_CAM_NONE) or whether the number of describing points are to be reduced with the help of approximation.

Offset (Expert)

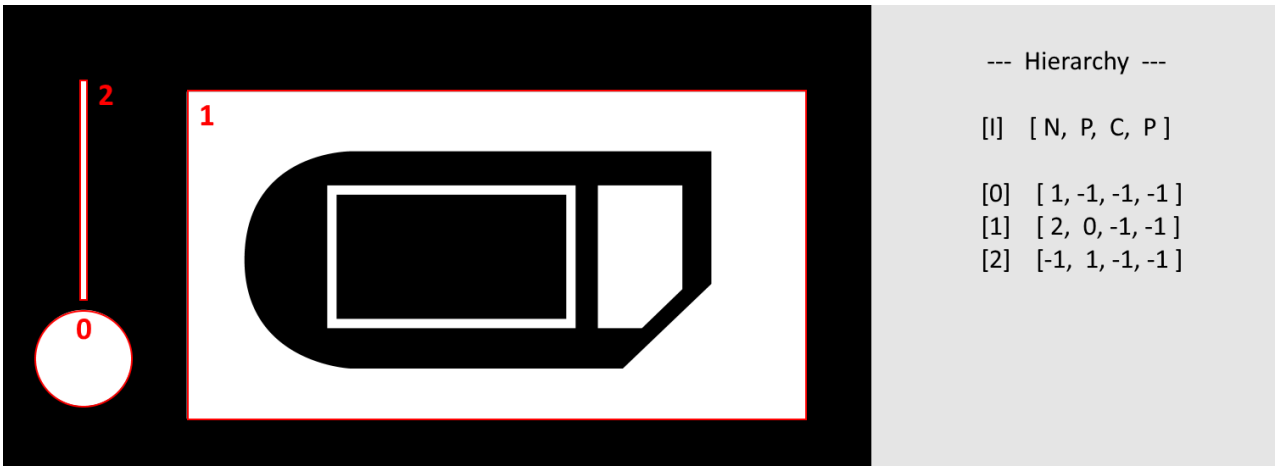
The offset `aOffset` defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Examples of contour hierarchies

Via eRetrievalMode, not only is it possible to specify whether all contours or just the outer contours are to be found, but also how contours are to be sorted and how they are hierarchically related to one another. This can be read from the parameter `ipHierarchy`:

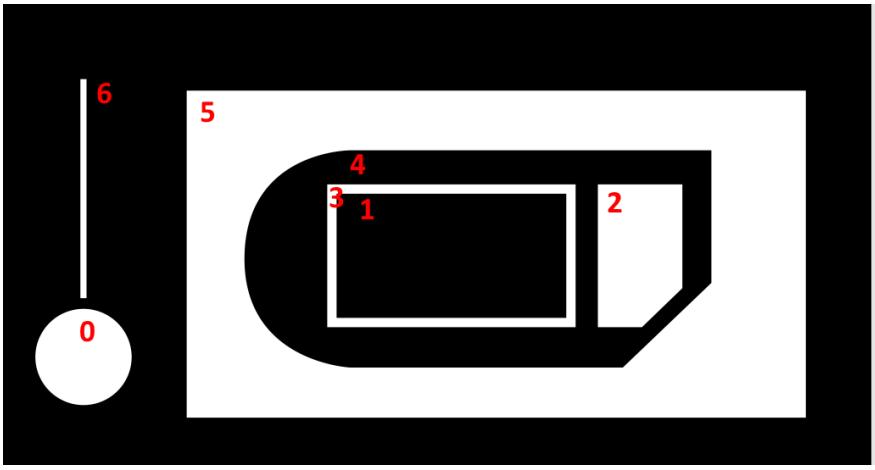
- **TCVN_CRM_EXTERNAL**

  Only the outer contours are returned as a sorted list. The **C**hild `[2]` and **P**arent `[3]` relationships are not determined and are invalid with `-1`.



- **TCVN_CRM_LIST**

  All contours are returned as a sorted list. The **C**hild `[2]` and **P**arent `[3]` relationships are not determined and are invalid with `-1`.



--- Hierarchy ---

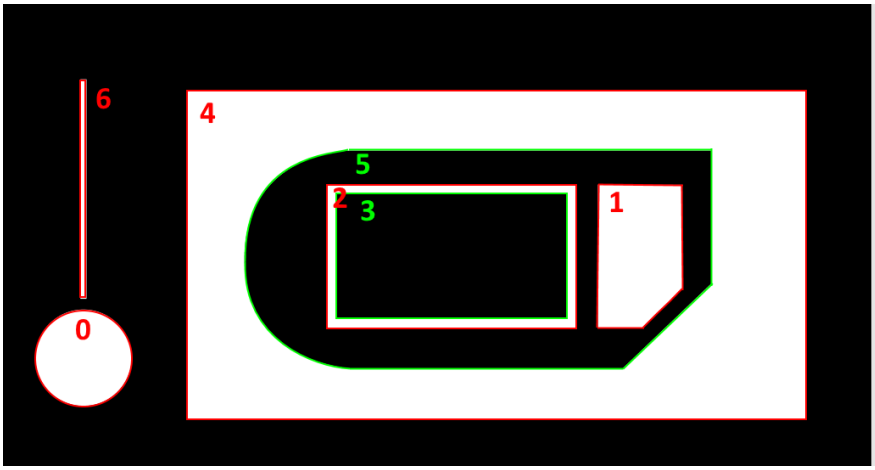
[I] [N, P, C, P]

[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**

  All contours are sorted according to a 2-step hierarchy. The outer contours are considered first, then in the second step the contour lying directly within them. If there are further contours within the second step, they are set to the first step like outer contours

  In this sample, this means that the inner **C**hild `[2]` contours (green in the image) cannot be **P**arent `[3]` contours (red in the image) at the same time. Therefore, contours that would lie in a tree structure at Level 3 are regarded again as outer contours and set to Level 1 (**P**arent `[3] = -1)`, in this case contours 1 and 2.



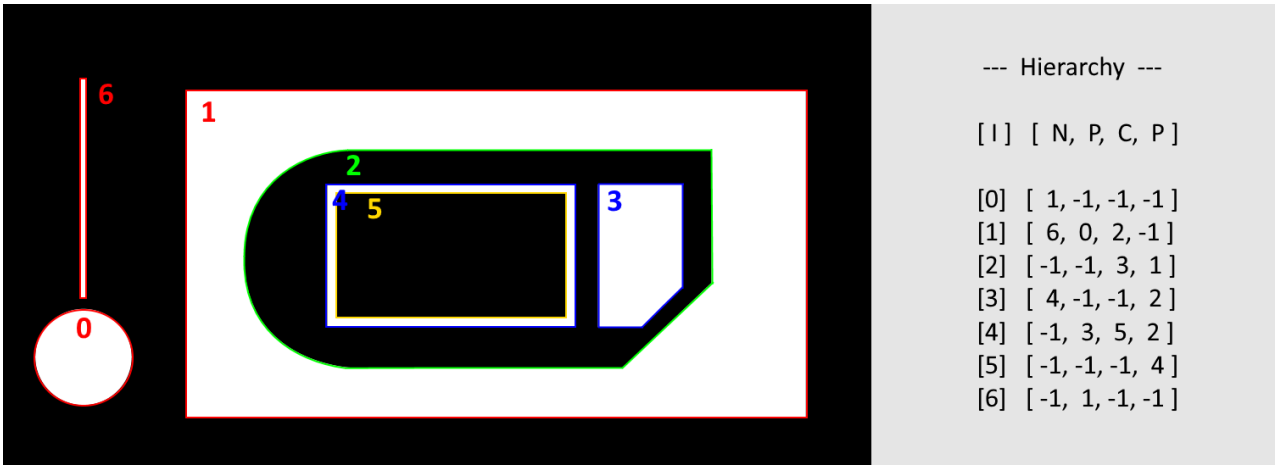
--- Hierarchy ---

[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**

  All contours are set according to a tree structure. It is possible for contours to be both **C**hild `[2]` and **P**arent `[3]` contours, depending on how they are nested within one another.



Samples

- Find Contour - Hierarchy & Retrieval Mode [▶ 2791], for the parameter ETcVnContourRetrievalMode [▶ 191]
- Find Contour - Approximation Method [▶ 2788], for parameter ETcVnContourApproximationMethod [▶ 188]
- Find contour instead of Blob Detection [▶ 2786], for comparison with function F_VN_DetectBlobs [▶ 1178]

Related functions

- F_VN_FindContours(Exp) [▶ 1187] for general contour finding
- F_VN_FindContourHierarchyExp [▶ 1183] with return of the hierarchy
- F_VN_DetectBlobs [▶ 1178] with integrated filtering of the contours

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FindContours` detects outer contours in the input image.

Parameter

Input image

The input image `ipSrcImage` must have precisely 1 channel and should be a binary image.

If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipContours`. The container is a two-dimensional array of points and therefore has the type `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

The contours that are found are returned in `ipContours`. Each contour consists of a sum of points which are combined into a container [▶ 152]. Another higher-level container then consolidates all contours.

Expert parameters

The expert version F_VN_FindContoursExp [▶ 1190] contains additional parameters. This enables inner contours to be found too, for example.

Application

The finding and processing of contours looks like this, for example:

Pre-processing

As a binary image is to be used for the input image `ipSrcImage`, a threshold value or binary segmentation should be applied to the image prior to the function call, e.g. using the function F_VN_Threshold [▶ 1334]:

```
hr := F_VN_Threshold(ipImageIn, ipImageWork, 128, 255, TCVN_TT_BINARY, hr);
```

Function call

Nothing further needs to be observed with regard to the function call. Parameter settings are only necessary with the expert version F_VN_FindContoursExp [▶ 1190].

```
hr := F_VN_FindContours(
    ipSrcImage  :=  ipImageIn,
    ipContours  :=  ipContours,
    hrPrev      :=  hr
);
```

Further processing

F_VN_GetForwardIterator [▶ 766] can be used to apply an iterator to the higher-level container, which makes it possible to retrieve the containers of the individual contours one after the other via F_VN_GetContainer [▶ 764] in order to analyze them with the Contour Analysis functions, for example. Remember to increment the iterator with F_VN_IncrementIterator [▶ 768].

```
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
```

Alternatively you can also retrieve a single contour from the container of all contours with F_VN_GetAt_ITcVnContainer [▶ 645].

The number of contours found can be queried with F_VN_GetNumberOfElements [▶ 766].

Visualization

Use the function F_VN_DrawContours(Exp) [▶ 1009] to draw the contours found.

```
hr := F_VN_DrawContours(
    ipContours      :=  ipContours,
    nContourIndex   :=  -1,
    ipDestImage     :=  ipImageRes,
    aColor          :=  aColorRed,
    nThickness      :=  5,
    hrPrev          :=  hr
);
```

All contours in the contour list `ipContours` are drawn by setting `nContoursIndex` to `-1`.

Samples

- Find contour instead of Blob Detection [▶ 2786]

Related functions

- F_VN_FindContours(Exp) [▶ 1187] for general contour finding
- F_VN_FindContourHierarchyExp [▶ 1183] with return of the hierarchy
- F_VN_DetectBlobs [▶ 1178] with integrated filtering of the contours

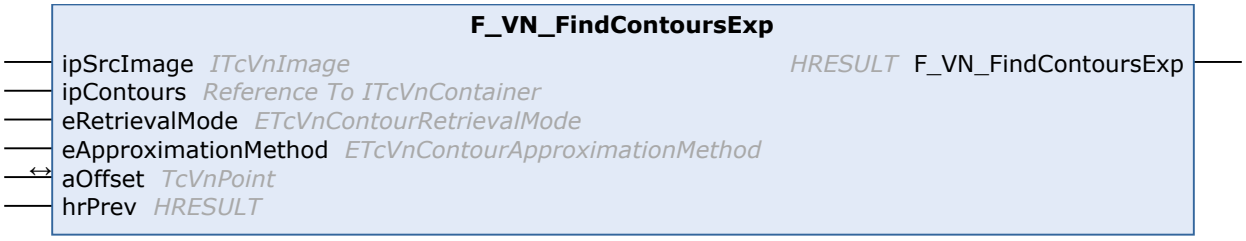
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 370]	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 [▶ 191]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 188]	Contour encoding
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aOffset	TcVnPoint [▶ 160]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FindContoursExp` is the expert version of F_VN_FindContours [▶ 1187]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must have precisely 1 channel and should be a binary image.

If the input is not a binary image, it will be converted internally to one by means of a threshold value of 1. Therefore, the input image should be manually pre-processed with a threshold value or a binary segmentation.

List of contours found (Return value)

All contours found in the input image are returned in the container `ipContours`. The container is a two-dimensional array of points and therefore has the type `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

The contours that are found are returned in `ipContours`. Each contour consists of a sum of points which are combined into a container [▶ 152]. Another higher-level container then consolidates all contours.

Search mode (Expert)

The search mode `eRetrievalMode` of the type ETcVnContourRetrievalMode [▶ 191] defines which contours are to be sought:

- `TCVN_CRM_LIST`: All contours
- `TCVN_CRM_EXTERNAL`: Only outer contours (as with F_VN_FindContours [▶ 1187])
- `TCVN_CRM_CONNECTED_COMPONENTS`: Contours with a two-layer hierarchy
- `TCVN_CRM_TREE`: All contours; used in interaction with the function F_VN_FindContourHierarchyExp [▶ 1183] in order to determine the hierarchy of the contours
- `TCVN_CRM_FLOODFILL`: Returns the contours found with the help of a floodfill algorithm (available only for DINT images)

Approximation method (Expert)

The approximation method `eApproximationMethod` of the type ETcVnContourApproximationMethod [▶ 188] specifies whether the contours are to be described on the basis of all surrounding points (TCVN_CAM_NONE) or whether the number of describing points are to be reduced with the help of approximation.

Offset (Expert)

The offset `aOffset` defines constant X/Y values by which all contour points are shifted. This can be helpful when contours are sought in an ROI and their positions should be referenced to the original image.

Application

The finding of all contours in the input image, where the contour points are not approximated, but shifted by 100 px in the x direction and by -50 px in the y direction, looks like this, for example:

```
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
);
```

Samples

- Find Contour - Hierarchy & Retrieval Mode [▶ 2791], for the parameter ETcVnContourRetrievalMode [▶ 191]
- Find Contour - Approximation Method [▶ 2788], for parameter ETcVnContourApproximationMethod [▶ 188]
- Find contour instead of Blob Detection [▶ 2786], for comparison with function F_VN_DetectBlobs [▶ 1178]

Related functions

- F_VN_FindContours(Exp) [▶ 1187] for general contour finding
- F_VN_FindContourHierarchyExp [▶ 1183] with return of the hierarchy
- F_VN_DetectBlobs [▶ 1178] with integrated filtering of the contours

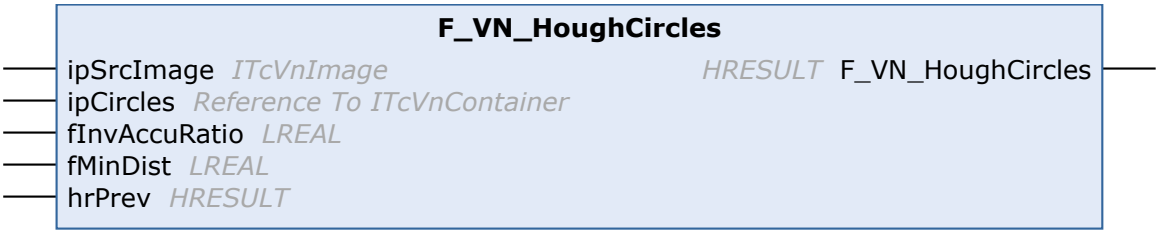
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, gray-level)
ipCircles	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

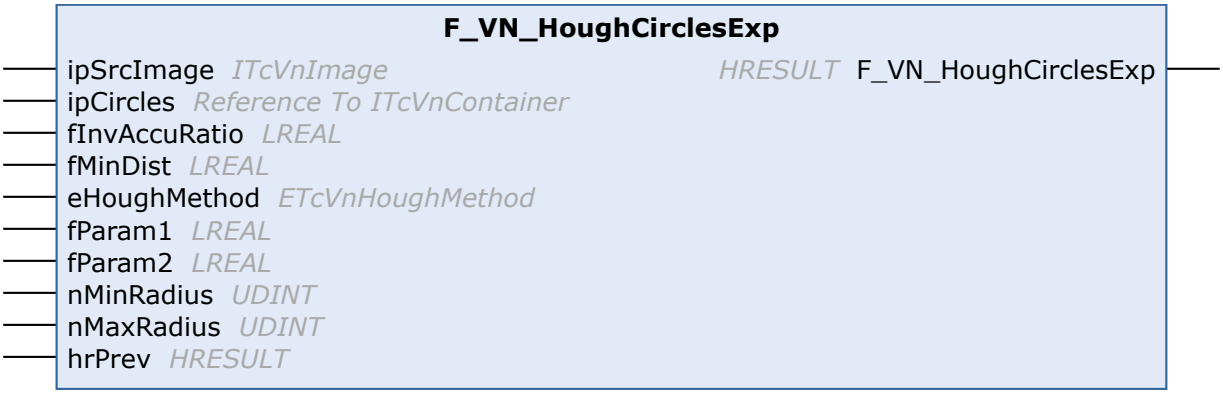
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, gray-level)
ipCircles	Reference To ITcVnContainer [▶ 370]	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 [▶ 205]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

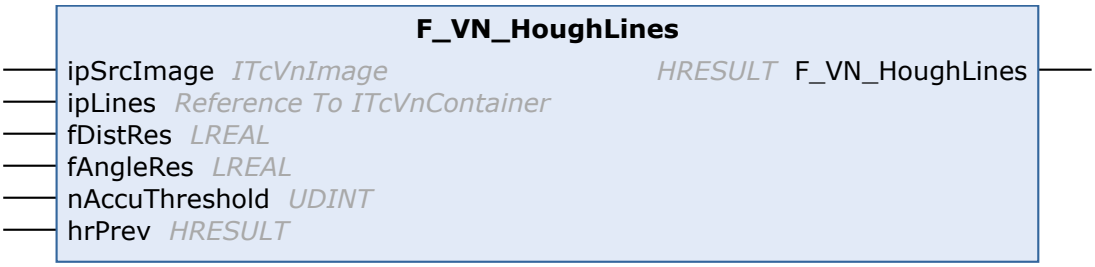
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

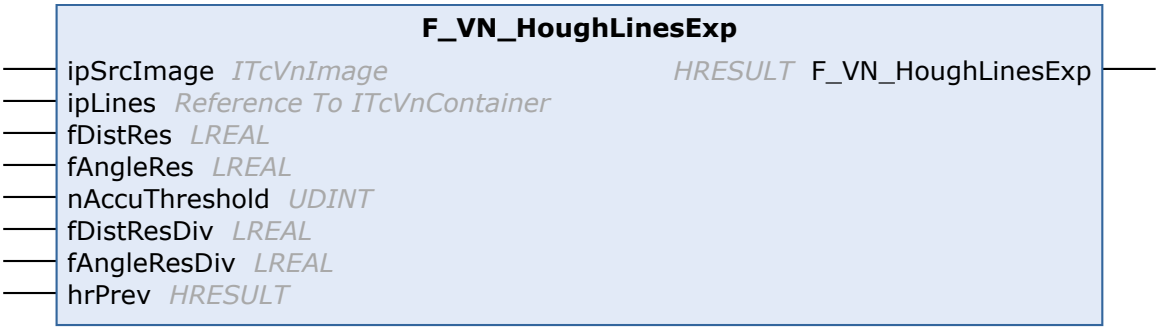
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

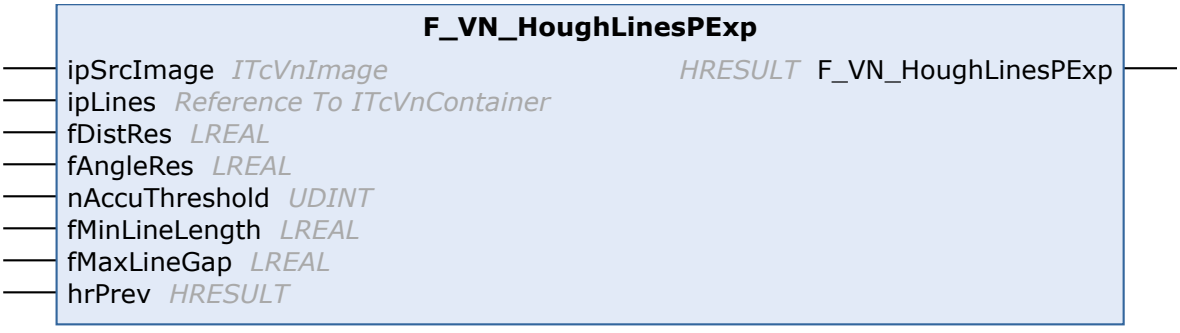
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

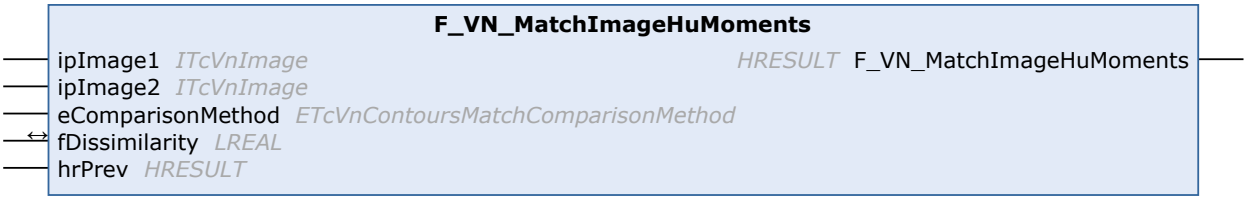
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	First image (1 channel)
ipImage2	ITcVnImage [▶ 411]	Second image (1 channel)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 193]	Method used for comparing the Hu moment invariants of the images
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

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 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 [▶ 411]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 411]	Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

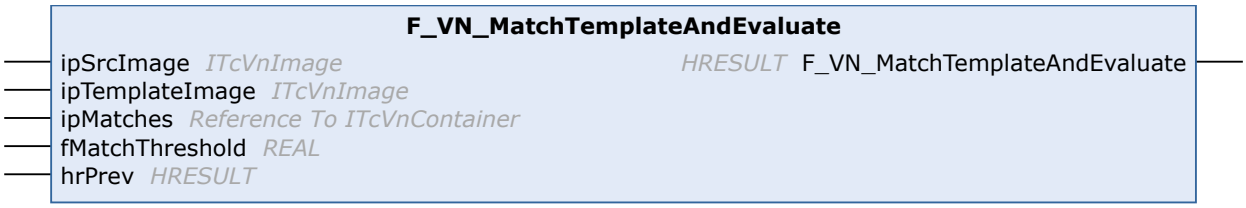
Required License

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 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 [▶ 411]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 411]	Template image (same type as ipSrcImage, smaller width and height)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

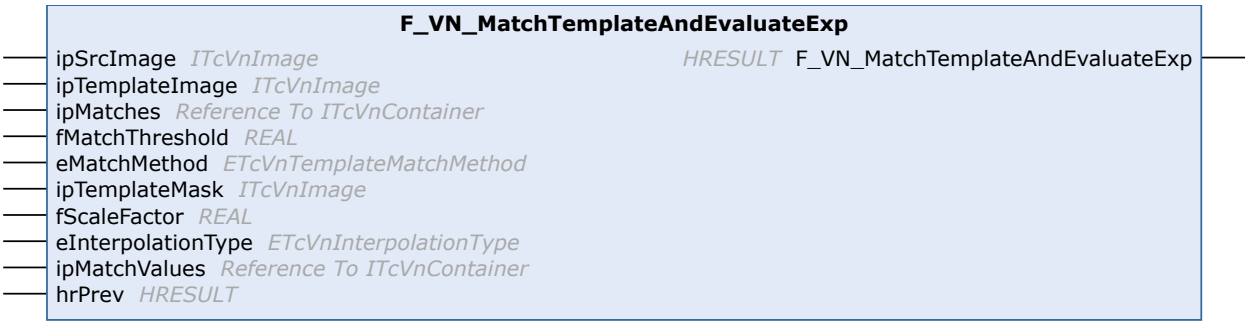
Required License

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 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 [▶ 411]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 411]	Template image (same type as ipSrcImage, smaller width and height)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 219]	Specifies the template match method
ipTemplateMask	ITcVnImage [▶ 411]	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 [▶ 206]	Image resize interpolation type (only used if fScaleFactor != 1, TCVN_IT_BILINEAR recommended for most cases)
ipMatchValues	Reference To ITcVnContainer [▶ 370]	Optionally returns the matching values (ContainerType_Vector_REAL, same size and sort order as ipMatches. Set to 0 if not required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

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 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 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 [▶ 411]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 411]	Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	Reference To ITcVnImage [▶ 411]	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 [▶ 219]	Specifies the template match method
ipTemplateMask	ITcVnImage [▶ 411]	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

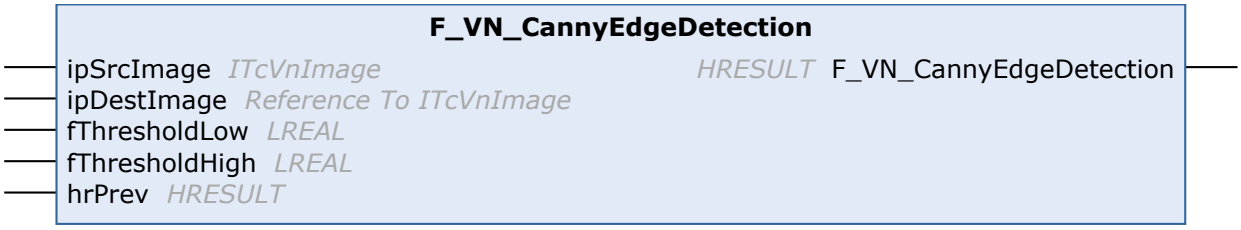
Required License

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 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 [▶ 411]	Source image (elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function finds edges in an image and illustrates them by white lines on a black background. In addition, this expert version F_VN_CannyEdgeDetectionExp [▶ 1206] exists with advanced parameterization options.

Algorithm

The Canny Edge edge detection is a multi-stage algorithm for the most robust edge detection possible. It consists of the following steps:

1. Reduction of noise by means of Gaussian filter.
2. Edge finding by means of Sobel operator.
3. Line thinning by means of non-maximum suppression, i.e. the resulting lines are 1 px thick.

4. Hysteresis, so that only thick lines remain (see Threshold parameters below).

Parameter

Original image

A 1-channel 8-bit intensity image (pixel format `TCVN_ET_USINT`) is expected for the original image `ipSrcImage`. You can therefore use a grayscale image or the individual channels of a color image.

Result image

The result image `ipDestImage` has the same format.

Threshold values

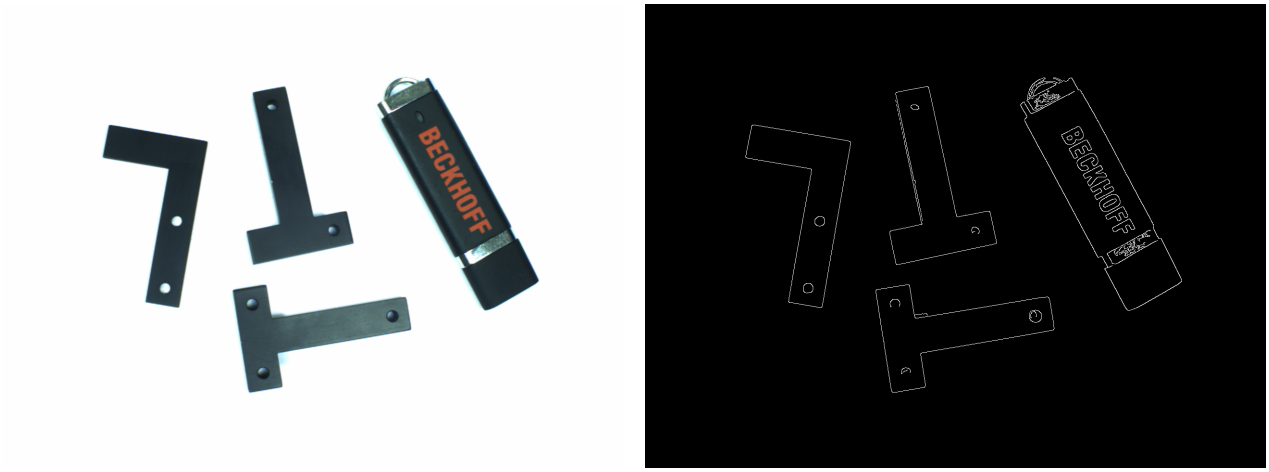
The two thresholds `fThresholdLow` and `fThresholdHigh` determine which of the edges found are accepted: Pixels with a gradient higher than `fThresholdHigh` are accepted. Pixels with a color gradient below `fThresholdLow` are discarded. Pixels with gradients in between are only accepted if they are linked to already accepted pixels.

The two thresholds make a flexible setting possible. A hard threshold is also realizable by setting both values the same.

Important edge information may be lost if you set the thresholds too high. If the thresholds are too low, noise and other irrelevant information might be interpreted as edges.

Application

```
hr := F_VN_CannyEdgeDetection(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageWork,
    fThresholdLow   :=  0,
    fThresholdHigh  :=  200,
    hr
);
```



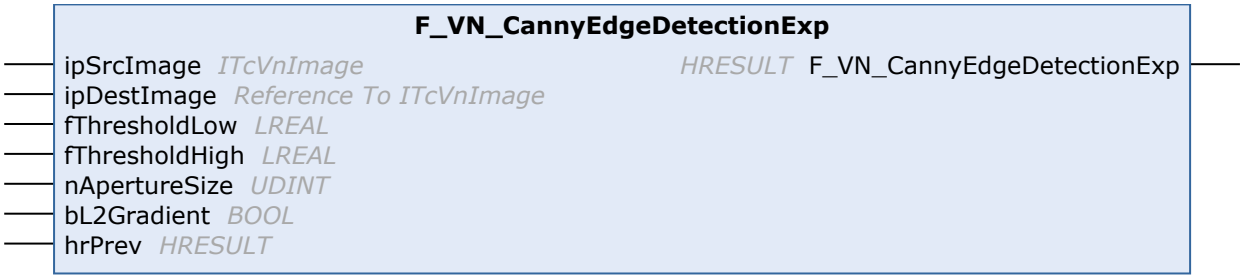
Required License

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 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 [▶ 411]	Source image (elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function is the expert version of F_VN_CannyEdgeDetection [▶ 1204]. It contains advanced parameterization options.

Parameter

The following parameters are additionally available in this expert version:

Aperture size

Only odd values of `3` or more are valid for `nApertureSize`.

L1/L2 standard

Two standards are available for the calculation of the gradients. The L1 standard is faster. The L2 standard, conversely, is more accurate.

$$L_1 = \left|\frac{dI}{dx}\right| + \left|\frac{dI}{dy}\right| \qquad L_2 = \sqrt{\left(\frac{dI}{dx}\right)^2 + \left(\frac{dI}{dy}\right)^2}$$

Set the parameter `bL2Gradient` to `FALSE` to use the L1 standard and to `TRUE` to use the L2 standard.

Application

An application of this function can look like this, for example:

```
hr := F_VN_CannyEdgeDetectionExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageWork,
    fThresholdLow   :=  0,
    fThresholdHigh  :=  200,
    nApertureSize   :=  5,
    bL2Gradient     :=  TRUE,
    hr
);
```

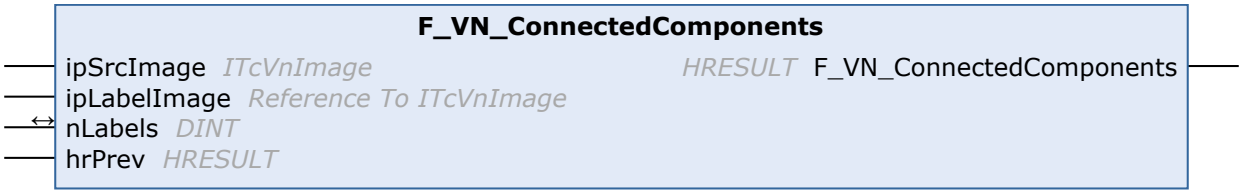
Required License

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 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 [▶ 411]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 411]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

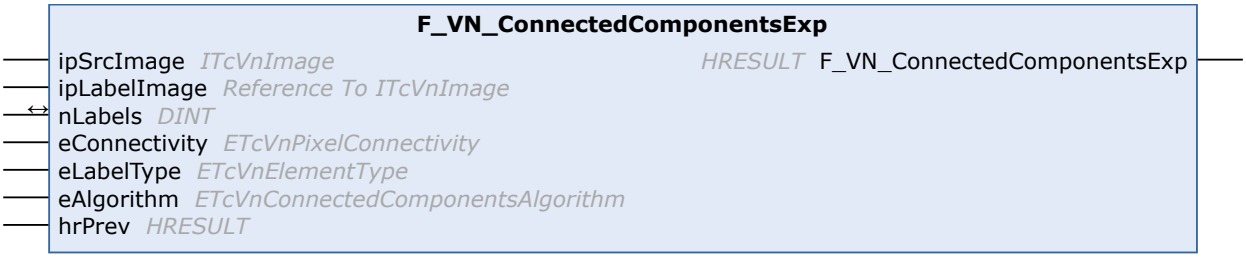
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 411]	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 [▶ 212]	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 199]	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 187]	Selects the applied algorithm
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

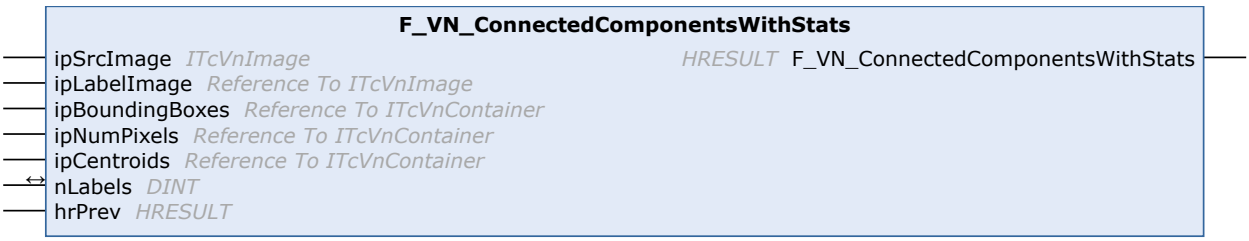
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 411]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 370]	Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	Reference To ITcVnContainer [▶ 370]	Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	Reference To ITcVnContainer [▶ 370]	Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

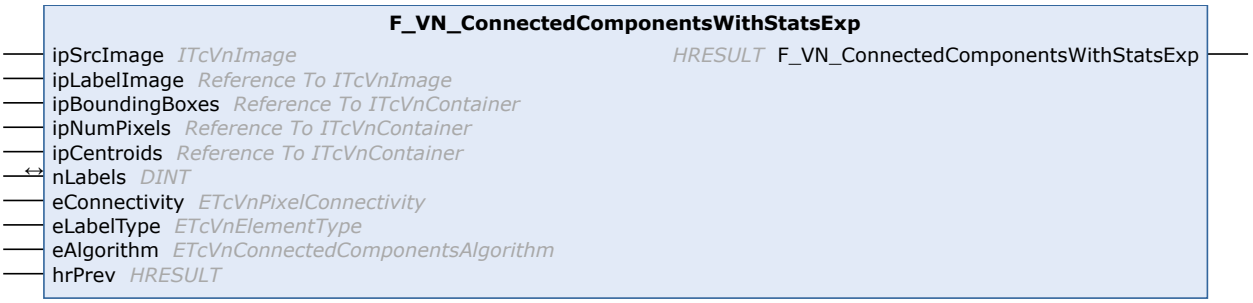
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 411]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 370]	Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	Reference To ITcVnContainer [▶ 370]	Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	Reference To ITcVnContainer [▶ 370]	Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)
eConnectivity	ETcVnPixelConnectivity [▶ 212]	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 199]	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 187]	Selects the applied algorithm
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

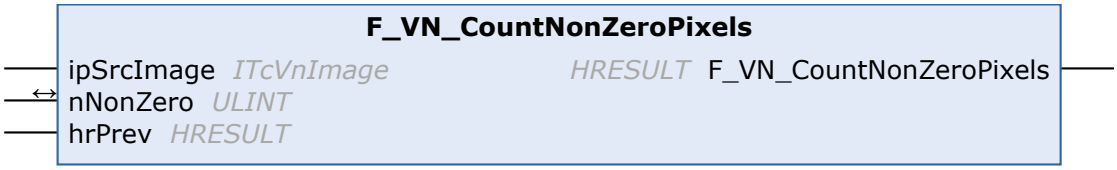
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

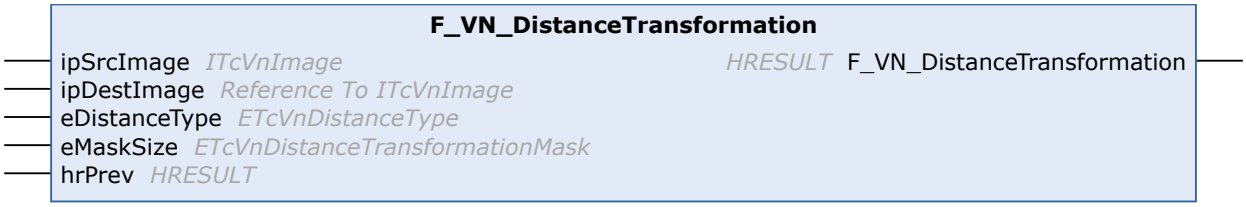
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 196]	Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 195]	Size of the distance transformation mask
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function performs the distance transformation of an image. In addition, there is an expert version F_VN_DistanceTransformationExp [▶ 1215] with advanced parameterization options.

Parameter

The following parameters are available in the basic version:

Original image

A 1-channel 8-bit image (element type `TCVN_ET_USINT`) is expected for the original image `ipSrcImage`. Other image formats must first be converted.

Result image

The result image `ipDestImage` is returned with the element type `TCVN_ET_REAL`. The value of each pixel describes the distance between the respective pixel and the nearest 0-value pixel in the original image.

Display in the ADS Image Watch

In order to be able to display the result image in the ADS Image Watch, it must first be converted into a compatible image format. Depending on the use case, it may be useful to scale the pixel values.

Distance type

The distance type `eDistanceType` determines the equation with which the distance between a pixel and the nearest 0-value pixel is calculated. The following distance types from the enum ETcVnDistanceType [▶ 196] are supported:

- TCVN_DT_L1
- TCVN_DT_L2
- TCVN_DT_C

Mask sizes

The mask size `eMaskSize` describes the approximation of the Euclidean distance if the L2 standard is selected as the distance type.

- TCVN_DTM_PRECISE: The Euclidean distance is calculated precisely.
- TCVN_DTM_3: The Euclidean distance is approximated as the sum of distance elements within a 3x3 mask (horizontal, vertical and diagonal).
- TCVN_DTM_5: The Euclidean distance is approximated as the sum of distance elements within a 5x5 mask (horizontal, vertical, diagonal and knight's move (as in chess)).

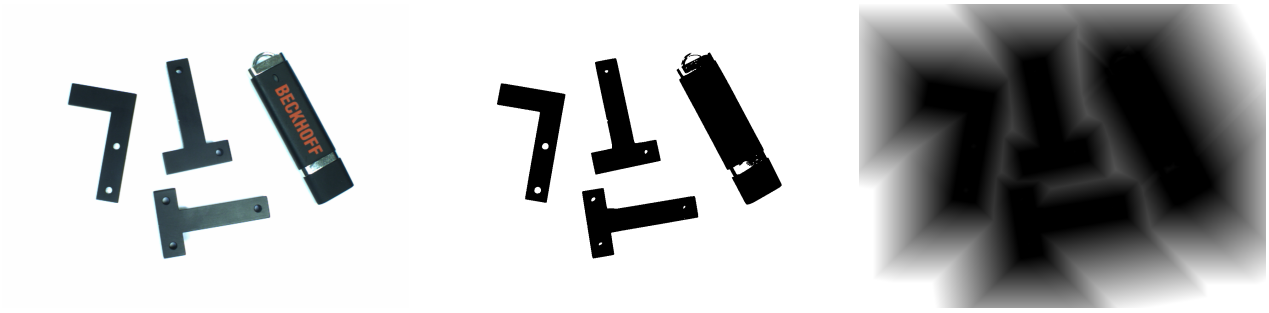
The mask size has no effect with the distance types L1 and C. Instead, a 3x3 mask is always used internally.

Application

An application of this function can look like this, for example:

```
// 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);
```

Original image | **Binary image** | **Result 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 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 [▶ 411]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 196]	Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 195]	Size of the distance transformation mask (PRECISE not supported for label computation)
ipDestLabels	Reference To ITcVnImage [▶ 411]	Returns the component labels (Discrete Voronoi diagram; DINT, 1 channel. An appropriate image will be created if required.)
eLabelType	ETcVnDistanceTransformationLabel [▶ 195]	Type of the labels
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

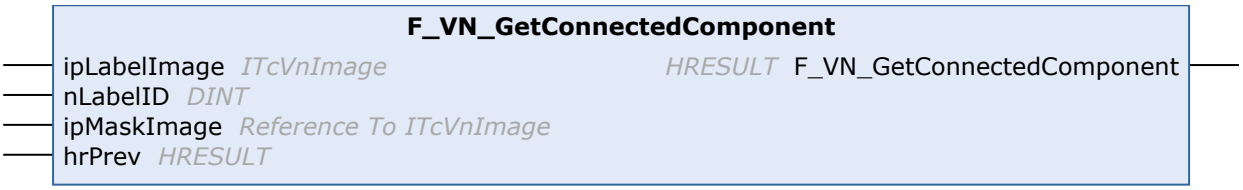
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of the requested label
ipMaskImage	Reference To ITcVnImage [▶ 411]	Returns a mask image marking the pixels that belong to the component (1 channel USINT.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

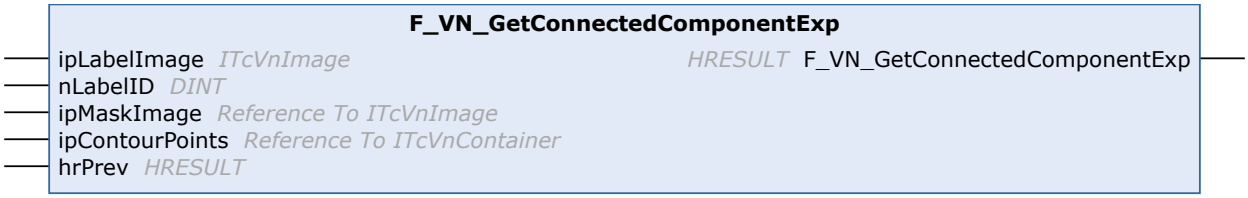
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of the requested label
ipMaskImage	Reference To ITcVnImage [▶ 411]	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 [▶ 370]	Returns the outer contour of the component(ContainerType_Vector_TcVnPoint2_DINT. Contour is optional, set to 0 if not required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

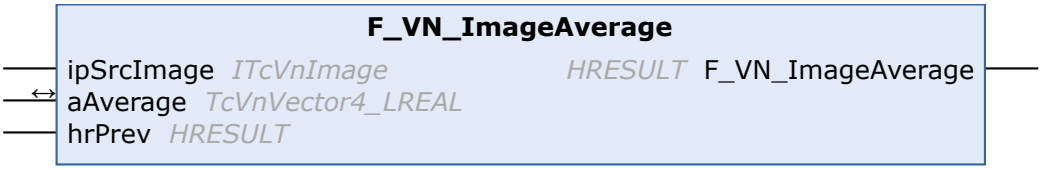
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) average pixel value of ipSrcImage

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) average pixel value of ipSrcImage

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) pixel value standard deviation of ipSrcImage

Return value

HRESULT [▶ 141]

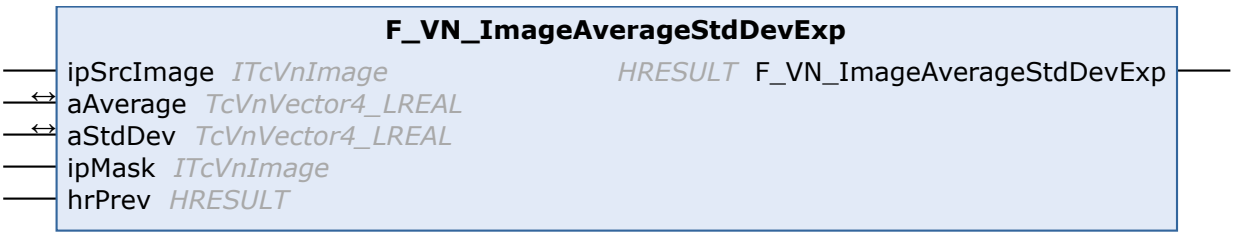
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 162]	Returns the (channel-wise) pixel value standard deviation of ipSrcImage

Return value

HRESULT [▶ 141]

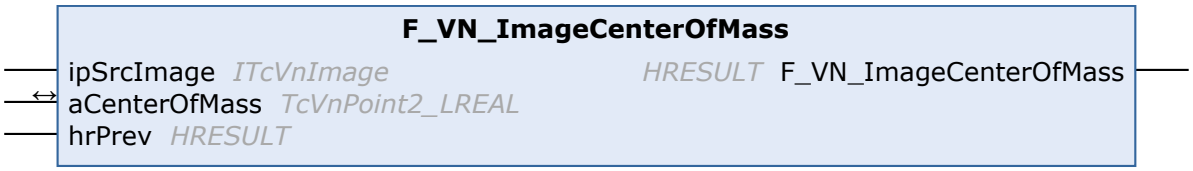
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 160]	Returns the center of mass of the image

Return value

HRESULT [▶ 141]

Further information

This function calculates the center of mass of the input image. The expert version F_VN_ImageCenterOfMassExp [▶ 1223] exists with advanced parameterization options.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel image. If you wish to calculate the center of mass of a multi-channel image, you have the following options:

- You can split the multi-channel image into individual channels using F_VN_SplitImageChannels [▶ 811] and calculate a center of mass for each channel.
- You can convert the multi-channel image into a 1-channel image using F_VN_ConvertColorSpace [▶ 1241] and calculate a center of mass for it.

Center of mass (return value)

The calculated center of mass is returned via the reference `aCenterOfMass`.

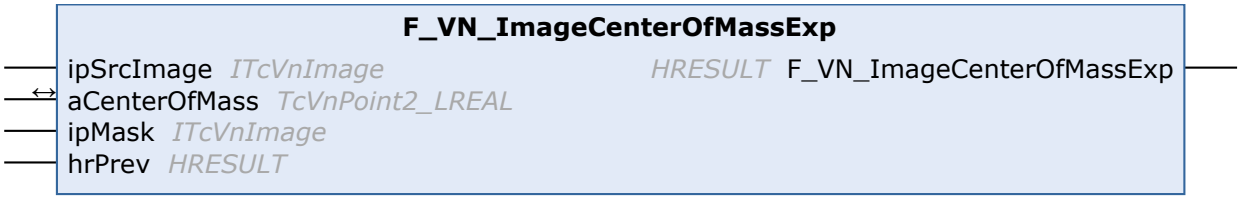
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel)
ipMask	ITcVnImage [▶ 411]	Optional mask (1 channel of type USINT, same width and height as ipSrcImage)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 160]	Returns the center of mass of the image

Return value

HRESULT [▶ 141]

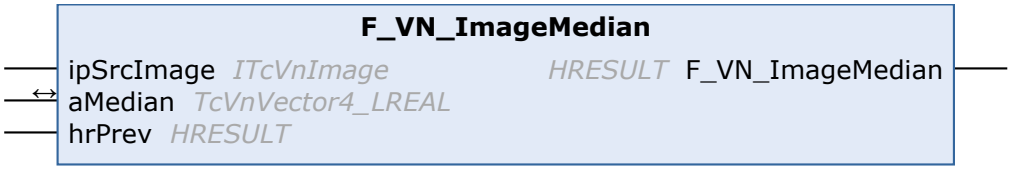
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 162]	Returns the (approximated, channel-wise) median pixel value of ipSrcImage

Return value

HRESULT [▶ 141]

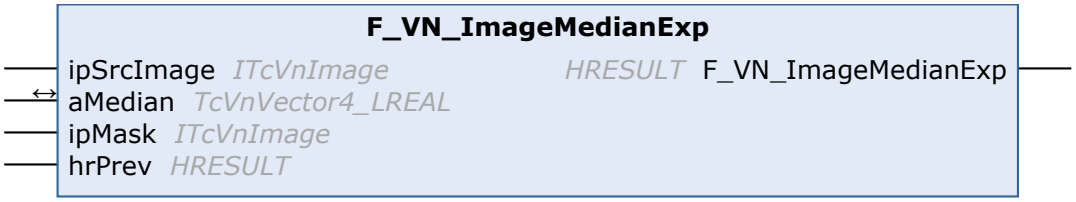
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 411]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 162]	Returns the (approximated, channel-wise) median pixel value of ipSrcImage

Return value

HRESULT [▶ 141]

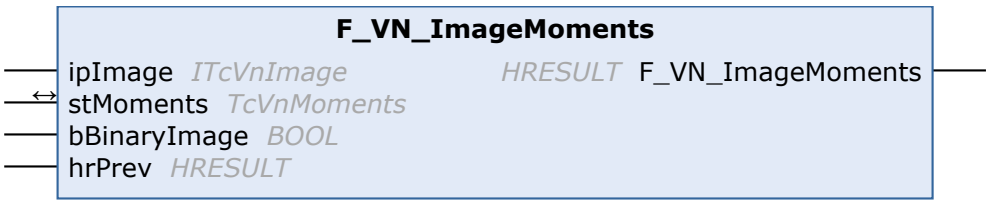
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, elements of type USINT or REAL)
bBinaryImage	BOOL	If true, all non-zero pixels are treated as ones
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 232]	Returns a struct containing the moments

Return value

HRESULT [▶ 141]

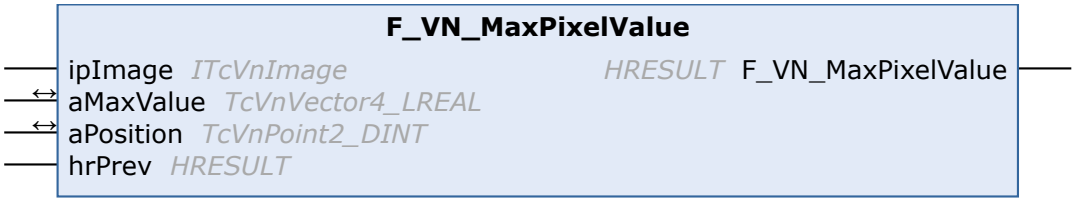
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMaxValue	TcVnVector4_LREAL [▶ 162]	Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 160]	Returns the first found position of aMaxValue

Return value

HRESULT [▶ 141]

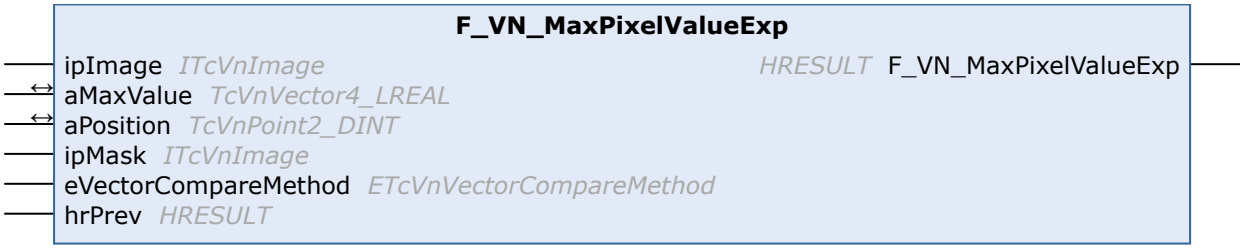
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipMask	ITcVnImage [▶ 411]	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 222]	Select a vector compare method for multi-channel images
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMaxValue	TcVnVector4_LREAL [▶ 162]	Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 160]	Returns the first found position of aMaxValue (not supported for multi-channel images with ELEMENTWISE)

Return value

HRESULT [▶ 141]

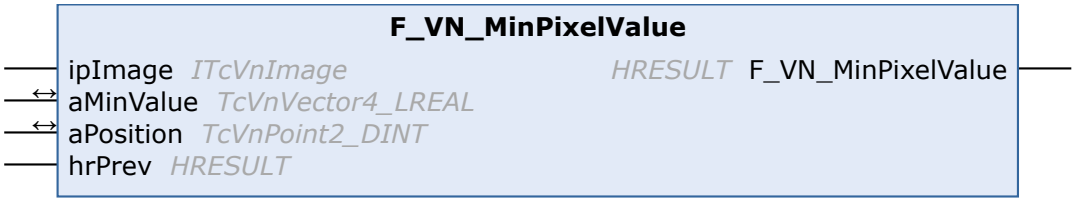
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMinValue	TcVnVector4_LREAL [▶ 162]	Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 160]	Returns the first found position of aMinValue

Return value

HRESULT [▶ 141]

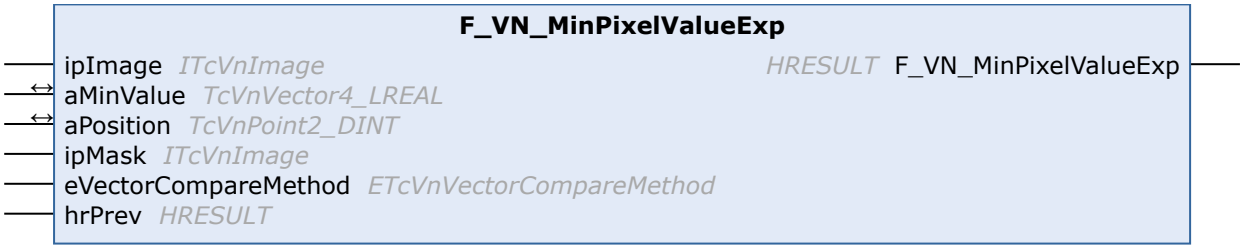
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipMask	ITcVnImage [▶ 411]	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 222]	Select a vector compare method for multi-channel images
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMinValue	TcVnVector4_LREAL [▶ 162]	Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 160]	Returns the first found position of aMinValue (not supported for multi-channel images with ELEMENTWISE)

Return value

HRESULT [▶ 141]

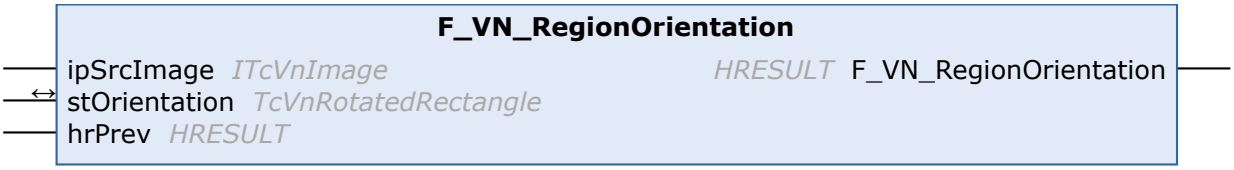
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, binary).
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, binary)
eMethod	ETcVnOrientationMethod [▶ 211]	Method for calculating the orientation.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 245]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.

Return value

HRESULT [▶ 141]

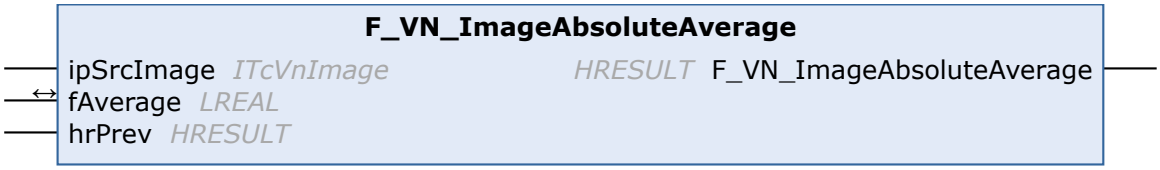
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

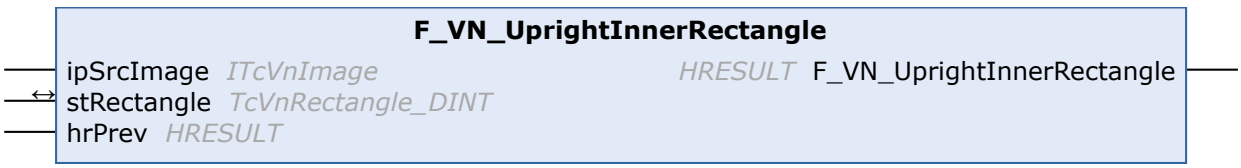
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 1 channel, binary, values > 0 mark valid area)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 244]	Returns the maximum area upright inner rectangle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 244]	Returns the maximum area upright inner rectangle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15 Image Color and Contrast Processing

This group contains functions for color and contrast processing.

Functions

Frequency distribution, intensity and contrast Adjustment of an image

- F_VN_Clahe(Exp) [▶ 1239]
- F_VN_Histogram(Exp) [▶ 1244]
- F_VN_HistogramEqualization(Exp) [▶ 1245]
- F_VN_NormalizeImage(Exp) [▶ 1251]

- F_VN_NormalizeImageForDisplay [▶ 1253]

Look-Up Table and Color Maps

- F_VN_ApplyColorMap [▶ 1236]
- F_VN_ApplyLut [▶ 1238]
- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]

Color Matching

- F_VN_ReferenceColorSimilarity(Exp) [▶ 1254]
- F_VN_TrainImageColor(Exp) [▶ 1261]
- F_VN_TrainImageColor(Exp)_ITcVnMlModel [▶ 1263]

White balance

- F_VN_WhiteBalance [▶ 1275]

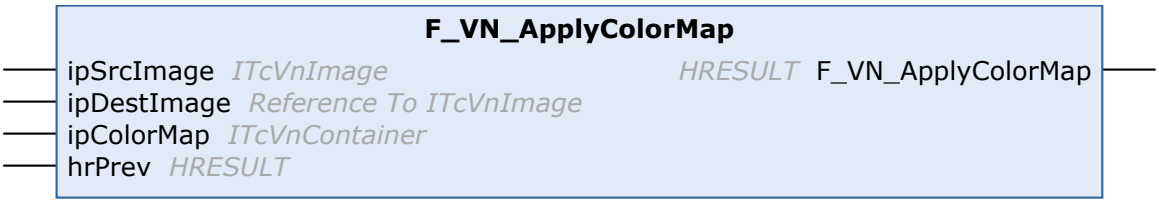
Miscellaneous

- F_VN_ConvertColorSpace [▶ 1241]
- F_VN_InvertImageColor(Exp) [▶ 1249]

Samples

Image Color and Contrast Processing [▶ 2800]

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 [▶ 411]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the colored image (TCVN_ET_USINT, 3 channel. An appropriate image will be created if required.)
ipColorMap	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ApplyColorMap` applies a color table to the input image.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image and have an element type of `USINT` (8-bit) or `UINT` (16-bit).

Result image

The result image `ipDestImage` returns the 3-channel RGB color image with the element type `USINT` (8-bit) that was transformed with the color table.

Color table

The color table `ipColorMap` defines how the input image is transformed. The color table is a container that can be created either manually or with one of the following two functions:

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]

The container must contain 256 elements (for 8-bit) or 65,536 elements (for 16-bit), depending on the element type of the input image, and must be of the type `ContainerType_Vector_TcVnVector3_REAL`.

Application

The application of a typical heat map to an 8-bit image looks like this, for example:

```
hr := F_VN_GenerateColorMap(ipColorMap, TCVN_CMM_HOT, TCVN_CMS_256, hr);
hr := F_VN_ApplyColorMap(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    ipColorMap      :=  ipColorMap,
    hrPrev          :=  hr
);
```

Samples

- Display with color tables [▶ 2803]

Related functions

- F_VN_GenerateColorMap [▶ 1241]

- F_VN_GenerateCustomColorMap [▶ 1243]
- F_VN_ApplyColorMap [▶ 1236]

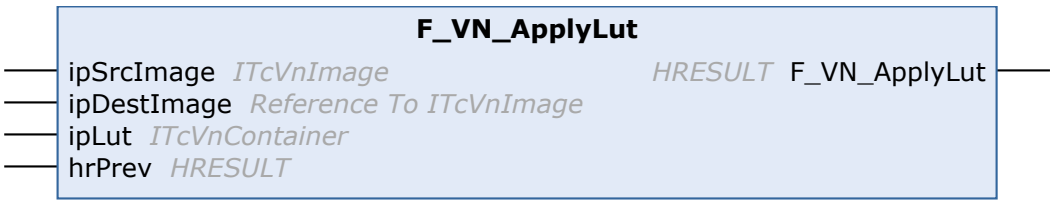
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1-4 channels of 8 or 16 bit types SINT, USINT, INT, UINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the result image (same size, type and channels as ipSrcImage. An appropriate image will be created if required.)
ipLut	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the resulting image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

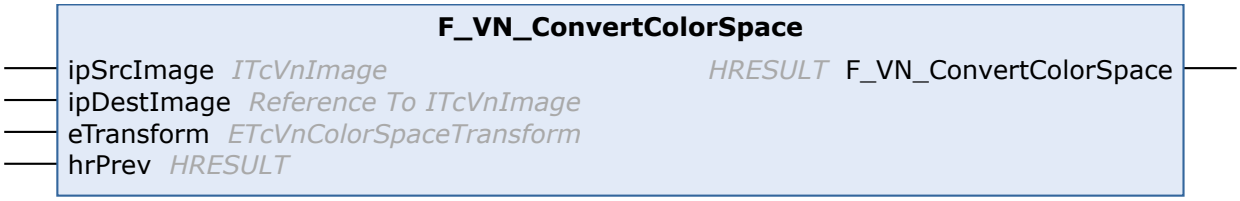
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eTransform	ETcVnColorSpaceTransform [▶ 175]	Transforms to be applied
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

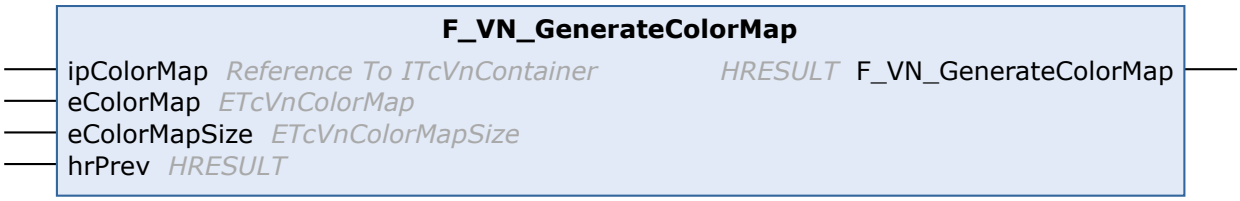
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
eColorMap	ETcVnColorMap [▶ 170]	Selects a color map (similar to GNU Octave/MATLAB types)
eColorMapSize	ETcVnColorMapSize [▶ 175]	Defines how many elements the generated color map should have
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GenerateColorMap` creates a color table that can be applied to an image with the function F_VN_ApplyColorMap [▶ 1236].

Parameter

Color Map

The parameter `ipColorMap` returns the color table created.

Color gradient

The color gradient `eColorMap` of the color table is defined via the enum ETcVnColorMap [▶ 170]. If you wish to define a color gradient manually, use the function F_VN_GenerateCustomColorMap [▶ 1243].

Size of the color table

The size `eColorMapSize` is specified via the enum ETcVnColorMapSize [▶ 175] and defines whether the color table is created for 8-bit images (`TCVN_CMS_256`) or for 16-bit images (`TCVN_CMS_65536`).

Application

```
VAR
    ipColorMap      :   ITcVnColorMap;
END_VAR

hr := F_VN_GenerateColorMap(
    ipColorMap      :=  ipColorMap,
    eColorMap       :=  TCVN_CM_HOT,
    eColorMapSize   :=  TCVN_CMS_256,
    hrPrev          :=  hr
);
```

Samples

- Display with color tables [▶ 2803]

Related functions

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]
- F_VN_ApplyColorMap [▶ 1236]

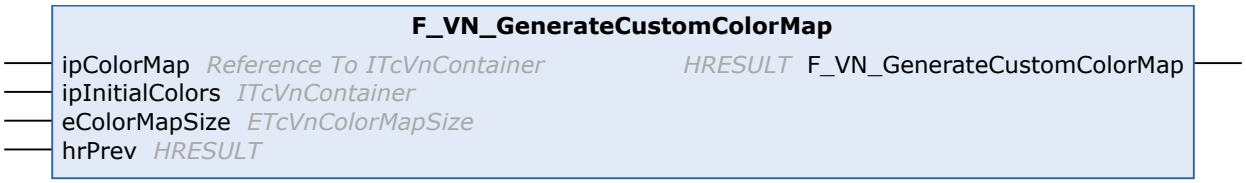
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
ipInitialColors	ITcVnContainer [▶ 370]	User defined colors (ContainerType_Vector_TcVnVector3_REAL, at least 2 elements)
eColorMapSize	ETcVnColorMapSize [▶ 175]	Defines how many elements the generated color map should have
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Samples

- Display with color tables [▶ 2803]

Related functions

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_GenerateCustomColorMap [▶ 1243]
- F_VN_ApplyColorMap [▶ 1236]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestHistogram	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_Histogram` calculates the histogram of the input image channel by channel.

Parameter

Input image

The input image `ipSrcImage` can have any format.

Histogram

The calculated multichannel histogram is returned as a container `ipDestHistogram` of the type `ContainerType_Vector_Vector_UDINT`. Each sub-element corresponds to the histogram of one channel. Each container of the histogram is represented as a `UDINT`, which specifies the number of pixels belonging to the container.

Expert parameters

The expert version F_VN_HistogramExp [▶ 1247] contains additional parameters.

Application

The determination of the most frequently occurring pixel value in a grayscale image looks like this, for example:

```
VAR
    ipDestHistograms    :   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.
```

Samples

- Handling histograms

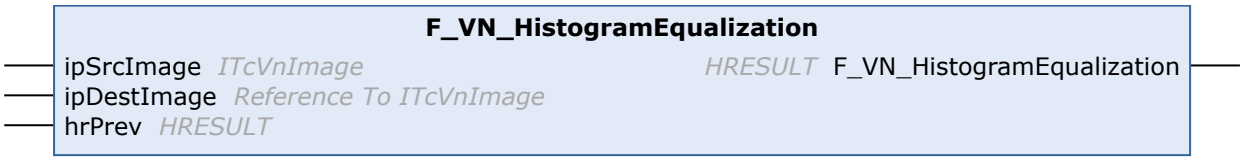
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

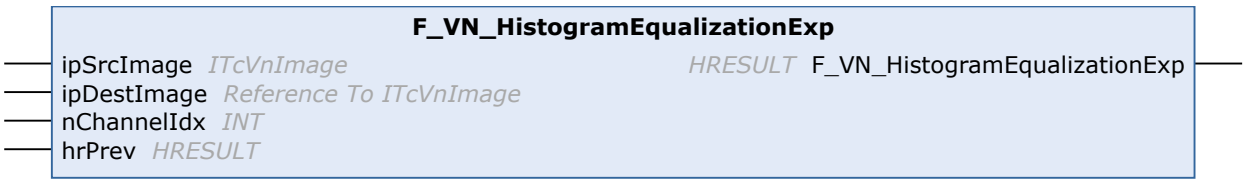
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestHistogram	Reference To ITcVnContainer [▶ 370]	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 [▶ 411]	Source image mask (TCVN_ET_USINT, 1 channel. Mask is optional, set to 0 if not required.)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_HistogramExp` is the expert version of F_VN_Histogram [▶ 1244]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` can have any format.

Histogram

The calculated multichannel histogram is returned as a container `ipDestHistogram` of the type `ContainerType_Vector_Vector_UDINT`. Each sub-element corresponds to the histogram of one channel. Each container of the histogram is represented as a `UDINT`, which specifies the number of pixels belonging to the container.

Container

The number `nBins` defines how many containers the observed pixel spectrum is to be divided into.

Limits

The lower limit `fLowerBound` and the upper limit `fUpperBound` define which pixel values are taken into account when creating the histogram.

Mask

The mask `ipMask` defines which pixels of the input image `ipSrcImage` are taken into account in the calculation of the histogram. It is described by a 1-channel image with the element type `USINT`, which has the same size as the input image `ipSrcImage`. All pixels in the mask image with a value `<> 0` are taken into account in the calculation of the histogram.

Application

The calculation of a histogram with 10 containers for the value range 100-200 with a circular mask looks like this, for example:

```
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
);
```

Samples

- Handling histograms

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

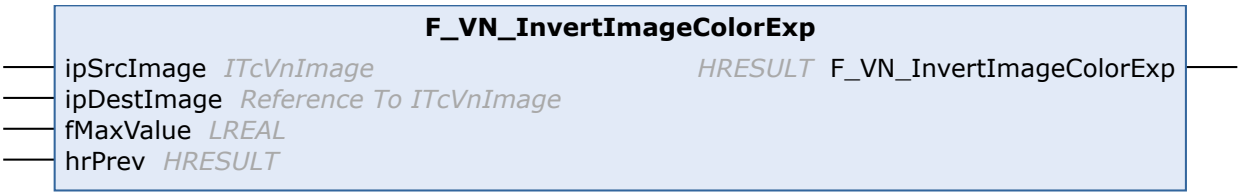
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

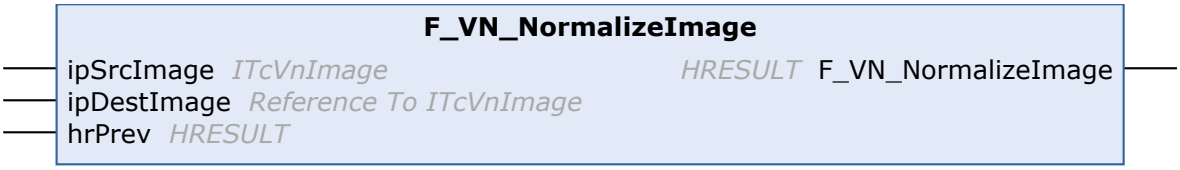
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

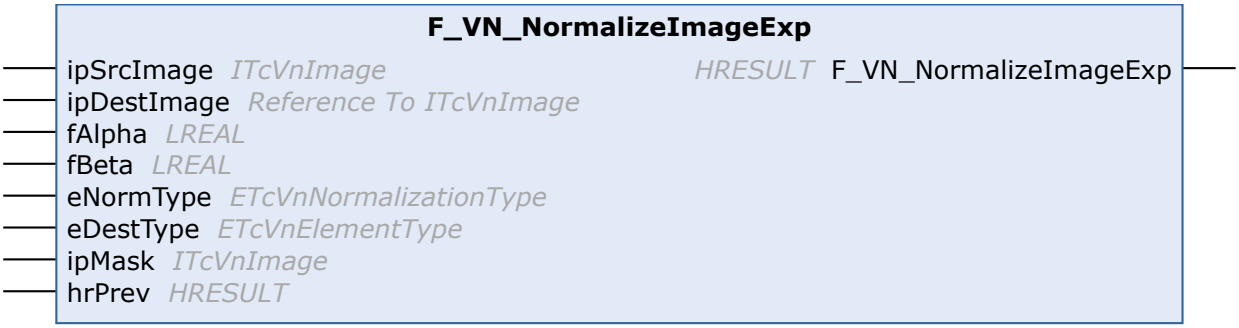
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 210]	Normalization type (only INF, L1, L2 or MINMAX)
eDestType	ETcVnElementType [▶ 199]	Destination image depth (usually SAME_AS_SOURCE)
ipMask	ITcVnImage [▶ 411]	Mask to restrict the normalization to specific pixel positions (set 0 to normalize the whole image)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

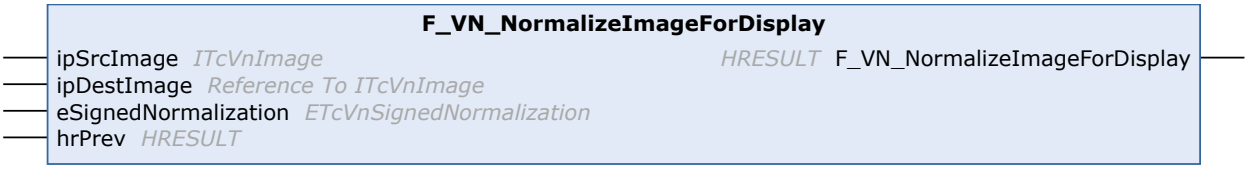
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eSignedNormalization	ETcVnSignedNormalization [▶ 216]	Option for normalizing signed values
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

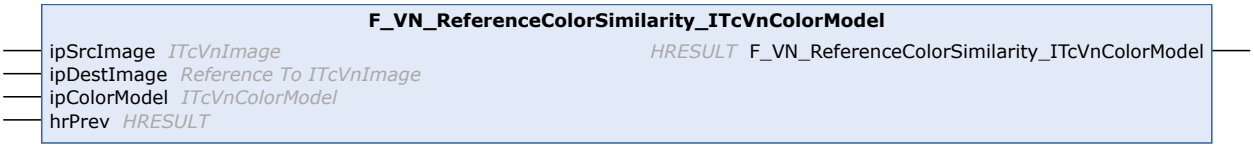
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel [▶ 418]	Color model
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

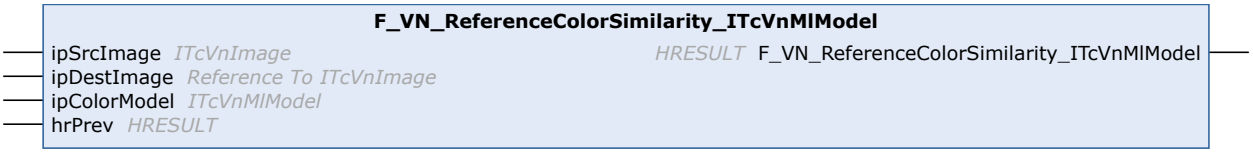
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel [▶ 420]	Color model
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

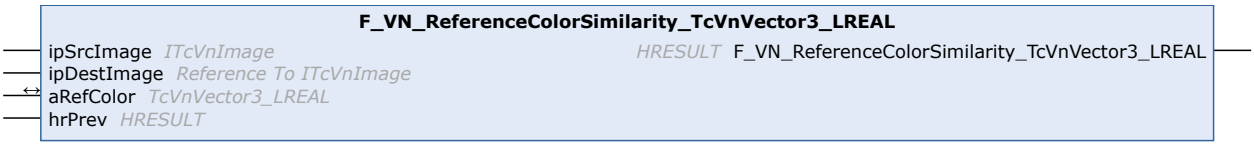
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRefColor	TcVnVector3_LREAL [▶ 162]	Reference color (RGB, [0..255])

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel [▶ 418]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel [▶ 420]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

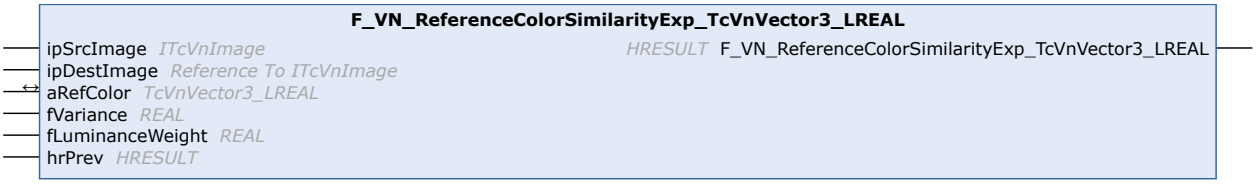
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRefColor	TcVnVector3_LREAL [▶ 162]	Reference color (RGB, [0..255])

Return value

HRESULT [▶ 141]

Further information

There are three ways to pass the reference color to the corresponding function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`, see sample

To train a color model, a corresponding F_VN_TrainImageColor [▶ 1261] function must be used. The two color models differ only in the data type, functionally there are no differences. The `ITcVnMlModel` is the general data type for ML models for which there are also function blocks for loading and saving. Therefore, this data type and the corresponding functions should be used if the trained color model is to be saved to disk.

Samples

- Color similarity with RGB reference color [▶ 2800]

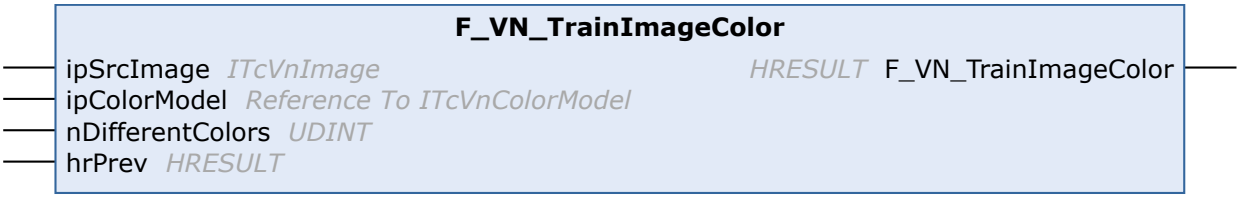
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 418]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColor` trains a color model on the basis of a reference image. If the trained color model is to be saved to disk, the F_VN_TrainImageColor_ITcVnMlModel [▶ 1263] with the type `ITcVnMlModel` must be used.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of the type ITcVnColorModel [▶ 418].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Expert parameters

Further parameters can be found in the expert version F_VN_TrainImageColorExp [▶ 1264].

Application

The training of a color model for a reference image with 4 differently colored objects and a dark background looks like this, for example:

```
VAR
    ipColorModel    :   ITcVnColorModel;
END_VAR

hr := F_VN_TrainImageColor(
            ipSrcImage      := ipImageRef,
            ipColorModel    := ipColorModel,
            nDifferentColors:= 5, // 4 colors + 1 background = 5
            hrPrev          := hr);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model
- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

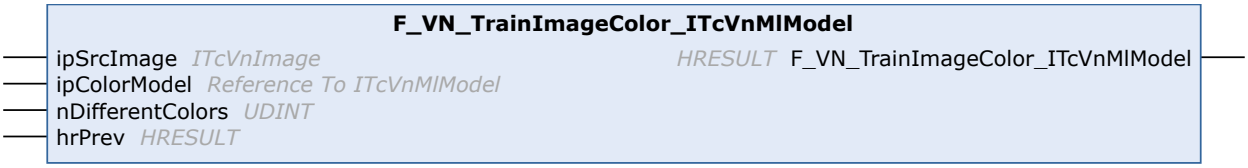
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 420]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColor_ITcVnMlModel` trains a color model on the basis of a reference image. The color model can be saved and loaded on the hard disk with the corresponding function blocks.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of type ITcVnMlModel [▶ 420].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Expert parameters

Further parameters can be found in the expert version F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1267].

Application

The training of a color model for a reference image with 4 differently colored objects and a dark background looks like this, for example:

```
VAR
    ipColorModel    :   ITcVnMlModel;
END_VAR

hr := F_VN_TrainImageColor_ITcVnMlModel(
            ipSrcImage      := ipImageRef,
            ipColorModel    := ipColorModel,
            nDifferentColors:= 5, // 4 colors + 1 background = 5
            hrPrev          := hr);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model
- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

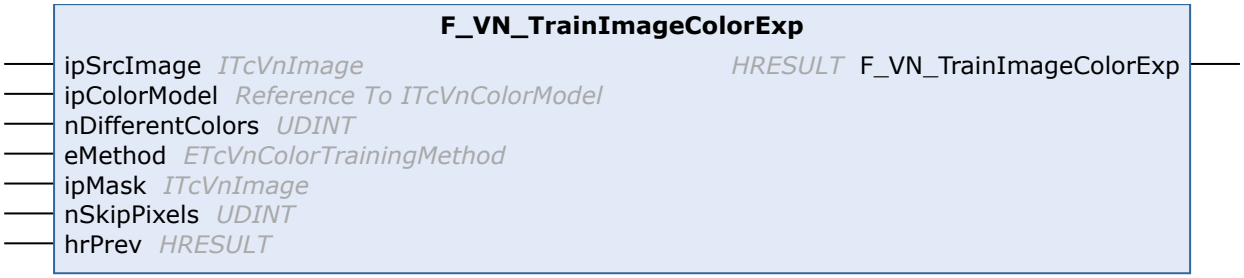
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 418]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
eMethod	ETcVnColorTrainingMethod [▶ 187]	Color training method
ipMask	ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColorExp` is the expert version of F_VN_TrainImageColor [▶ 1261]. It contains further parameters. If the trained color model is to be saved to disk, the F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1267] with the type `ITcVnMlModel` must be used.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of the type ITcVnColorModel [▶ 418].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Expert parameters

Additional parameters can be found in another expert version F_VN_TrainImageColorExp2 [▶ 1269].

Training methods

When selecting the training method, the LAB and RGB color spaces are available.

Mask

The mask `ipMask` defines which pixels of the input image `ipSrcImage` are taken into account in the calculation. It is described by a 1-channel image with the element type USINT, which has the same size as the input image `ipSrcImage`. All pixels in the mask image with a value <> 0 are taken into account in the calculation of the cluster.

Pixel gap

For performance optimization, a number of pixels can be skipped so that fewer pixels need to be considered when dividing into the individual clusters.

Application

```
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);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model
- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 420]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
eMethod	ETcVnColorTrainingMethod [▶ 187]	Color training method
ipMask	ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColorExp_ITcVnMlModel` is the expert version of F_VN_TrainImageColor_ITcVnMlModel [▶ 1263]. It contains further parameters. The color model can be saved and loaded on the hard disk with the corresponding function blocks.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of type ITcVnMlModel [▶ 420].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Expert parameters

Additional parameters can be found in another expert version F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1272].

Training methods

When selecting the training method, the LAB and RGB color spaces are available.

Mask

The mask `ipMask` defines which pixels of the input image `ipSrcImage` are taken into account in the calculation. It is described by a 1-channel image with the element type USINT, which has the same size as the input image `ipSrcImage`. All pixels in the mask image with a value <> 0 are taken into account in the calculation of the cluster.

Pixel gap

For performance optimization, a number of pixels can be skipped so that fewer pixels need to be considered when dividing into the individual clusters.

Application

```
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);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model

- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

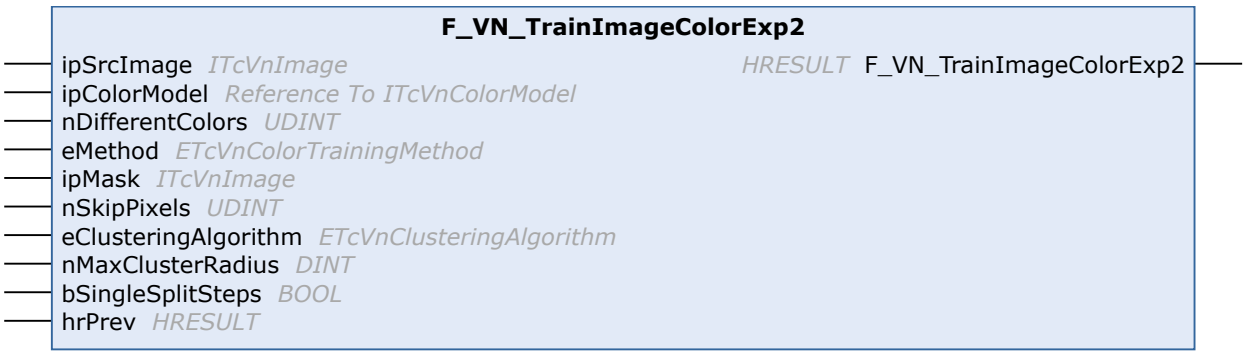
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 418]	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 [▶ 187]	Color training method
ipMask	ITcVnImage [▶ 411]	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 [▶ 169]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColorExp2` is the extended expert version of F_VN_TrainImageColorExp [▶ 1264]. It contains further parameters. If the trained color model is to be saved to disk, the F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1272] with the type `ITcVnMlModel` must be used.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of the type ITcVnColorModel [▶ 418].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Clustering algorithm

When selecting the clustering algorithm, KMeans++ (fixed number of clusters) and LBG (dynamic number of clusters) are available.

Maximum cluster radius

Used only with the LBG cluster algorithm. This can be used to influence the distribution of the individual clusters.

Global optimization

Used only with the LBG cluster algorithm. This allows a choice between shorter execution time (`bSingleSplitSteps = FALSE`) and improved results (`bSingleSplitSteps = TRUE`).

Less frequent use of global optimization is faster, but may lead to less optimal results.

Application

```
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);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model
- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 420]	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 [▶ 187]	Color training method
ipMask	ITcVnImage [▶ 411]	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 [▶ 169]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainImageColorExp2_ITcVnMlModel` is the extended expert version of F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1267]. It contains further parameters. The color model can be saved and loaded on the hard disk with the corresponding function blocks.

Parameter

Reference image

The reference image `ipSrcImage` must have 3 channels in RGB format and the element type `USINT` (8-bit).

Color model

The parameter `ipColorModel` returns the trained color model. This is an interface pointer of type ITcVnMlModel [▶ 420].

Number of colors to be distinguished

The number `nDifferentColors` defines the number of colors to be trained. Note that the image background, for example, can also have its own color.

Clustering algorithm

When selecting the clustering algorithm, KMeans++ (fixed number of clusters) and LBG (dynamic number of clusters) are available.

Maximum cluster radius

Used only with the LBG cluster algorithm. This can be used to influence the distribution of the individual clusters.

Global optimization

Used only with the LBG cluster algorithm. This allows a choice between shorter execution time (`bSingleSplitSteps = FALSE`) and improved results (`bSingleSplitSteps = TRUE`).

Less frequent use of global optimization is faster, but may lead to less optimal results.

Application

```
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);
```

Related functions

There are three ways to pass the reference color to the corresponding `F_VN_ReferenceColorSimilarity` function:

- As a trained color model of the type `ITcVnColorModel`
- As a trained color model of the type `ITcVnMlModel`
- Directly with values in a `TcVnVector3_LREAL`

For the color models there are corresponding functions for training and for all three data types one function each for execution:

- F_VN_TrainImageColor [▶ 1261] for the training of a color model
- F_VN_ReferenceColorSimilarity [▶ 1254] for segmentation by means of a color model

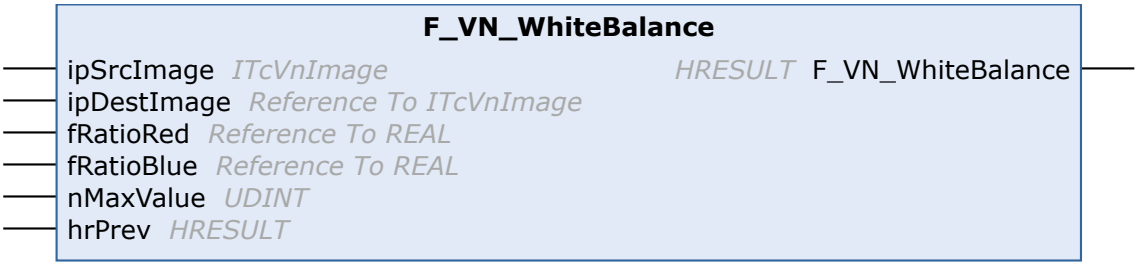
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (3 channels (RGB), USINT or UINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16 Image Filtering

This group contains functions for filtering images.

Functions

Blurring

- F_VN_BilateralFilter(Exp) [▶ 1276]
- F_VN_BoxFilter(Exp) [▶ 1281]
- F_VN_GaussianFilter(Exp) [▶ 1295]
- F_VN_MedianFilter [▶ 1306]

Edge detection

- F_VN_LaplacianFilter(Exp) [▶ 1299]
- F_VN_ScharrFilter(Exp) [▶ 1313]
- F_VN_SobelFilter(Exp) [▶ 1321]

User-defined filters

- F_VN_CustomFilter(Exp) [▶ 1287]
- F_VN_SeparableCustomFilter(Exp) [▶ 1317]

Fill areas

- F_VN_BrightBorderObjects [▶ 1283]
- F_VN_DarkBorderObjects [▶ 1292]
- F_VN_FillHoles [▶ 1293]

Morphological operator

- F_VN_CreateStructuringElement [▶ 1285]
- F_VN_MorphologicalOperator [▶ 1309]

Local extremes

- F_VN_LocalMaxima [▶ 1303]
- F_VN_LocalMinima [▶ 1305]
- F_VN_RemoveLocalMaxima [▶ 1310]
- F_VN_RemoveLocalMinima [▶ 1312]

Local variance

- F_VN_VarianceFilter(Exp) [▶ 1325]

Samples

Image filtering [▶ 2805]

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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_BilateralFilter` applies a bilateral filter to the original image. The filter smooths the image and reduces noise while retaining the edges in the image.

Parameter

Original image

The original image `ipSrcImage` must be of the element type `TCVN_ET_USINT` or a floating point type and have 1 or 3 channels.

Result image

The result image `ipDestImage` must be a different interface pointer to the original image `ipSrcImage`!

Filter diameter

The filter diameter `nDiameter` determines the size of a filter area. An odd number is expected. If `nDiameter = 0`, then the filter diameter is automatically calculated on the basis of `fSigmaSpace`.

Color area

This value may not be `0`.

Room area

This value may not be `0`.

Expert parameters

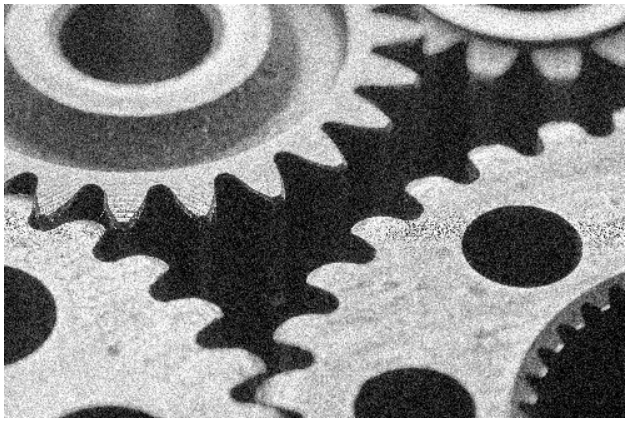
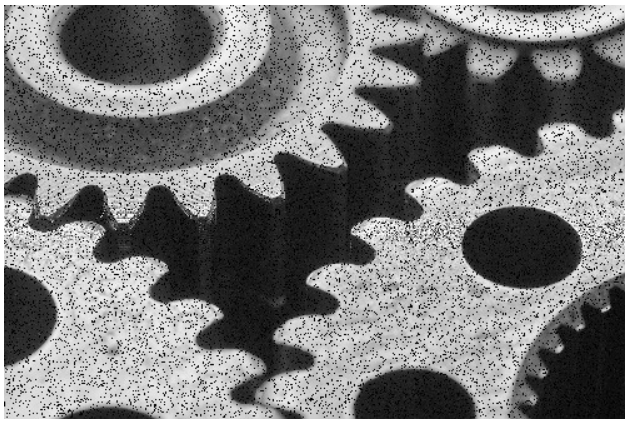
Further parameters can be found in the expert version F_VN_BilateralFilterExp [▶ 1280].

Application

An application of this function can look like this, for example:

```
hr := F_VN_BilateralFilter(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImageRes,
    nDiameter       :=  nDiameter,
    fSigmaColor     :=  fSigmaColor,
    fSigmaSpace     :=  fSigmaSpace,
    hrPrev          :=  hr
);
```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

Original images	Result images: Diameter = 7 Sigma = 100
	
	
	

Samples

- Blur filter [▶ 2805]

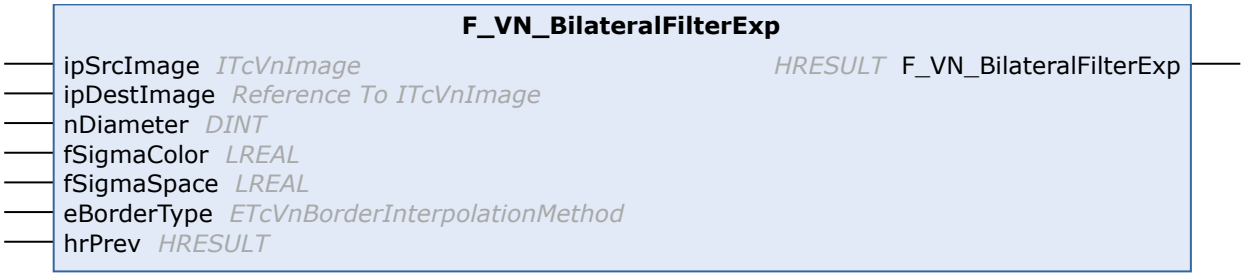
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_BilateralFilterExp` is the expert version of F_VN_BilateralFilter [▶ 1276]. It contains additional parameters.

Parameter

Original image

The original image `ipSrcImage` must be of the element type `TCVN_ET_USINT` or a floating point type and have 1 or 3 channels.

Result image

The result image `ipDestImage` must be a different interface pointer to the original image `ipSrcImage`!

Filter diameter

The filter diameter `nDiameter` determines the size of a filter area. An odd number is expected. If `nDiameter = 0`, then the filter diameter is automatically calculated on the basis of `fSigmaSpace`.

Color area

This value may not be `0`.

Room area

This value may not be `0`.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

An application of this function can look like this, for example:

```
hr := F_VN_BilateralFilterExp(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImageRes,
    nDiameter       :=  nDiameter,
    fSigmaColor     :=  fSigmaColor,
    fSigmaSpace     :=  fSigmaSpace,
    eBorderType     :=  eBorderType,
    hrPrev          :=  hr
);
```

Samples

- Blur filter [▶ 2805]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

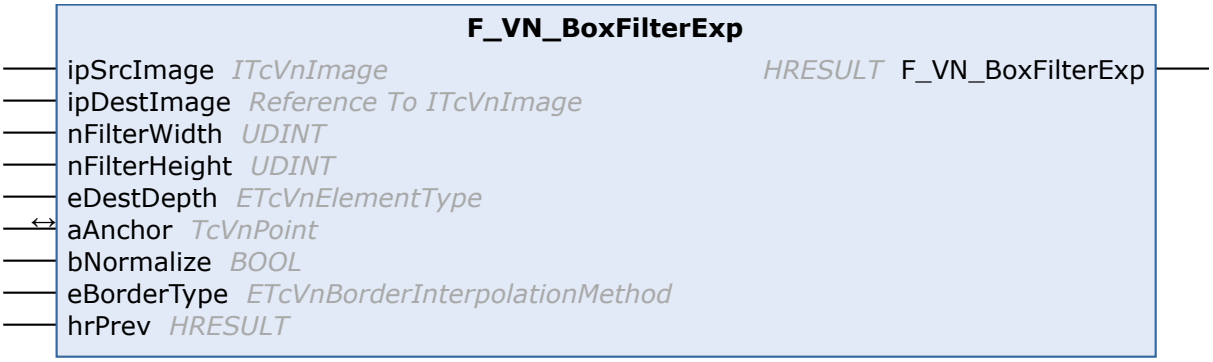
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 199]	Destination image depth
bNormalize	BOOL	If true, the kernel is normalized by nFilterWidth * nFilterHeight
eBorderType	ETcVnBorderInterpolationMethod [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAnchor	TcVnPoint [▶ 160]	Anchor point of the kernel ([-1, -1] for center)

Return value

HRESULT [▶ 141]

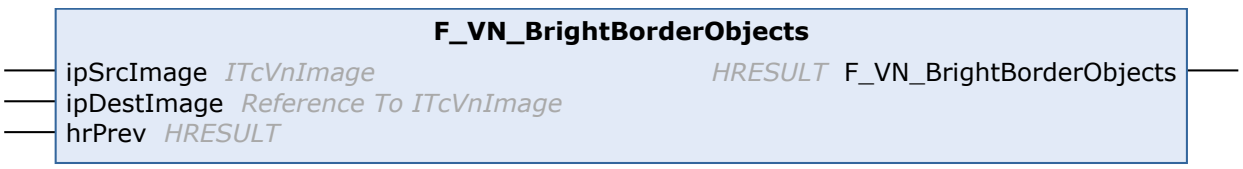
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function transfers all bright border objects in the input image to the result image.

Algorithm

All objects in the image that are brighter than the surrounding pixels are removed.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: `USINT`, `UINT`, `INT`, `REAL` or `LREAL`.

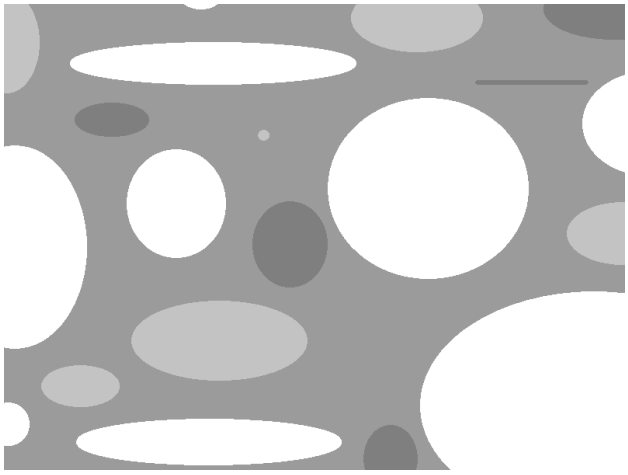
Result image

The result image `ipDestImage` returns the bright border objects.

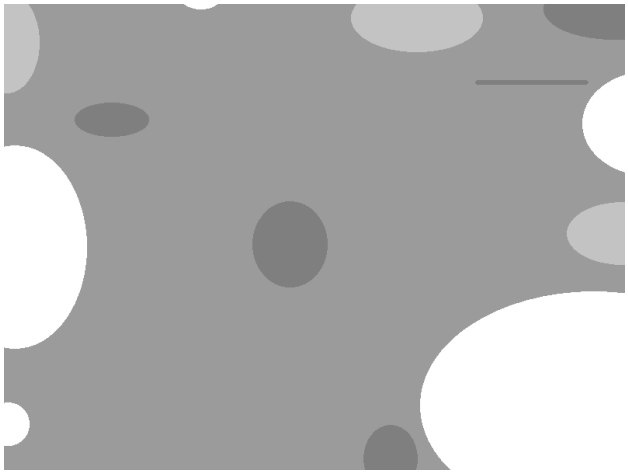
Application

```
hr := F_VN_BrightBorderObjects(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Input image



Result image



Related functions

- F_VN_DarkBorderObjects [▶ 1292]
- F_VN_FillHoles [▶ 1293]

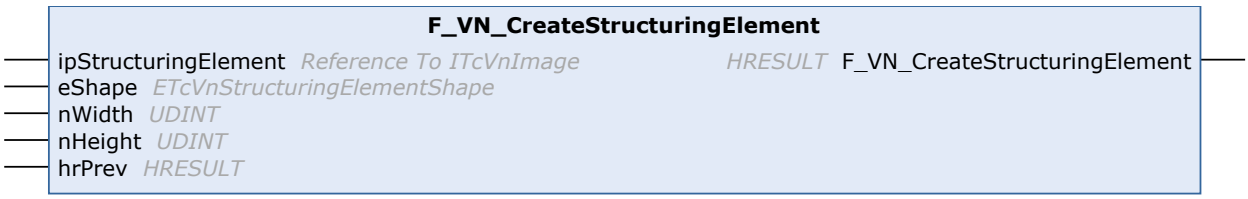
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Returns the created structuring element (Non-zero interface pointers are reused.)
eShape	ETcVnStructuringElementShape [▶ 217]	Shape of the structuring element (rectangle, cross, or ellipse)
nWidth	UDINT	Width
nHeight	UDINT	Height
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function creates a structure element that can be used for morphological operations with F_VN_MorphologicalOperator [▶ 1309].

Parameter

Structure element

The structure element created, `ipStructuringElement`, is returned as ITcVnImage [▶ 411] by representing the structure element with pixels with the value `1`.

Shape

The shape `eShape` of the structure element is defined by the enum ETcVnStructuringElementShape [▶ 217]. The following shapes are available:

TCVN_SES_RECTANGLE



TCVN_SES_CROSS



TCVN_SES_ELLIPSE



As the pixel values of the structure element are 1, they were scaled up here for a better display.

Size

The size of the structure element is defined by the width `nWidth` and height `nHeight` of the image `ipStructuringElement` to be created.

Application

The creation of a square structure element of the size `9` looks like this, for example:

```
hr := F_VN_CreateStructuringElement(
    ipStructuringElement    :=  ipElement,
    eShape                  :=  TCVN_SES_RECTANGLE,
    nWidth                  :=  9,
    nHeight                 :=  9,
    hrPrev                  :=  hr
);
```

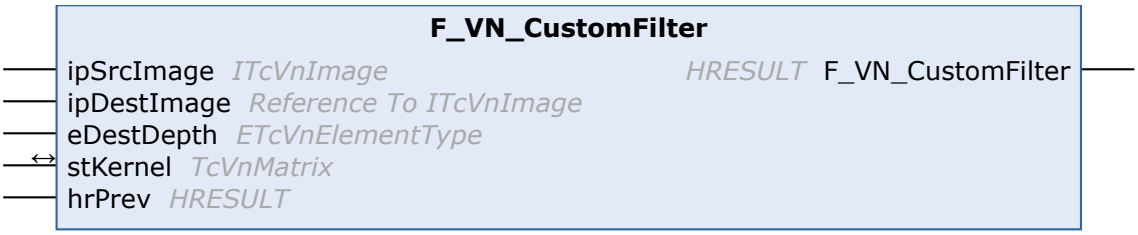
Required License

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernel	TcVnMatrix [▶ 231]	Custom filter kernel with values of type REAL or LREAL

Return value

HRESULT [▶ 141]

Further information

The function F_VN_CustomFilter applies a user-defined filter to the input image. The filter kernel is defined by a matrix. This matrix is folded with the image.

Parameter

Input image

The input image ipSrcImage may have any available format.

Result image

The result image ipDestImage has the same format as the input image ipSrcImage.

Result depth

The result depth eDestDepth defines which element type the result image ipDestImage has. A larger element type can display more information.

Filter kernel

The user-defined filter kernel stKernel is transferred as a matrix. Permitted element types of the matrix are TCVN_ET_REAL and TCVN_ET_LREAL. A corresponding matrix of the type TcVnMatrix [▶ 231] can be created with the aid of the function F_VN_InitMatrixStruct [▶ 1539].

Expert parameters

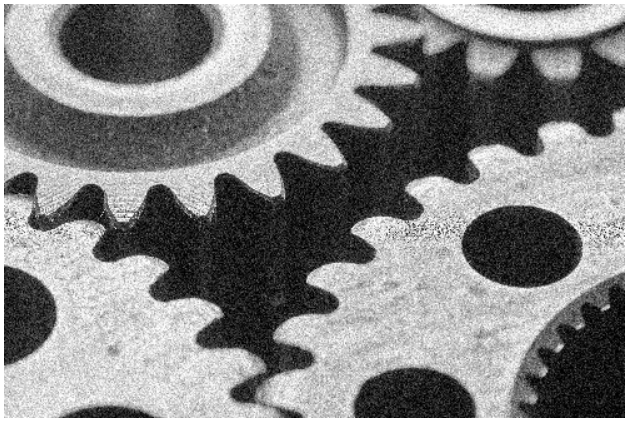
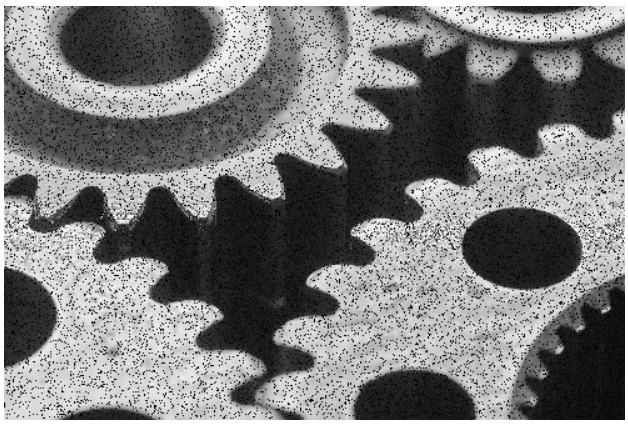
The expert version F_VN_CustomFilterExp [▶ 1290] contains additional parameters.

Application

```
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
);
```

- The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row). An average filter is implemented as an example; a more detailed description can be found in the sample for the Blur filter [▶ 2805]:

Original images	Result images: Mean filter
	
	
	

Samples

- Blur filter [▶ 2805]
- User-defined filters

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
fDelta	LREAL	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernel	TcVnMatrix [▶ 231]	Custom filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 160]	Anchor point of the kernel

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CustomFilterExp` is the expert version of F_VN_CustomFilter [▶ 1287]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` has the same format as the input image `ipSrcImage`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

Filter kernel

The user-defined filter kernel `stKernel` is transferred as a matrix. Permitted element types of the matrix are `TCVN_ET_REAL` and `TCVN_ET_LREAL`. A corresponding matrix of the type TcVnMatrix [▶ 231] can be created with the aid of the function F_VN_InitMatrixStruct [▶ 1539].

Anchor point

The anchor point `aAnchor` defines which point of the filter kernel is moved over each pixel of the input image `ipSrcImage`.

Delta

The constant delta `fDelta` is added to the result of the filter operation.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

```
hr := F_VN_CustomFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernel    :=  stKernelMatrix,
    aAnchor     :=  aAnchorPoint,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr
);
```

Samples

- User-defined filters

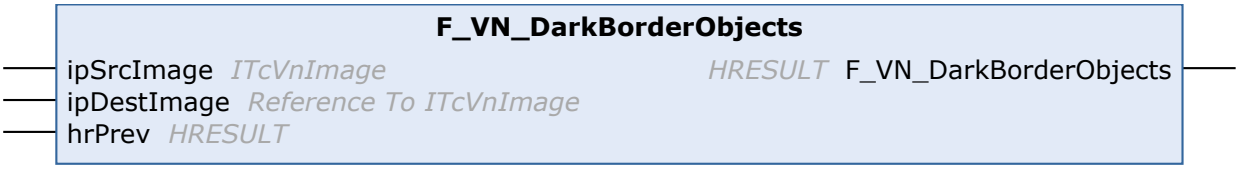
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function transfers all dark border objects in the input image to the result image.

Algorithm

All objects in the image that are darker than the surrounding pixels are removed.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: `USINT`, `UINT`, `INT`, `REAL` or `LREAL`.

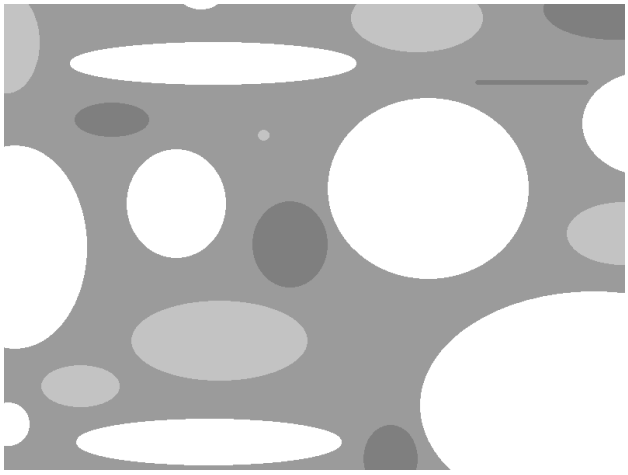
Result image

The result image `ipDestImage` returns the dark border objects.

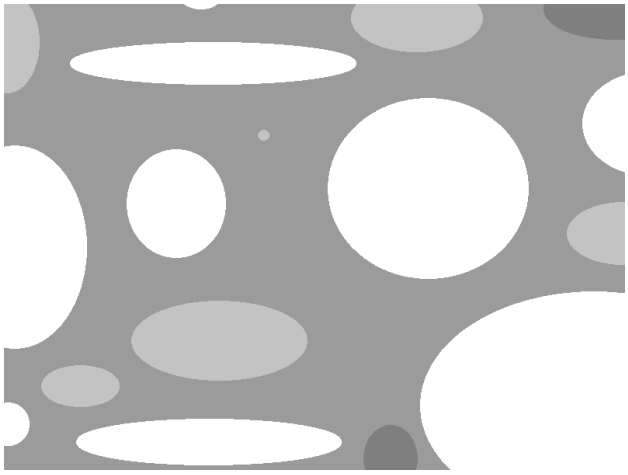
Application

```
hr := F_VN_DarkBorderObjects(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Input image



Result image



Related functions

- F_VN_FillHoles [▶ 1293] als Alias
- F_VN_BrightBorderObjects [▶ 1283]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

This function fills holes in the input image.

Algorithm

All objects in the image that are darker than the surrounding pixels are removed.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: `USINT`, `UINT`, `INT`, `REAL` or `LREAL`.

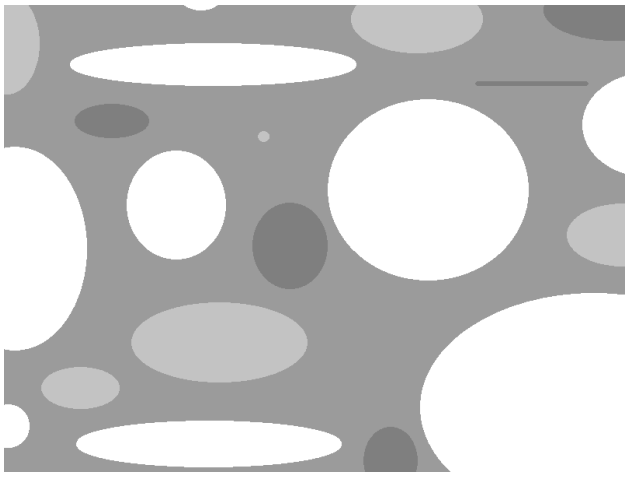
Result image

The result image `ipDestImage` returns the image with the holes filled.

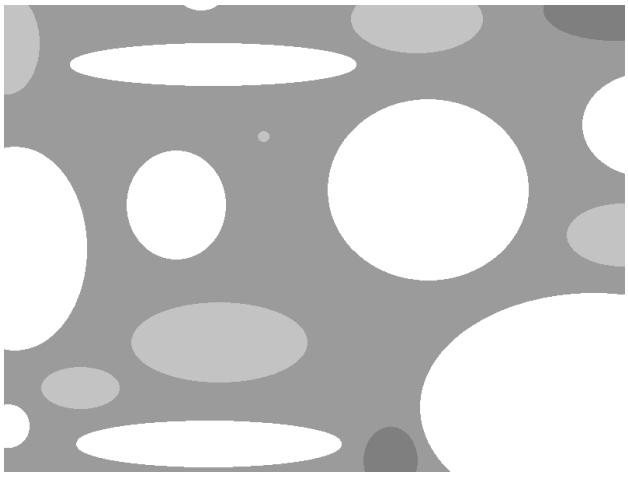
Application

```
hr := F_VN_FillHoles(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Input image



Result image



Related functions

- F_VN_DarkBorderObjects [▶ 1292] as alias
- F_VN_BrightBorderObjects [▶ 1283]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GaussianFilter` applies a Gaussian filter to the input image.

Algorithm

The Gaussian filter is often used in image processing to reduce noise or smooth an image. Each pixel is replaced by the weighted average of its environment, removing smaller details but preserving larger ones. The larger the filter, the more smoothed the image will be (and therefore appears more blurred).

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` is given the same format as the input image `ipSrcImage`.

Filter size

The size of the Gauss filter is described in the X direction by `nFilterWidth` and in the Y direction by `nFilterHeight`. The size specifications must be odd numbers, as there must always be a central pixel.

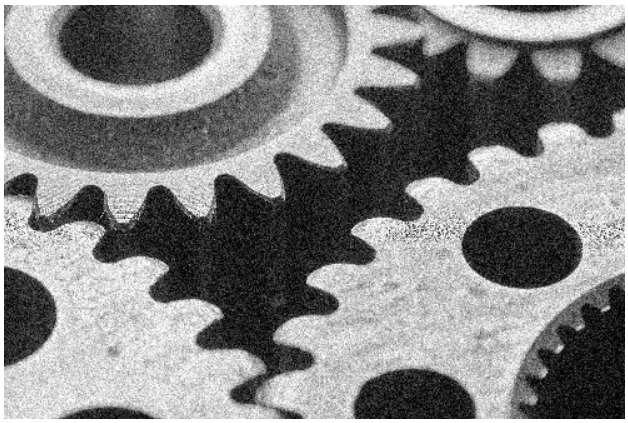
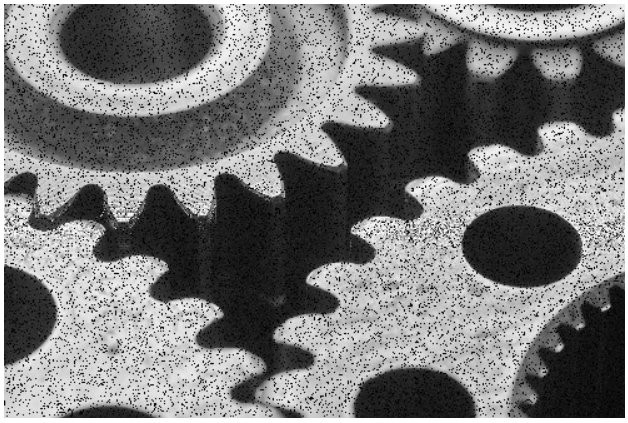
Expert parameters

The expert version F_VN_GaussianFilterExp [▶ 1298] contains additional parameters.

Application

```
hr := F_VN_GaussianFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterWidth    :=  7,
    nFilterHeight   :=  7,
    hrPrev          :=  hr
);
```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

Original images	Result images
	
	
	

Samples

- Blur filter [▶ 2805]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GaussianFilterExp` is the expert version of F_VN_GaussianFilter [▶ 1295]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` is given the same format as the input image `ipSrcImage`.

Filter size

The size of the Gauss filter is described in the X direction by `nFilterWidth` and in the Y direction by `nFilterHeight`. The size specifications must be odd numbers, as there must always be a central pixel.

Standard deviation

The standard deviation in the X direction fSigmaX and in the Y direction fSigmaY defines not only the filter size but also the shape of the Gauss curve used in the filter.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

```
hr := F_VN_GaussianFilterExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterWidth    :=  3,
    nFilterHeight   :=  3,
    fSigmaX         :=  5,
    fSigmaY         :=  5,
    eBorderType     :=  TCVN_BIM_DEFAULT,
    hrPrev          :=  hr
);
```

Samples

- Blur filter [▶ 2805]

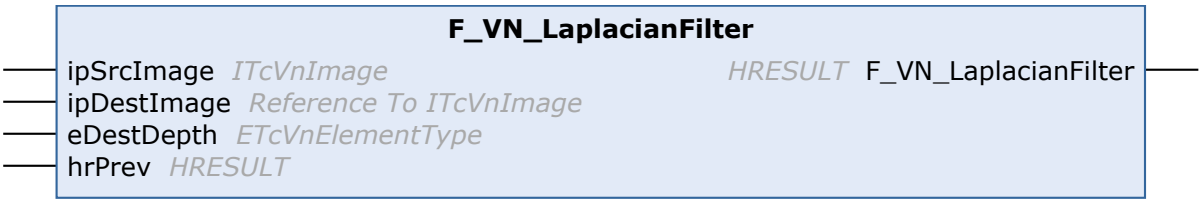
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LaplacianFilter` applies a Laplace filter to the input image.

Algorithm

The Laplace filter can be used to detect edges by approximating the second derivative of the transitions of adjacent pixel intensities. Since the derivative also has negative values, it is advisable to use a signed element type for the result image (e.g. `TCVN_ET_INT`).

For nKernelSize = 3, 5, ... the Laplace filter is calculated by adding the second derivative in x- and y-direction over the specified window size. For nKernelSize = 1, the following special 3 x 3 kernel is used:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Parameter

Input image

The input image ipSrcImage may be in any available format, except for the data types SINT and DINT, which are generally not supported.

Result image

The result image ipDestImage has the same size as the input image ipSrcImage, but the pixel format is determined by the result depth `eDestDepth`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

NOTICE

Combination of input image and result depth

Only the same or a larger data type than that of the input image can be used as the result depth, otherwise data would be lost. For example, a UINT cannot be converted to a USINT. Furthermore, there are two restrictions, so that a UINT cannot be converted into an INT and a REAL cannot be converted into an LREAL.

Expert parameters

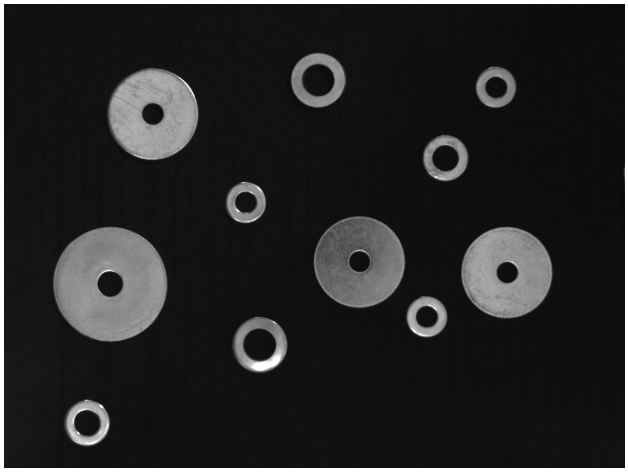
The expert version F_VN_LaplacianFilterExp [▶ 1301] contains additional parameters.

Application

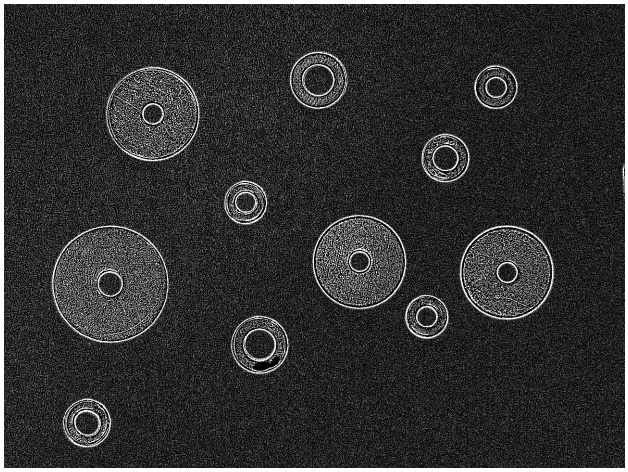
The application of a Laplace filter with a result depth of 8 bits looks like this, for example:

```
hr := F_VN_LaplacianFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    hrPrev      :=  hr
);
```

Original image



Result image



Samples

- Edge detection filters [▶ 2807]

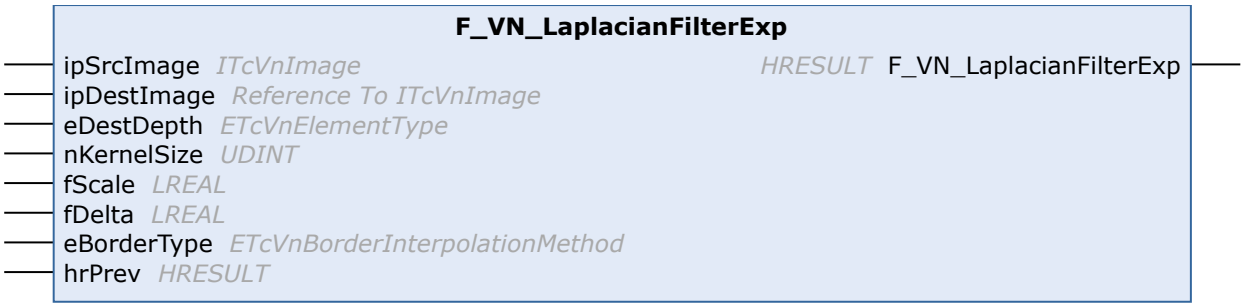
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LaplacianFilterExp` is the expert version of F_VN_LaplacianFilter [▶ 1299]. It contains additional parameters.

Parameter

Input image

The input image ipSrcImage may be in any available format, except for the data types SINT and DINT, which are generally not supported.

Result image

The result image ipDestImage has the same size as the input image ipSrcImage, but the pixel format is determined by the result depth `eDestDepth`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

NOTICE

Combination of input image and result depth

Only the same or a larger data type than that of the input image can be used as the result depth, otherwise data would be lost. For example, a UINT cannot be converted to a USINT. Furthermore, there are two restrictions, so that a UINT cannot be converted into an INT and a REAL cannot be converted into an LREAL.

Kernel size

The parameter `nKernelSize` defines the size of the filter kernel for the approximation of the 2nd derivative. Only odd numbers between `1` and `31` are allowed.

Scaling

The result of the Laplace operation is multiplied by the scaling factor `fScale`.

Delta

The constant delta `fDelta` is added to the result of the Scharr operation after scaling with `fScale`.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

The application of a Laplace filter with a kernel size of `3×3` looks like this, for example:

```
hr := F_VN_LaplacianFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    nKernelSize :=  3,
    fScale      :=  1,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr
);
```

Samples

- Edge detection filters [▶ 2807]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT or UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocalMaxima` finds local maxima in the input image.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image with a bit depth of 8 or 16 bits (USINT or UINT).

Result image

The result image `ipDestImage` is of the same type as the input image `ipSrcImage`. Each local maximum of the input image is marked here by a pixel value of 1.

Application

The finding of local maxima looks like this:

```
hr := F_VN_LocalMaxima(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    hrPrev      :=  hr
);
```

Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT or UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocalMinima` finds local minima in the input image.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image with a bit depth of 8 or 16 bits (USINT or UINT).

Result image

The result image `ipDestImage` is of the same type as the input image `ipSrcImage`. Each local minimum of the input image is marked here by a pixel value of 1.

Application

The finding of local minima looks like this:

```
hr := F_VN_LocalMinima(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    hrPrev      :=  hr
);
```

Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (for nFilterSize 3 or 5: USINT, UINT, REAL. For bigger filters, only USINT is supported.
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

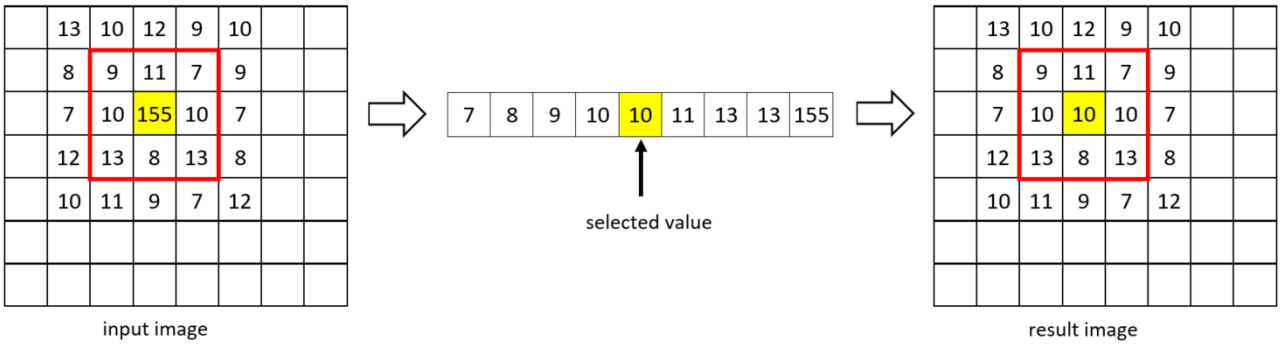
Further information

The function `F_VN_MedianFilter` applies a median filter to the input image.

Algorithm

The median filter is a ranking filter that is often used to suppress noise. All pixel values captured by the filter matrix are sorted by size, then the current value is replaced by the value at the center position of the sorted list. The advantage over a filter with a weighted mean value (e.g. a Gaussian filter) is that strong outliers of individual pixels (such as the 155 in the following sample image) are completely ignored and thus do not have a negative effect on the result.

The size of the filter matrix is specified by `nFilterSize`, only odd integers >= 3 being accepted.



Parameter

Input image

The input image `ipSrcImage` can have any number of channels. The element types `USINT`; `UINT` and `REAL` are permitted with a filter size `nFilterSize` of `3` or `5`. Only the element type `USINT` is supported for larger filters.

Result image

The result image `ipDestImage` returns the filtered image.

Filter size

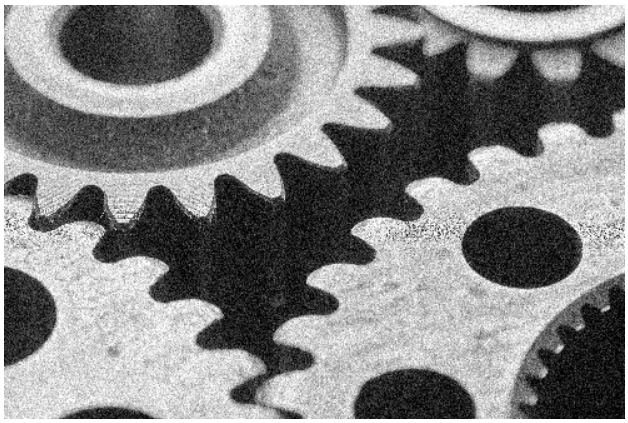
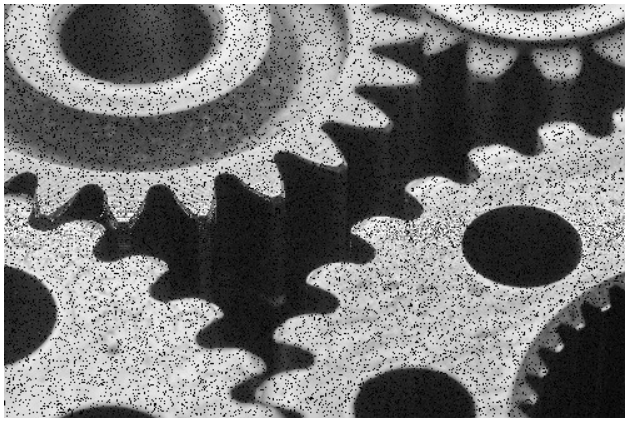
The filter size `nFilterSize` defines the height and width of the filter area. It must be an odd number and `>= 3`.

Application

The application of a median filter with the size `7×7` looks like this, for example:

```
hr := F_VN_MedianFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterSize     :=  7,
    hrPrev          :=  hr
);
```

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

Original images	Result images
	
	
	

Samples

- Blur filter [▶ 2805]

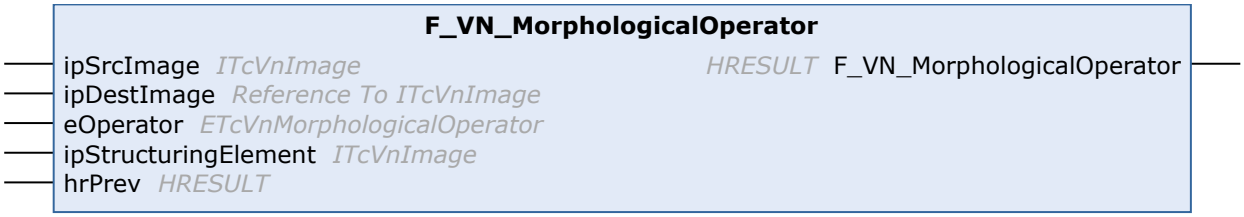
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (only 1 channel supported for reconstruction operators)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eOperator	ETcVnMorphologicalOperator [▶ 208]	Operator type
ipStructuringElement	ITcVnImage [▶ 411]	Structuring element to be applied (Typically created via F_VN_CreateStructuringElement.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MorphologicalOperator` applies a morphological operator to the input image.

Parameter

Original image

The Reconstruction operators can only be applied to a 1-channel image, the other operators have no special requirements for the original image `ipSrcImage`.

Result image

The modified image is returned in the result image `ipDestImage`.

Morphological operator

The parameter `eOperator` defines which morphological operator is applied to the original image `ipSrcImage`. All operators in the enum ETcVnMorphologicalOperator [▶ 208] are available.

Structure element

The structure element `ipStructuringElement` is the element that specifies the filter area of the morphological operator. It can be created with the help of the function F_VN_CreateStructuringElement [▶ 1285].

Application

The application of an Opening operator with a rectangular filter area of the size `3×3` looks like this, for example:

```
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
);
```

Samples

- Morphological operators [▶ 2809]

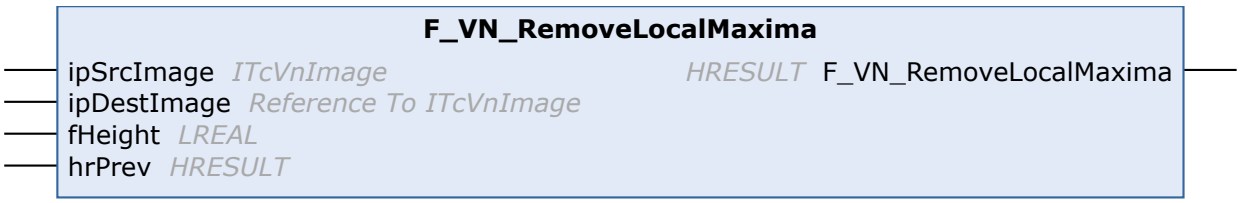
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_RemoveLocalMaxima` removes local maxima from the input image.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: `USINT`, `UINT`, `INT`, `REAL` or `LREAL`.

Result image

The result image `ipDestImage` returns the image from which the local maxima have been removed.

Maximum height

The maximum height `fHeight` defines the height up to which maxima in the input image are removed. The height describes the difference in intensity between the maximum and the value to which the pixels concerned are set. `fHeight` must be larger than `0`.

Application

The removal of local maxima up to a height of `10` looks like this, for example:

```
hr := F_VN_RemoveLocalMaxima(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fHeight         :=  10,
    hrPrev          :=  hr
);
```

Samples

- Removing local extremes

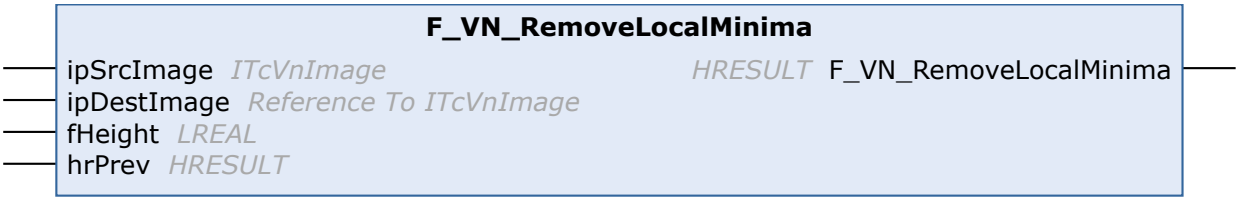
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_RemoveLocalMinima` removes local minima from the input image.

Parameter

Input image

The input image `ipSrcImage` may only have one channel and must have one of the following element types: `USINT`, `UINT`, `INT`, `REAL` or `LREAL`.

Result image

The result image `ipDestImage` returns the image from which the local minima have been removed.

Maximum height

The maximum height `fHeight` defines the height up to which minima in the input image are removed. The height describes the difference in intensity between the minimum and the value to which the pixels concerned are set. `fHeight` must be larger than `0`.

Application

The removal of local minima up to a height of `10` looks like this, for example:

```
hr := F_VN_RemoveLocalMinima(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fHeight         :=  10,
    hrPrev          :=  hr
);
```

Samples

- Removing local extremes

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 203]	Filter direction
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ScharrFilter` applies a Scharr filter to the input image. The Scharr filter functions analogously to the Sobel Filter [▶ 1321] and differs from it only by the kernel.

Algorithm

The Scharr filter is closely related to the Sobel filter. However, it provides more accurate results than a 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

Original image

The original image `ipSrcImage` can have any format.

Result image

The result image `ipDestImage` returns the filter result and has the same format as the original image `ipSrcImage`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

Filter direction

The filter direction `eFilterDirection` defines whether the Scharr filter is applied to the image in the X-direction (`TCVN_FD_X`) or in the Y-direction (`TCVN_FD_Y`).

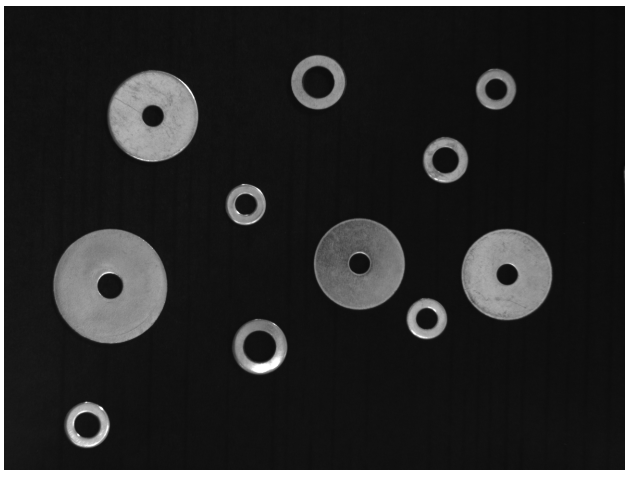
Expert parameters

Further parameters can be found in the expert version F_VN_ScharrFilterExp [▶ 1315].

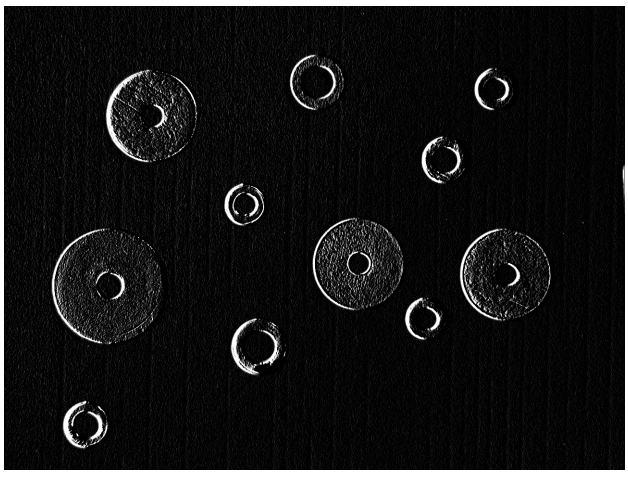
Application

```
hr := F_VN_ScharrFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    eDestDepth      :=  TCVN_ET_USINT,
    eFilterDirection:=  TCVN_FD_X,
    hrPrev          :=  hr,
);
```

Original image



Result image



Samples

- Edge detection filters [▶ 2807]

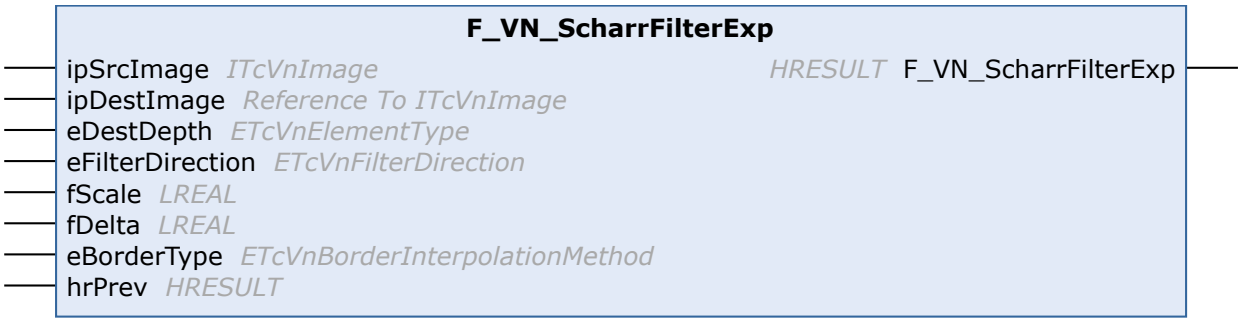
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 203]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ScharrFilterExp` is the expert version of F_VN_ScharrFilter [▶ 1313]. It contains additional parameters.

Parameter

Original image

The original image `ipSrcImage` can have any format.

Result image

The result image `ipDestImage` returns the filter result and has the same format as the original image `ipSrcImage`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

Filter direction

The filter direction `eFilterDirection` defines whether the Scharr filter is applied to the image in the X-direction (`TCVN_FD_X`) or in the Y-direction (`TCVN_FD_Y`).

Scaling

The result of the Scharr operation is multiplied by the scaling factor `fScale`.

Delta

The constant delta `fDelta` is added to the result of the Scharr operation after scaling with `fScale`.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

```
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,
);
```

Samples

- Edge detection filters [▶ 2807]

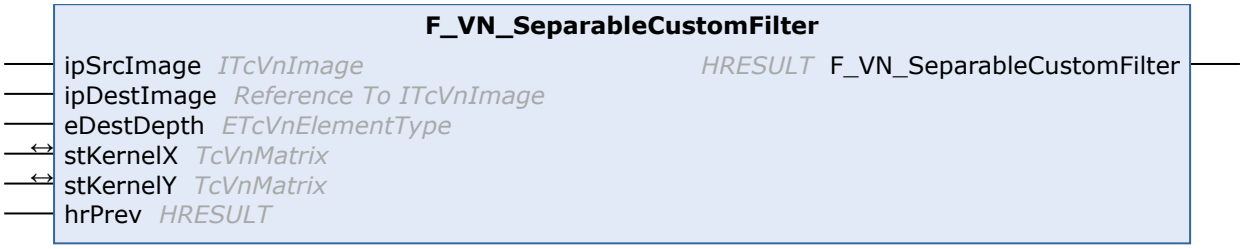
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernelX	TcVnMatrix [▶ 231]	1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 231]	1D custom column-filter kernel with values of type REAL or LREAL

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_SeparableCustomFilter` applies a separable 2D filter to the input image.

Algorithm

A separable 2D filter can be combined from two 1D filters. It is more efficient to apply a 1D filter to an image twice than to apply a 2D filter once. For example, the well-known Sobel filter [▶ 1321] can be constructed as follows from two separate one-dimensional filters:

$$stKernelX = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad stKernelY = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` has the same size as the input image `ipSrcImage`, but the pixel format is determined by the result depth `eDestDepth`.

X and Y kernel

The two 1-dimensional filter kernels `stKernelX` and `stKernelY` are transferred as a matrix. Permitted element types of the matrix are `TCVN_ET_REAL` and `TCVN_ET_LREAL`. Corresponding matrices of the type TcVnMatrix [▶ 231] can be created with the aid of the function F_VN_InitMatrixStruct [▶ 1539].

Expert parameters

The expert version F_VN_SeparableCustomFilterExp [▶ 1319] contains additional parameters.

Application

The creation and application of a separable 2D filter looks like this, for example:

```
hr := F_VN_SeparableCustomFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernelX   :=  stKernelMatrixX,
    stKernelY   :=  stKernelMatrixY,
    hrPrev      :=  hr
);
```

Samples

- User-defined filters

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	Destination image depth
fDelta	LREAL	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolation Method [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernelX	TcVnMatrix [▶ 231]	1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 231]	1D custom column-filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 160]	Anchor point of the kernel

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_SeparableCustomFilterExp` is the Expert version of F_VN_SeparableCustomFilter [▶ 1317]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` may have any available format.

Result image

The result image `ipDestImage` has the same size as the input image `ipSrcImage`, but the pixel format is determined by the result depth `eDestDepth`.

X and Y kernel

The two 1-dimensional filter kernels `stKernelX` and `stKernelY` are transferred as a matrix. Permitted element types of the matrix are `TCVN_ET_REAL` and `TCVN_ET_LREAL`. Corresponding matrices of the type TcVnMatrix [▶ 231] can be created with the aid of the function F_VN_InitMatrixStruct [▶ 1539].

Anchor point

The anchor point `aAnchor` defines which point of the filter kernel is moved over each pixel of the input image `ipSrcImage`.

Delta

The constant delta `fDelta` is added to the result of the filter operation.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

```
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
);
```

Samples

- User-defined filters

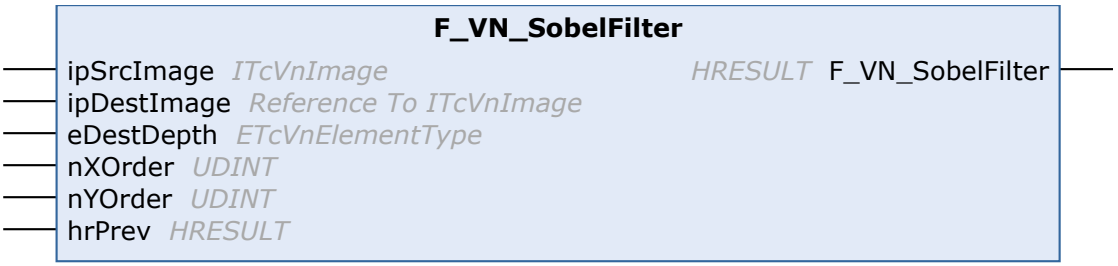
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_SobelFilter` applies a Sobel filter to the input image. The Sobel filter is used for edge detection. Edges in the input image are highlighted as bright lines on a dark background in the result image.

Algorithm

The extended Sobel filter can be used to detect edges by approximating the nth derivative of the transitions of adjacent pixel intensities. Since the derivative also has negative values, it is advisable to use a signed element type for the result image (e.g. `TCVN_ET_INT`).

The kernel of the filter is determined by the parameters of the function. Typically the function is called with nXOrder = 1, nYOrder = 0, nKernelSize = 3 or nXOrder = 0, nYOrder = 1, nKernelSize = 3. These parameters lead to the kernels G_x and G_y.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

For these kernels, the filter combines Gaussian smoothing with the calculation of the first derivative, so that the result is quite noise resistant. G_x emphasizes vertical edges, G_y emphasizes horizontal edges.

Parameter

Original image

The original image `ipSrcImage` can have any format.

Result image

The result image `ipDestImage` returns the filter result and has the same format as the original image `ipSrcImage`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

Derivative order

The derivative orders `nXOrder` and `nYOrder` define which derivative in the X and Y direction is to be used to create the filter.

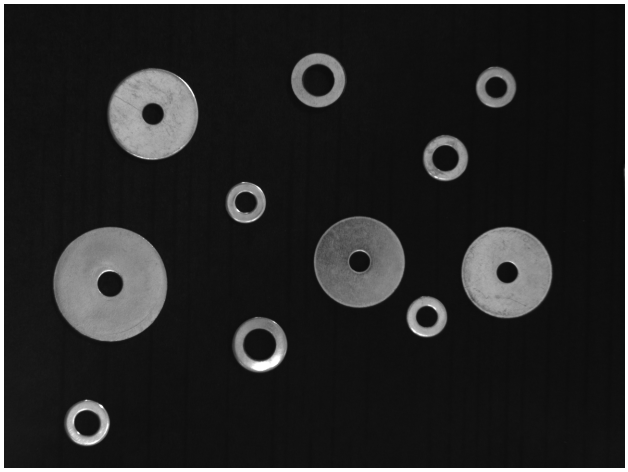
Expert parameters

Further parameters can be found in the expert version F_VN_SobelFilterExp [▶ 1323].

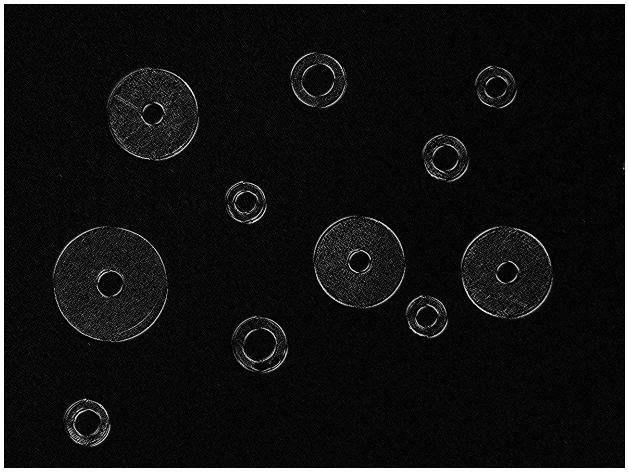
Application

```
hr := F_VN_SobelFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    nXOrder     :=  1,
    nYOrder     :=  1,
    hrPrev      :=  hr,
);
```

Original image



Result image



Samples

- Edge detection filters [▶ 2807]

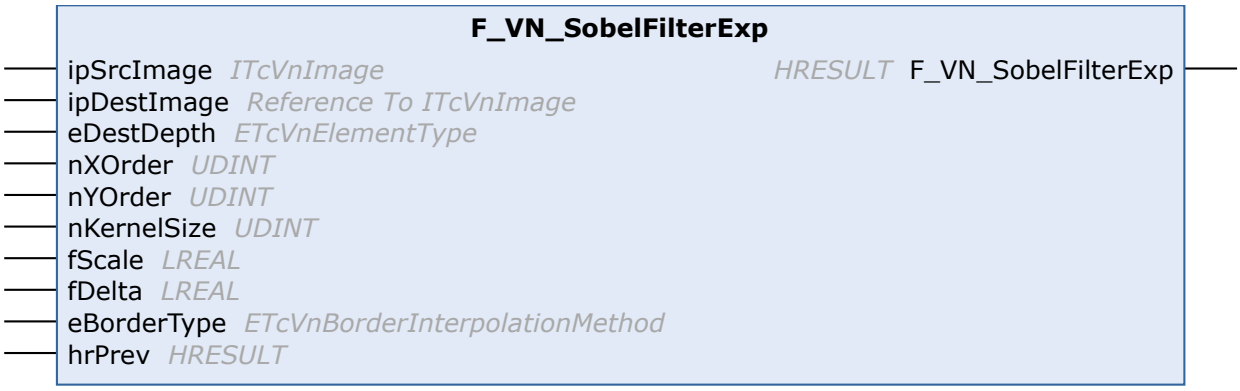
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 199]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_SobelFilterExp` is the expert version of F_VN_SobelFilter [▶ 1321]. It contains additional parameters.

Parameter

Original image

The original image `ipSrcImage` can have any format.

Result image

The result image `ipDestImage` returns the filter result and has the same format as the original image `ipSrcImage`.

Result depth

The result depth `eDestDepth` defines which element type the result image `ipDestImage` has. A larger element type can display more information.

Derivative order

The derivative orders `nXOrder` and `nYOrder` define which derivative in the X and Y direction is to be used to create the filter.

Kernel size

The size nKernelSize of the Sobel kernel determines how large the area is that influences the result value of a pixel.

Scaling

The result of the Sobel operation is multiplied by the scaling factor `fScale`.

Delta

The constant delta `fDelta` is added to the result of the Sobel operation after scaling with `fScale`.

Border extrapolation

The method of border extrapolation `eBorderType` defines how non-existent pixels are extrapolated beyond the image border in order to calculate the filter values at the image borders. For further details, see ETcVnBorderInterpolationMethod [▶ 166].

Application

```
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,
);
```

Samples

- Edge detection filters [▶ 2807]

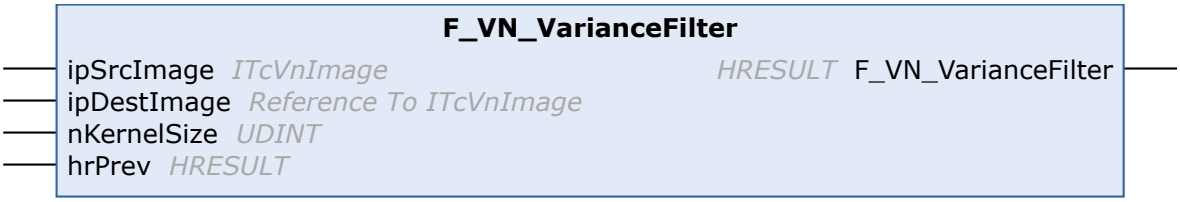
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image of type TCVN_ET_UINT (An appropriate destination image will be created if required.)
nKernelSize	UDINT	Size of the kernel
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

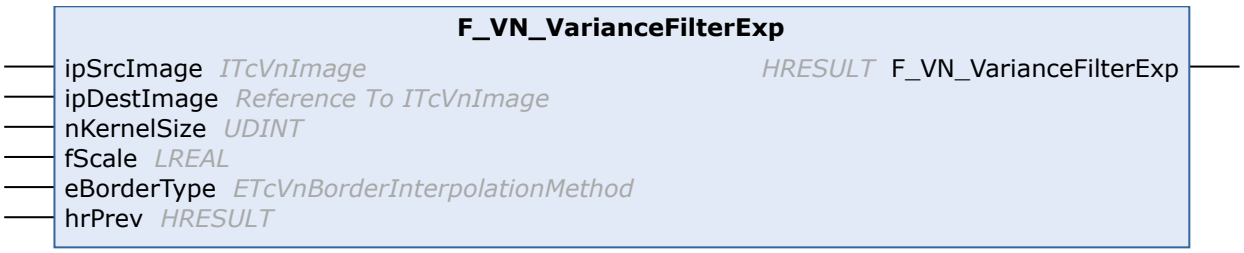
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 166]	Image border handling
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17 Image Segmentation

This group contains functions for segmenting images.

Functions

Segmentation

- F_VN_AdaptiveThreshold(Exp) [▶ 1329]
- F_VN_CheckColorRange [▶ 1331]
- F_VN_DoubleThreshold [▶ 1332]
- F_VN_Threshold [▶ 1334]
- F_VN_WatershedSegmentationExp [▶ 1336]
- F_VN_SauvolaThreshold [▶ 1337]

Samples

Image segmentation [▶ 2810]

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 [▶ 411]	Source image (1 channel, elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_AdaptiveThreshold` applies an adaptive threshold to the input image.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image with a bit depth of 8 bits (`USINT`).

Result image

The binary result image `ipDestImage` has the same format as the input image. The pixels have either the value `0` or the target value `fMaxValue`.

Target value

The target value `fMaxValue` defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to `(2^d)-1`, where d represents the bit depth of the element type; e.g. `(2^8)-1=255` with 8-bit images.

Expert parameters

The expert version F_VN_AdaptiveThresholdExp [▶ 1329] contains additional parameters.

Application

```
hr := F_VN_AdaptiveThreshold(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fMaxValue       :=  255, // max value of 8-bit image: (2^8)-1=255
    hrPrev          :=  hr
);
```

Related functions

- F_VN_DoubleThreshold [▶ 1332] for binary segmentation on the basis of double threshold values using morphological reconstruction
- F_VN_Threshold [▶ 1334] for binary segmentation on the basis of a threshold value
- F_VN_AdaptiveThreshold [▶ 1327] for binary, adaptive segmentation
- F_VN_CheckColorRange [▶ 1331] for binary segmentation on the basis of a color spectrum
- F_VN_ReferenceColorSimilarity [▶ 1254] for binary segmentation on the basis of a color model

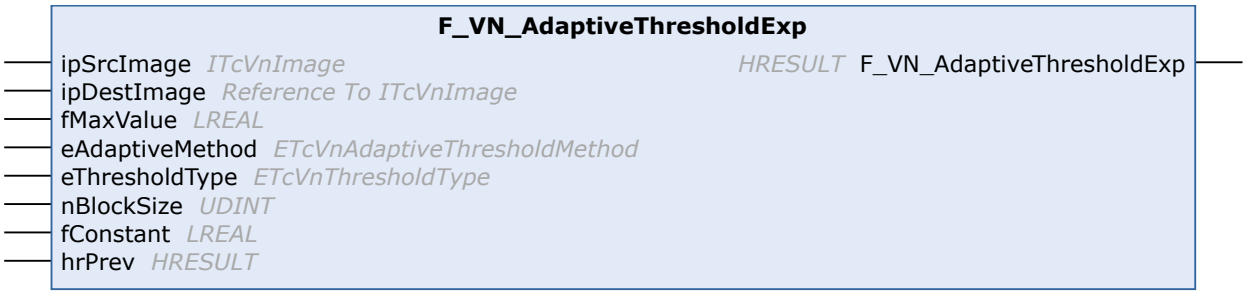
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (1 channel, elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 163]	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 [▶ 220]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_AdaptiveThresholdExp` is the expert version of F_VN_AdaptiveThreshold [▶ 1327]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image with a bit depth of 8 bits (`USINT`).

Result image

The binary result image `ipDestImage` has the same format as the input image. The pixels have either the value `0` or the target value `fMaxValue`.

Target value

The target value `fMaxValue` defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to `(2^d)-1`, where d represents the bit depth of the element type; e.g. `(2^8)-1=255` with 8-bit images.

Threshold method

The threshold method eAdaptiveMethod defines whether the adaptive threshold is calculated as an average or as a Gaussian weighted sum. The two values `TCVN_ATM_MEAN` and `TCVN_ATM_GAUSSIAN` are available for this in the enum ETcVnAdaptiveThresholdMethod [▶ 163].

Threshold inversion

The parameter eThresholdType specifies whether or not the result image should be inverted. Therefore, only the two values `TCVN_TT_BINARY` and `TCVN_TT_BINARY_INV` from the enum ETcVnThresholdType [▶ 220] are supported at this point.

Threshold neighborhood size

The parameter `nBlockSize` defines the size of the neighborhood that is considered for the calculation of the local threshold. The parameter must be an odd number (3, 5, 7, etc.).

Threshold adjustment

The constant value `fConstant` is subtracted from each local average value. The result is the local threshold that is used for the calculation of the result image.

Application

```
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
);
```

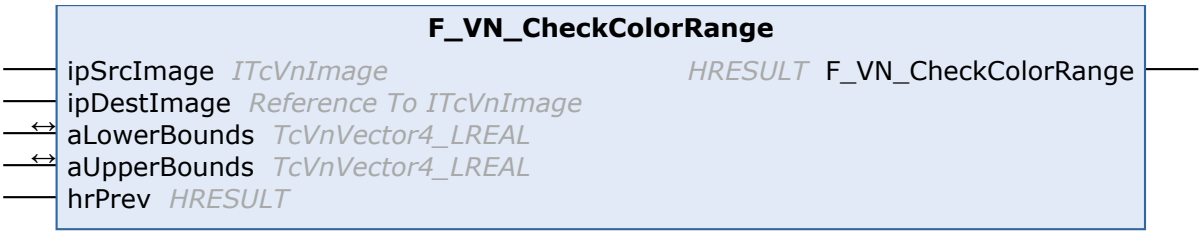
Required License

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 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLowerBounds	TcVnVector4_LREAL [▶ 162]	Channel-wise lower bounds (Unused channels are ignored.)
aUpperBounds	TcVnVector4_LREAL [▶ 162]	Channel-wise upper bounds (Unused channels are ignored.)

Return value

HRESULT [▶ 141]

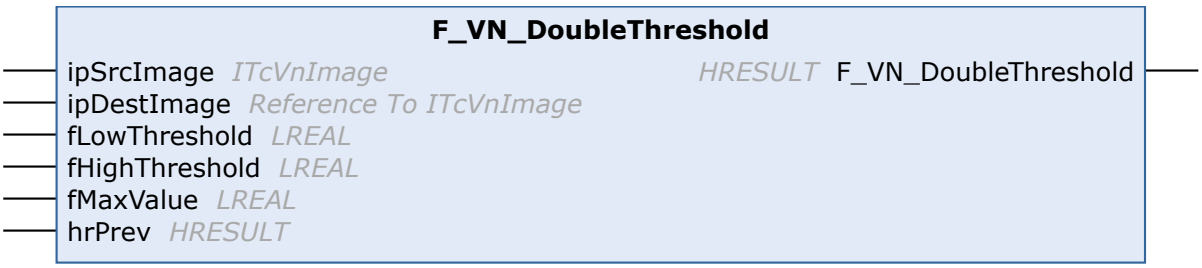
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_DoubleThreshold` calculates a double threshold value using morphological reconstruction. The function expects the searched segments to be brighter than the background and, compared to the function `F_VN_Threshold`, enables additional filtering of the determined image segments.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Result image

The binary result image `ipDestImage` has the same format as the input image. The pixels have either the value `0` or the target value `fMaxValue`.

Lower threshold value

The lower threshold value determines the shape of the segments found, such as the function `F_VN_Threshold`.

Upper threshold value

The upper threshold value is used to mark image segments whose shape is then calculated on the basis of the lower threshold value. In this way, unwanted segments whose grayscale values are greater than the lower threshold value and less than the upper threshold value can be filtered out.

Target value

The target value `fMaxValue` defines the value to be assigned to the pixels that fulfill the respective threshold value criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to `(2^d)-1`, where d is the bit depth of the element type; e.g. `(2^8)-1=255` for 8-bit images.

Application

```
hr := F_VN_DoubleThreshold(
    ipSrcImage      := ipSrcImage,
    ipDestImage     := ipDestImage,
    fLowThreshold   := fLowThreshold,
    fHighThreshold  := fHighThreshold,
    fMaxValue       := 255,
    hrPrev          := hr);
```

Related functions

- F_VN_DoubleThreshold [▶ 1332] for binary segmentation on the basis of double threshold values using morphological reconstruction
- F_VN_Threshold [▶ 1334] for binary segmentation on the basis of a threshold value
- F_VN_AdaptiveThreshold [▶ 1327] for binary, adaptive segmentation
- F_VN_CheckColorRange [▶ 1331] for binary segmentation on the basis of a color spectrum
- F_VN_ReferenceColorSimilarity [▶ 1254] for binary segmentation on the basis of a color model

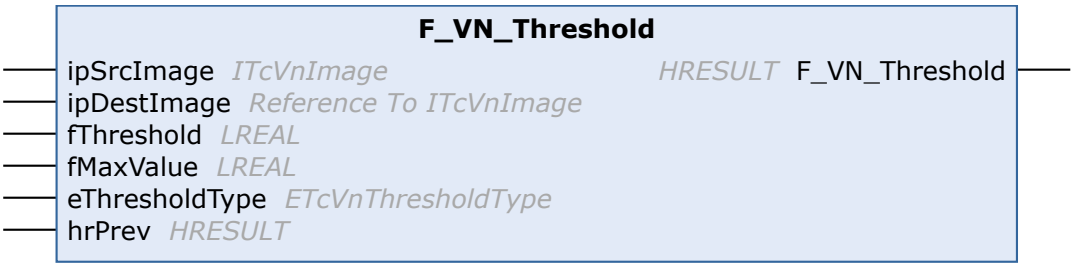
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (For Otsu and Triangle threshold types, only 1 channel of TCVN_ET_USINT is supported)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 220]	Threshold type to be applied
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_Threshold` segments the input image with regard to a threshold. Depending on this threshold and the algorithm used, each pixel of the binary result image is assigned the value `0` and a specified target value.

Algorithm

A threshold transforms a monochrome image into a binary image and thus creates segments. Good segmentation is a prerequisite for many subsequent methods such as contour search. TwinCAT Vision offers different types of threshold values that are suitable for different applications.

Fixed threshold value

A fixed threshold simply creates a binary image by setting pixels with a value greater than fThreshold to white (255) and others to black (0). fThreshold is a parameter of the corresponding functions. To find a good value, you will most probably approach it via different attempts.

Dynamic threshold value

A dynamic threshold according to Otsu works perfectly for bimodal images. The histogram of a bimodal image has two main peaks and a good threshold value is most likely exactly in the middle. If you have a noisy image, you can use a Gaussian filter to shift it towards bimodal.

Adaptive threshold value

An adaptive threshold value uses variable threshold values for the whole image. See F_VN_AdaptiveThresholdExp [▶ 1329] .

Parameter

Input image

The input image `ipSrcImage` may have one or three channels.

Result image

The result image `ipDestImage` returns the segmented binary image.

Threshold value

The threshold `fThreshold` defines the segmentation condition. It must be adjusted depending on the properties of the input image.

Target value

The target value `fMaxValue` defines which value should be given to the pixels that fulfil the threshold criterion. If the result image is to consist only of black and white pixels, then the maximum value of the element type of the input image must be set here. This corresponds to `(2^d)-1`, where `d` represents the bit depth of the element type; e.g. `(2^8)-1=255` with 8-bit images.

Threshold type

The threshold type `eThresholdType` defines the algorithm according to which the threshold `fThreshold` is applied to the input image. All values of the enum ETcVnThresholdType [▶ 220] are supported.

Application

The binary segmentation of an 8-bit grayscale image in the center of the value range looks like this, for example:

```
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
);
```

Related functions

- F_VN_DoubleThreshold [▶ 1332] for binary segmentation on the basis of double threshold values using morphological reconstruction
- F_VN_Threshold [▶ 1334] for binary segmentation on the basis of a threshold value
- F_VN_AdaptiveThreshold [▶ 1327] for binary, adaptive segmentation
- F_VN_CheckColorRange [▶ 1331] for binary segmentation on the basis of a color spectrum
- F_VN_ReferenceColorSimilarity [▶ 1254] for binary segmentation on the basis of a color model

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (DINT, 1 channel. An appropriate image will be created if required.)
ipMarkers	ITcVnImage [▶ 411]	Marker image (DINT, 1 channel)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
pipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 220]	Threshold type to be applied. Supported are TCVN_TT_BINARY and TCVN_TT_BINARY_INV.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18 Keypoint Features

This group contains functions for finding and matching distinctive features in an image.

NOTICE

Long computing duration

Many functions for finding keypoint features require a long computing duration. Therefore, pay attention to the set cycle time and if necessary use a watchdog [▶ 146] to secure the real-time behavior.

Functions

Key Points

- F_VN_KeyPointsAGAST(Exp) [▶ 1357]
- F_VN_KeyPointsFAST(Exp) [▶ 1366]
- F_VN_KeyPointsGFTT(Exp) [▶ 1368]
- F_VN_KeyPointsMSER(Exp) [▶ 1370]
- F_VN_KeyPointsSB(Exp) [▶ 1372]

Key points and descriptors

- F_VN_KeyPointsAndDescriptorsAKAZE(Exp) [▶ 1359]

- F_VN_KeyPointsAndDescriptorsBRISK(Exp) [▶ 1360]
- F_VN_KeyPointsAndDescriptorsaKAZE(Exp) [▶ 1362]
- F_VN_KeyPointsAndDescriptorsORB(Exp) [▶ 1364]

Descriptor matching

- F_VN_FilterGoodMatches [▶ 1339]
- F_VN_GetMatchCoordinates [▶ 1356]
- F_VN_MatchDescriptorsBF(Exp) [▶ 1374]
- F_VN_MatchDescriptorsFlannLsh(Exp) [▶ 1376]
- F_VN_MatchDescriptorsKnnBF(Exp) [▶ 1378]
- F_VN_MatchDescriptorsKnnFlannLsh(Exp) [▶ 1380]

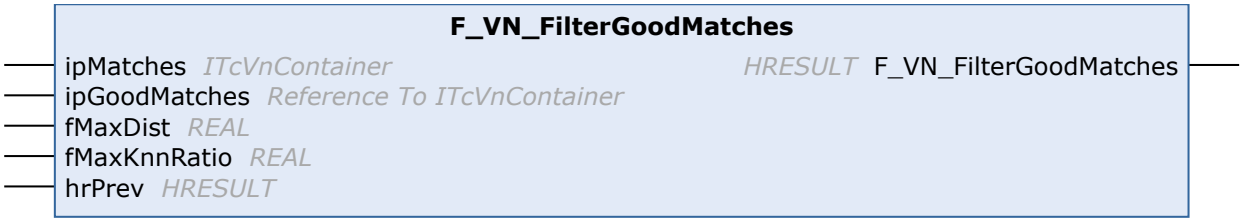
Key Points Referenz Matching

- F_VN_FindReferenceKeyPointsInImage(Exp) [▶ 1340]
- F_VN_FindReferenceKeyPointsInImageAKAZE(Exp) [▶ 1342]
- F_VN_FindReferenceKeyPointsInImageBRISK(Exp) [▶ 1346]
- F_VN_FindReferenceKeyPointsInImageORB(Exp) [▶ 1352]

Miscellaneous

- F_VN_RegionsMSER(Exp) [▶ 1382]

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 [▶ 370]	Container with descriptor matches (ContainerType_Vector_TcVnDMatch or ContainerType_Vector_Vector_TcVnDMatch)
ipGoodMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

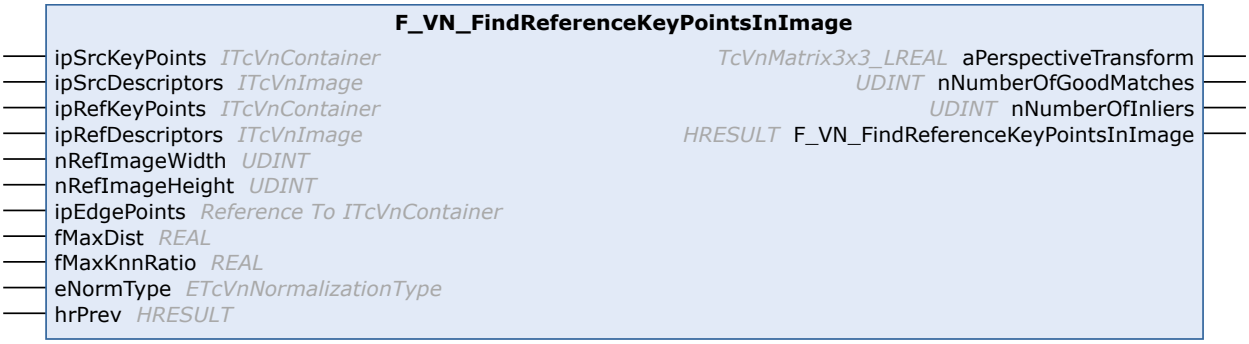
Required License

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 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 [▶ 370]	KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage [▶ 411]	Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 210]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

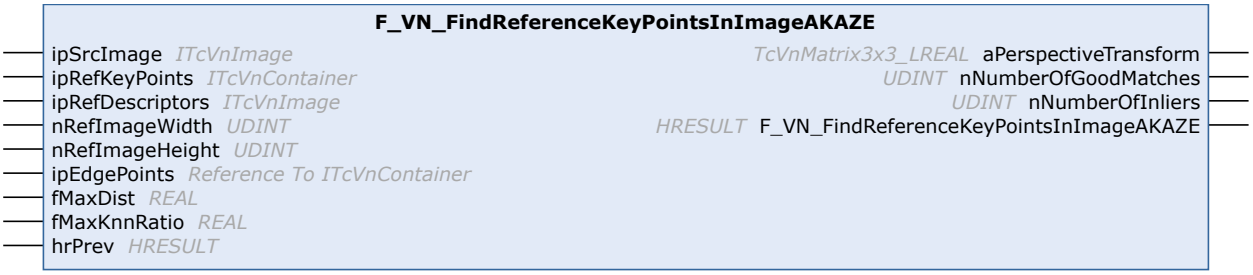
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

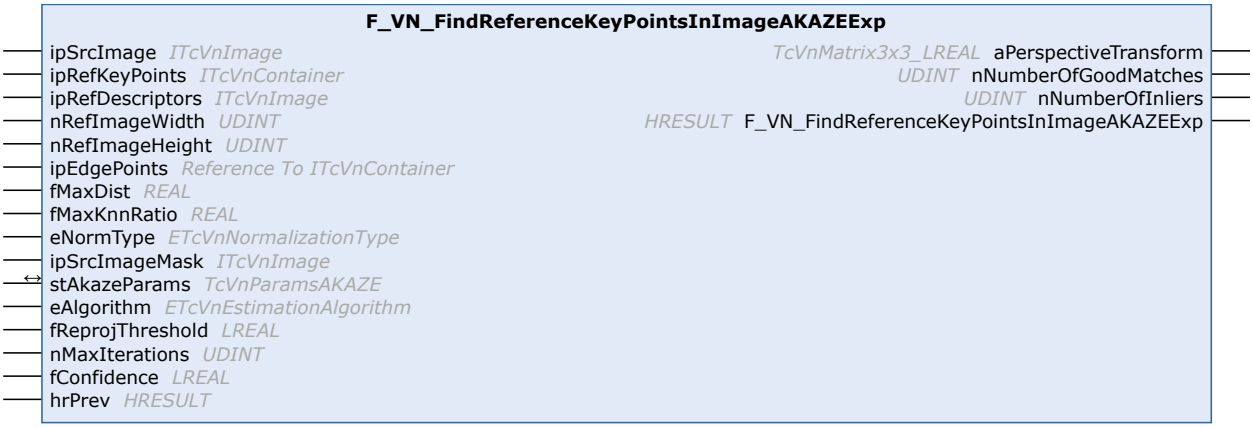
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 210]	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 200]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stAkazeParams	TcVnParamsAKAZE [▶ 234]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

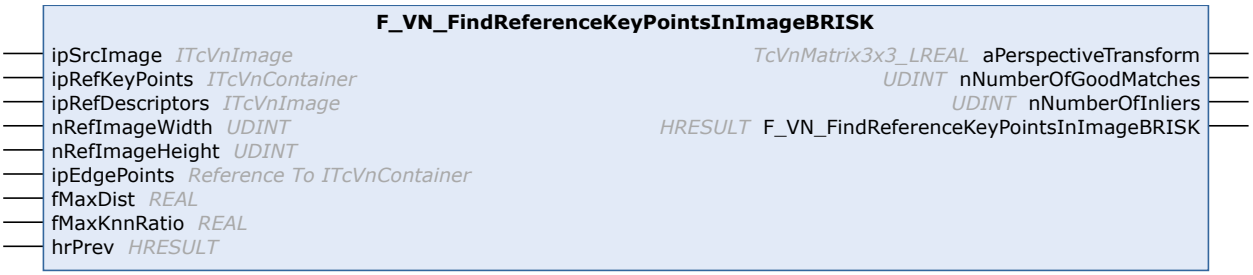
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

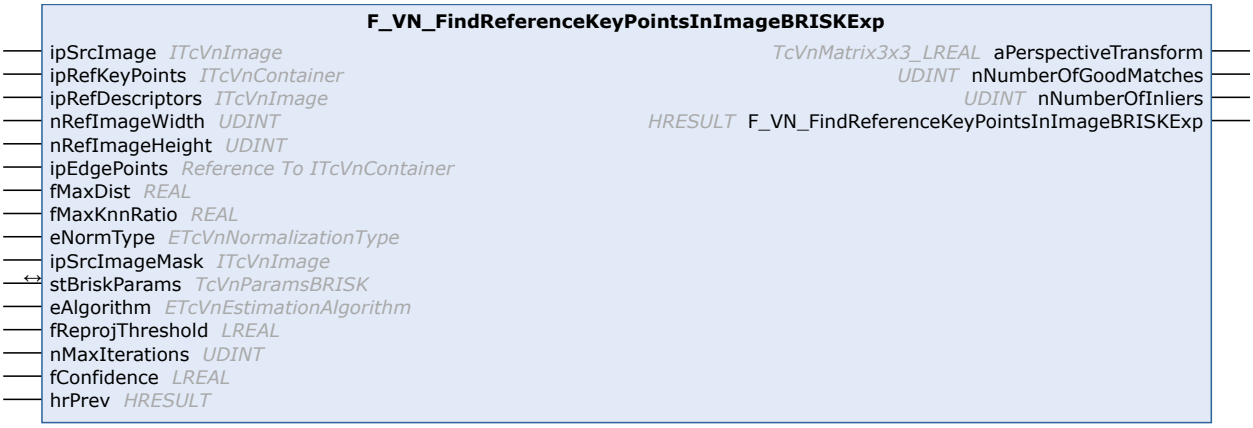
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 210]	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 200]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBriskParams	TcVnParamsBRISK [▶ 238]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

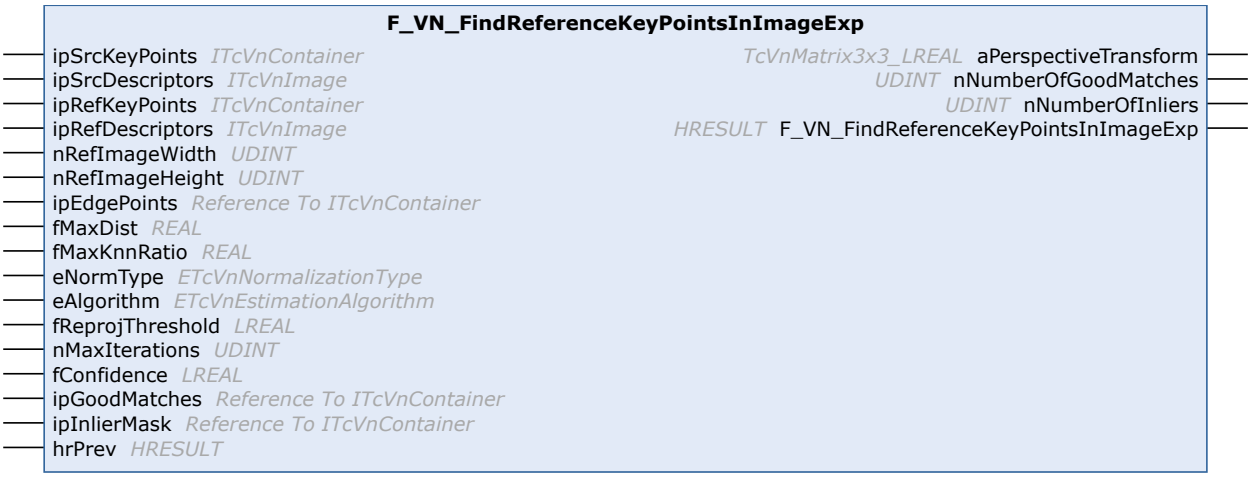
Required License

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 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 [▶ 370]	KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage [▶ 411]	Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 210]	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 [▶ 200]	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 [▶ 370]	Returns a container with good matches (optional, set to 0 if not required; ContainerType_Vector_TcVnDMatch)
ipInlierMask	Reference To ITcVnContainer [▶ 370]	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

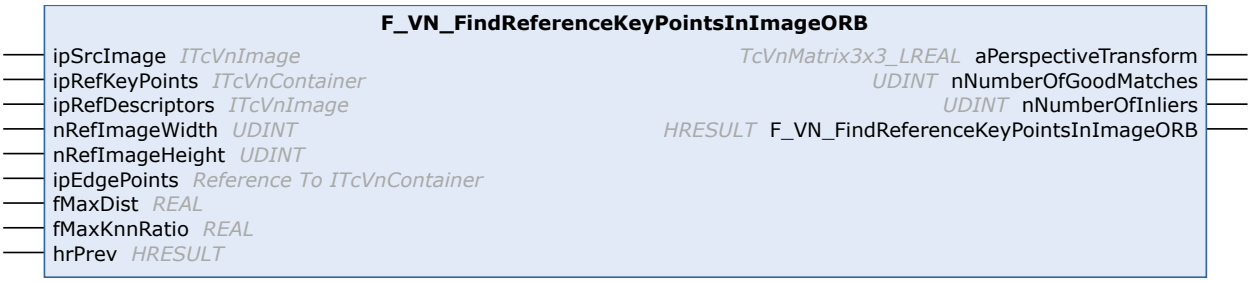
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

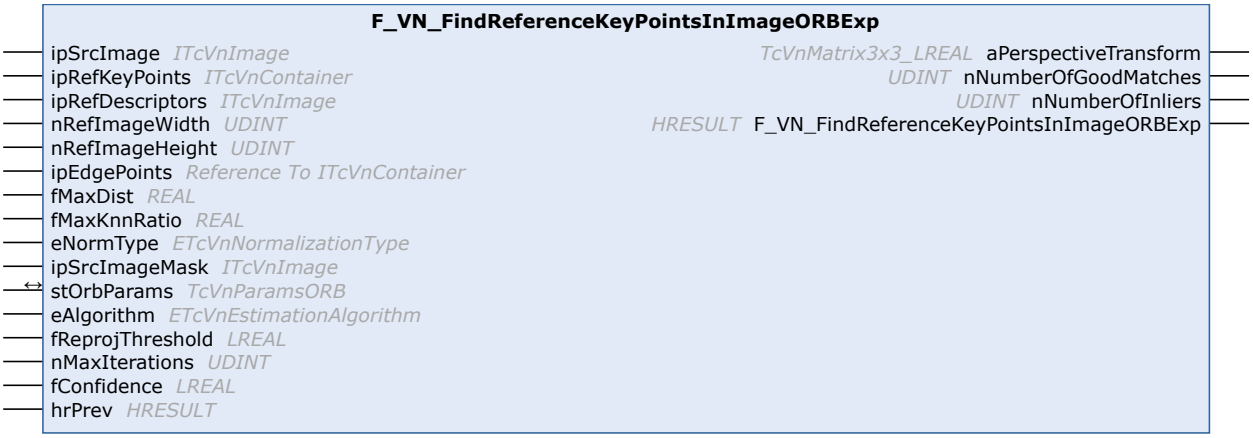
Required License

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 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 [▶ 411]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 370]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 411]	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 [▶ 370]	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 [▶ 210]	Normalization type used for descriptor matching (HAMMING recommended for ORB, HAMMING2 if the ORB nBriefPoints parameter is 3 or 4)
ipSrcImageMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 200]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrbParams	TcVnParamsORB [▶ 241]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 162]	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 [▶ 141]

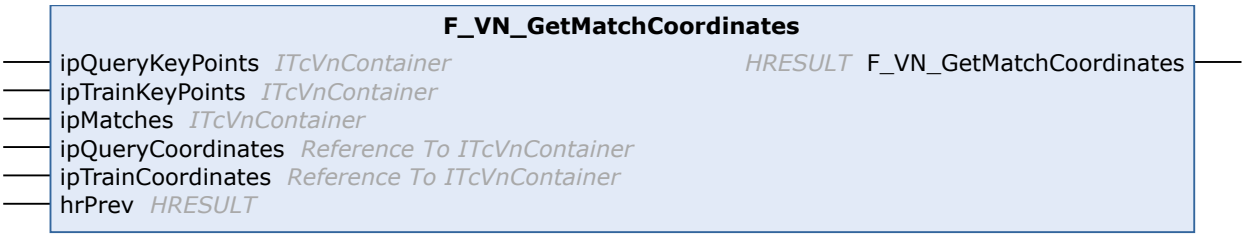
Required License

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 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 [▶ 370]	Container with query Keypoints (i.e. keypoints of a reference/ template image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipTrainKeyPoints	ITcVnContainer [▶ 370]	Container with training Keypoints (i.e. keypoints of a source/ input image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipMatches	ITcVnContainer [▶ 370]	Container with matches between query and train Keypoints (CTcVnContainer_Vector_TcVnDMatch).
ipQueryCoordinates	Reference To ITcVnContainer [▶ 370]	Returns a container with the coordinates of keypoints from the query Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
ipTrainCoordinates	Reference To ITcVnContainer [▶ 370]	Returns a container with the coordinates of keypoints from the training Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

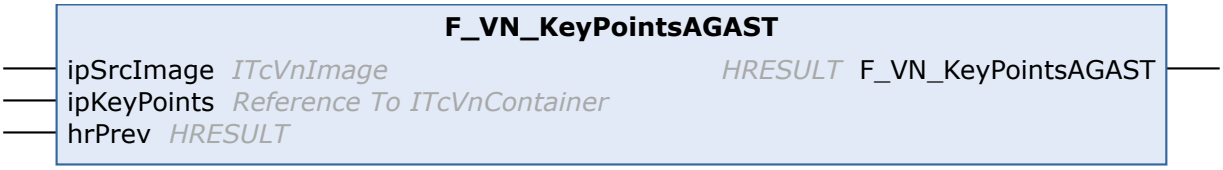
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

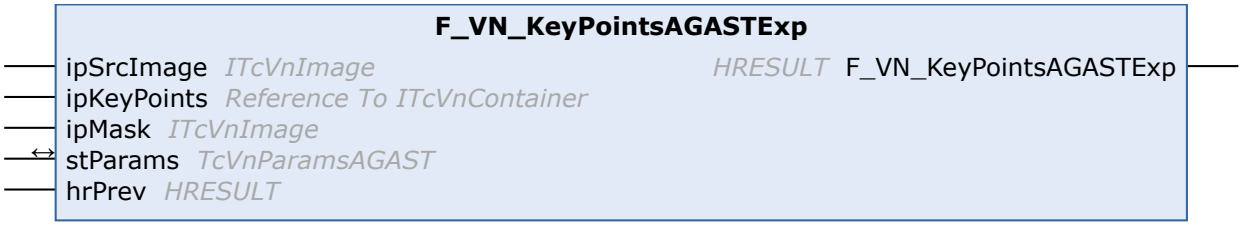
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsAGAST [▶ 233]	Additional expert parameters

Return value

HRESULT [▶ 141]

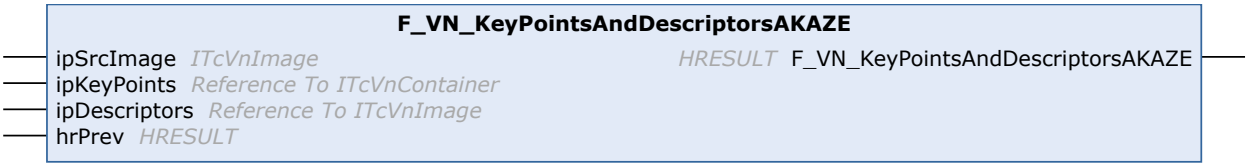
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

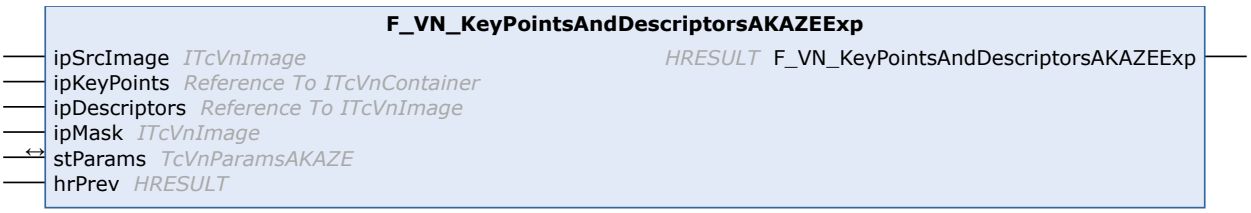
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsAKAZE [▶ 234]	Additional expert parameters

Return value

HRESULT [▶ 141]

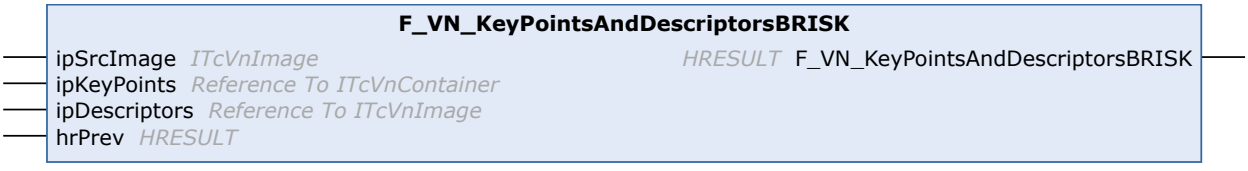
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBRISK [▶ 238]	Additional expert parameters

Return value

HRESULT [▶ 141]

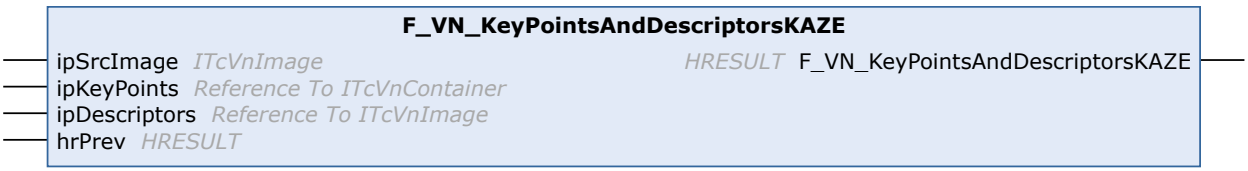
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

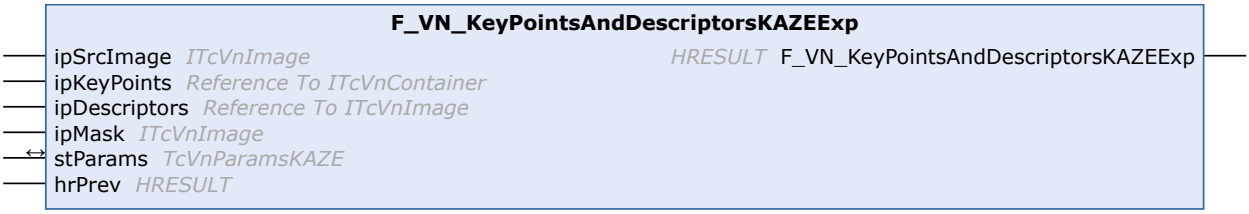
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsKAZE [▶ 240]	Additional expert parameters

Return value

HRESULT [▶ 141]

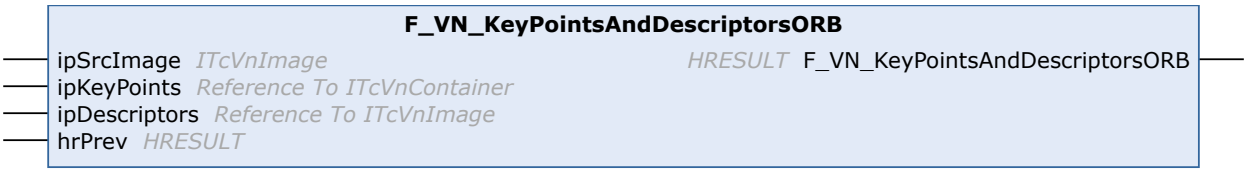
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 411]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsORB [▶ 241]	Additional expert parameters

Return value

HRESULT [▶ 141]

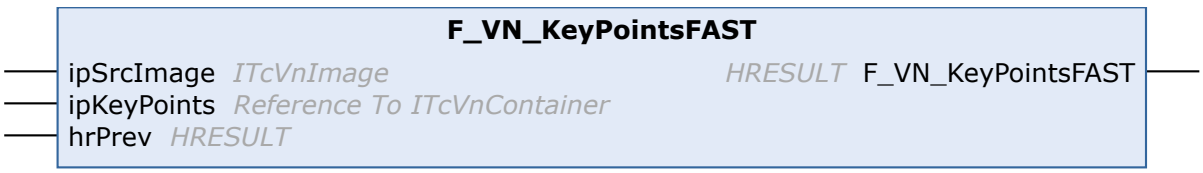
Required License

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 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 [▶ 411]	Source image (single-channel).
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

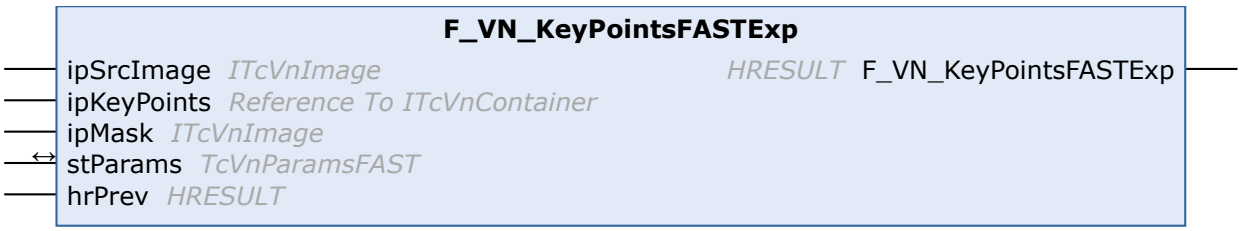
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsFAST [▶ 239]	Additional expert parameters

Return value

HRESULT [▶ 141]

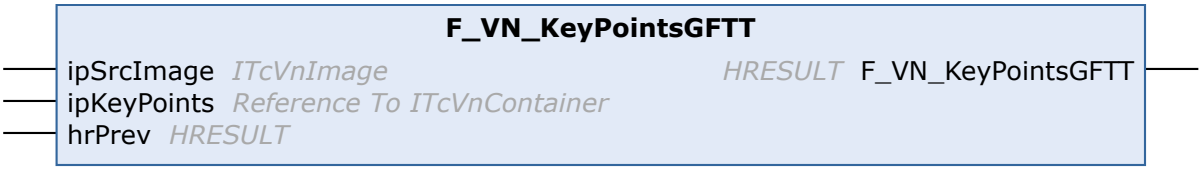
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

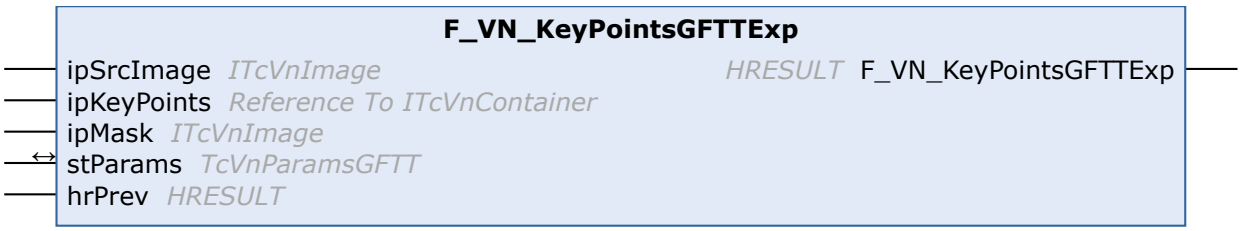
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsGFTT [▶ 239]	Additional expert parameters

Return value

HRESULT [▶ 141]

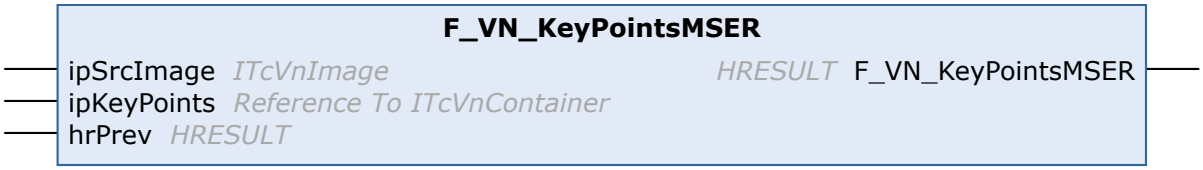
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

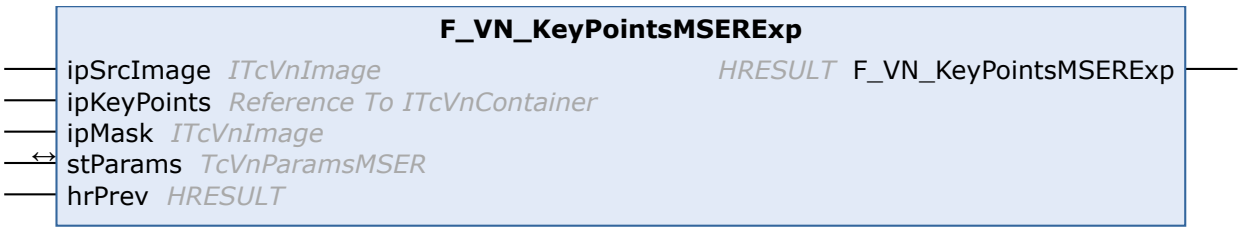
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsMSER [▶ 240]	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

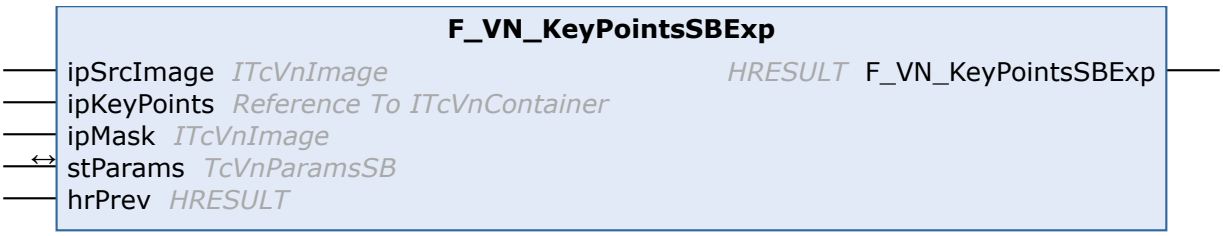
Required License

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 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 [▶ 411]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	Mask to specify, where to look for keypoints (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsSB [▶ 242]	Several parameters to filter the detected blobs

Return value

HRESULT [▶ 141]

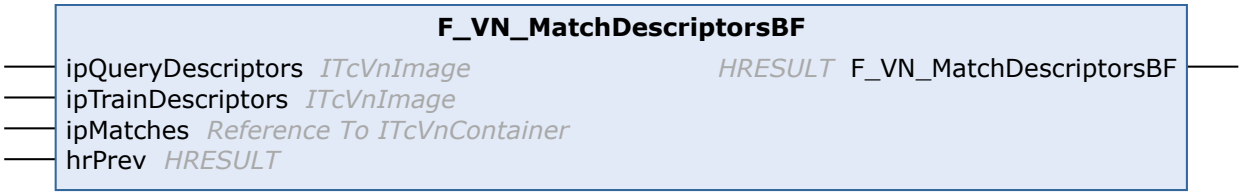
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

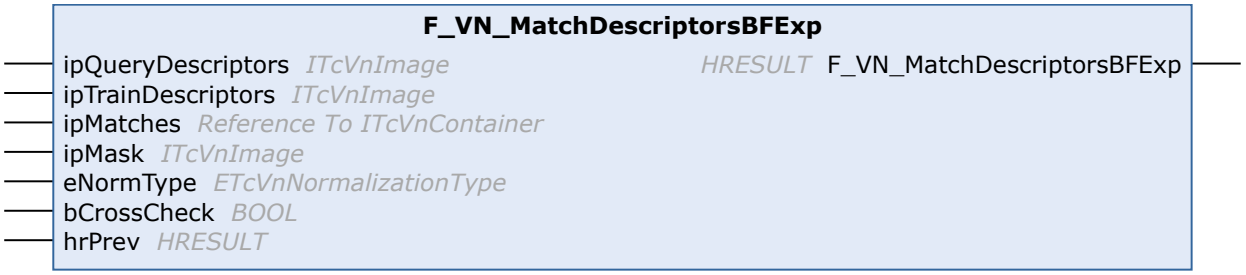
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 411]	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 [▶ 210]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

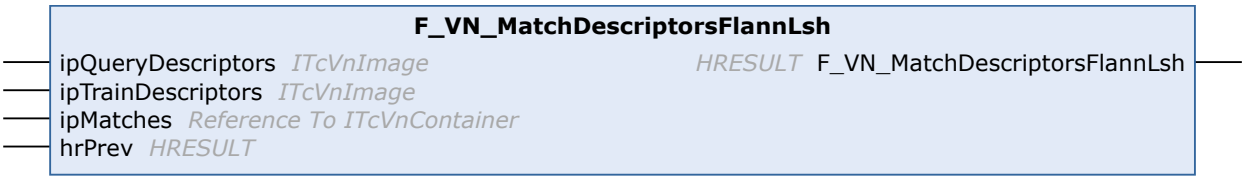
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

Number of matches

For reasons related to the method, the number of matches can vary from call to call with this function.

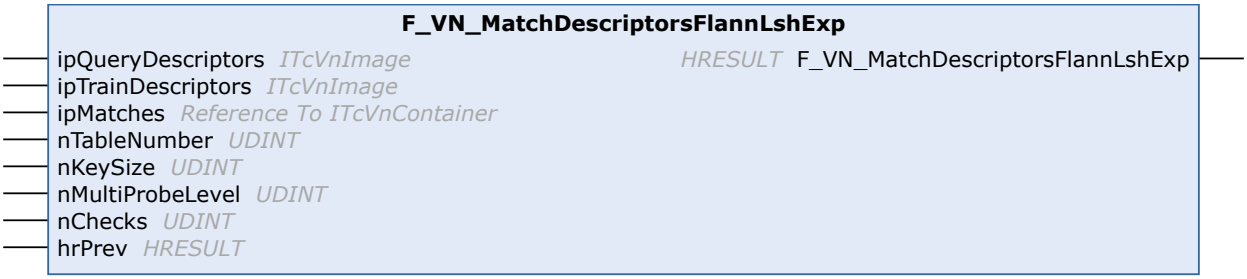
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

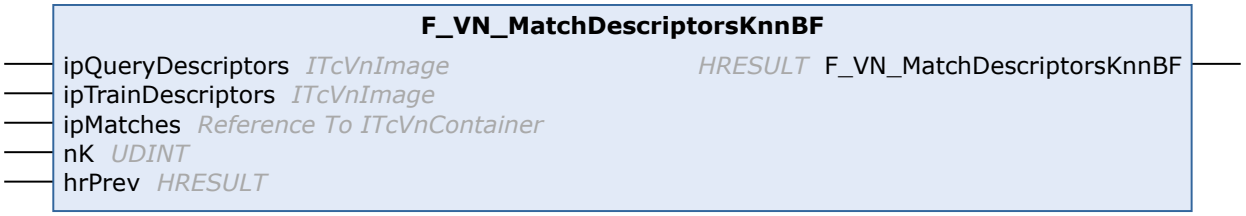
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

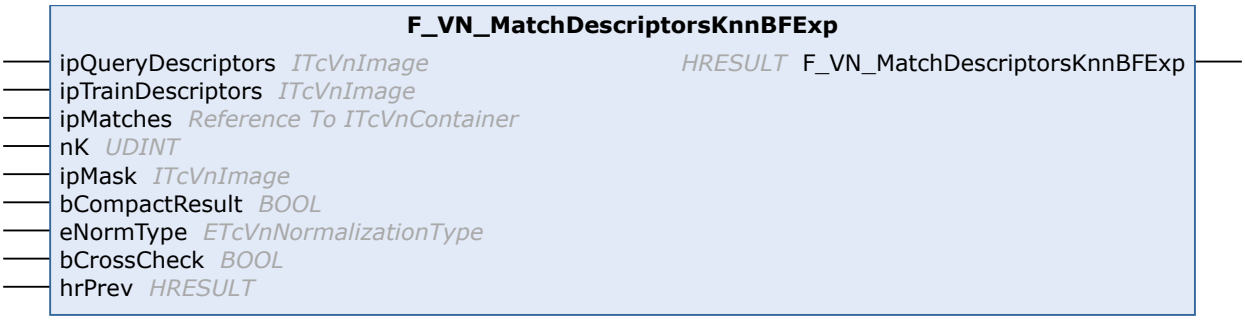
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 411]	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 [▶ 210]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

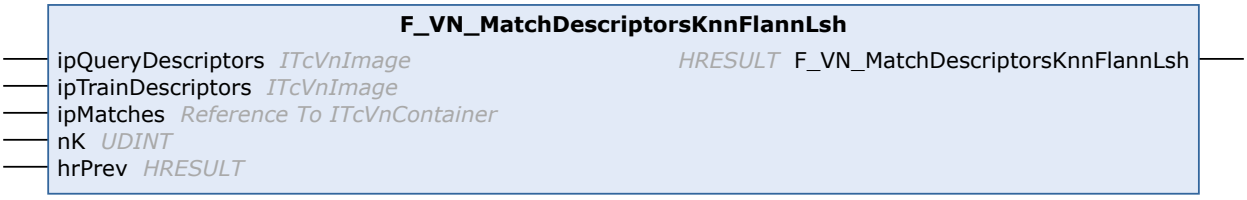
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

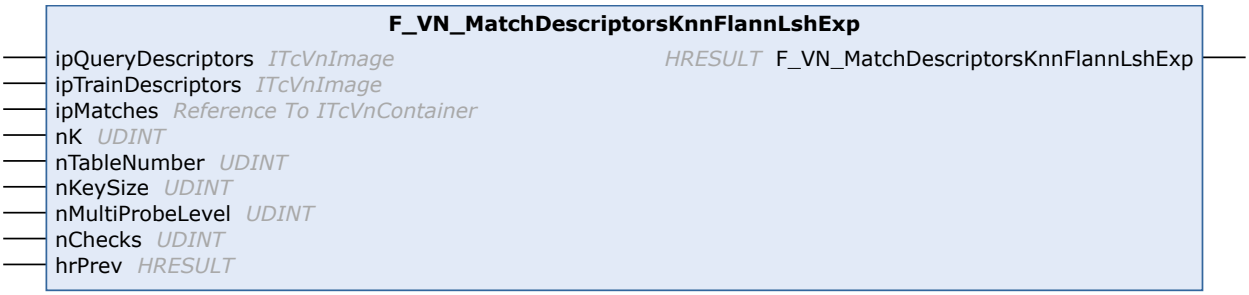
Required License

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 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 [▶ 411]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 411]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

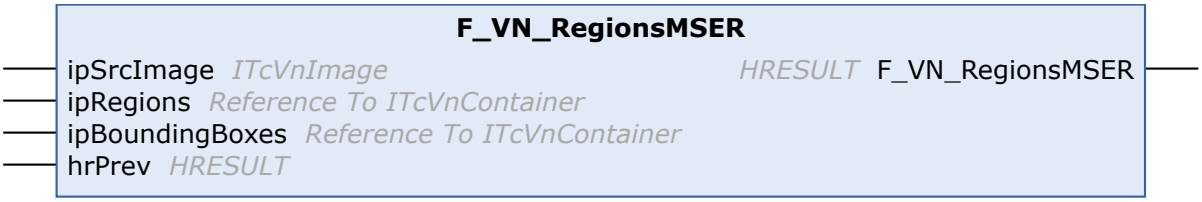
Required License

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 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 [▶ 411]	Source image
ipRegions	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

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 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 [▶ 411]	Source image
ipRegions	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsMSER [▶ 240]	Additional expert parameters

Return value

HRESULT [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19 Machine Learning

This group contains functions for creating, training and applying Machine Learning models as well as functions for feature pre-processing. A brief overview of the possible applications of Machine Learning [▶ 143] and further descriptions can be found under the software concept [▶ 137].

Functions for creating a model:

- F_VN_CreateBoostClassifier(Exp) [▶ 1385]
- F_VN_CreateKmppModel(Exp) [▶ 1390]
- F_VN_CreateKnnModel [▶ 1394]
- F_VN_CreateLbgModel(Exp) [▶ 1395]
- F_VN_CreateLdaTransform [▶ 1399]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1401]
- F_VN_CreateNbcModel [▶ 1402]
- F_VN_CreatePcaTransform [▶ 1404]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1405]
- F_VN_CreatePcaTransformViaVariance [▶ 1407]
- F_VN_CreateRTreesModel(Exp) [▶ 1408]
- F_VN_CreateStaModel(Exp) [▶ 1414]

- F_VN_CreateSvmModel(Exp) [▶ 1419]
- F_VN_CreateSvmSgdClassifier(Exp) [▶ 1427]

Functions for training a model:

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

Functions for obtaining information on the cluster models:

- F_VN_GetClusterCenter [▶ 1440]
- F_VN_GetClusterNum [▶ 1442]

Functions for applying models to a sample:

- F_VN_GetSampleCluster(Exp) [▶ 1445]
- F_VN_GetSampleNovelty [▶ 1448]
- F_VN_PredictSampleClass (Exp) [▶ 1461]
- F_VN_PredictSampleScalar(Exp) [▶ 1464]
- F_VN_PredictSampleVector(Exp) [▶ 1467]

Functions for applying models to a batch:

- F_VN_GetBatchClusters(Exp) [▶ 1436]
- F_VN_GetBatchNovelties [▶ 1439]
- F_VN_PredictBatch(Exp) [▶ 1457]

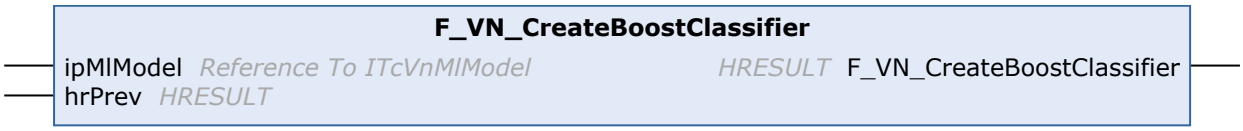
Functions for applying models for feature transformation / dimension reduction:

- F_VN_FeatureTransform [▶ 1434]
- F_VN_InverseFeatureTransform [▶ 1456]

Functions for feature normalization:

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_FeatureScaling(Exp) [▶ 1431]
- F_VN_InverseFeatureScaling(Exp) [▶ 1449]
- F_VN_InverseFeatureScaling(Exp)_REAL [▶ 1451]

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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateBoostClassifier` creates a boost classifier model.

Boost classifier models

The boost classifier model is only suitable for binary classification problems. This model learns to distinguish between samples labeled with two user-defined class labels by incrementally adding weak classifiers to improve classification results.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Expert parameters

The expert variants F_VN_CreateBoostClassifierExp [▶ 1387] and F_VN_CreateBoostClassifierExp2 [▶ 1388] contain additional parameters.

Application

A boost classifier model can be created like this, for example:

```
hr := F_VN_CreateBoostClassifier(
    ipMlModel   := ipMlModel,
    hrPrev      := hr);
```

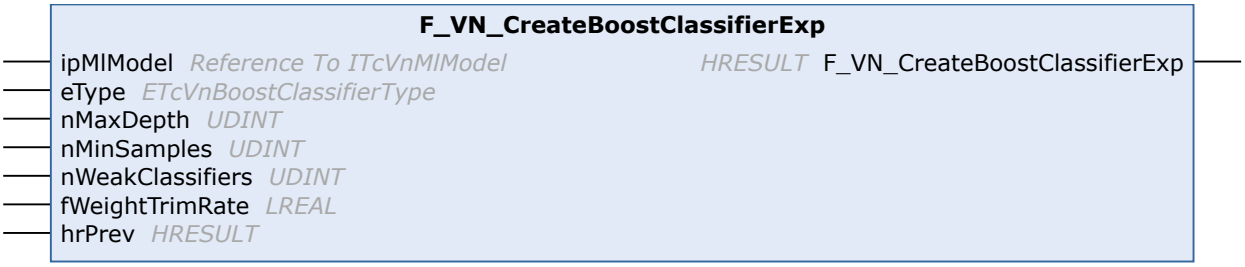
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 166]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateBoostClassifierExp` is an expert variant of F_VN_CreateBoostClassifier [▶ 1385]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Learning technique

The learning algorithm can be selected via `eType`. Supported are: Discrete AdaBoost, Real AdaBoost, LogitBoost and Gentle AdaBoost.

Maximum tree depth

`nMaxDepth` indicates the maximum number of decision levels in a tree. Decision trees with a tree depth of 1 are usually used for boost classifiers. However, higher values are possible.

Minimum number of samples within a node

`nMinSamples` is the minimum number of samples that must remain in a node during tree formation. Splits that would fall below this number are not carried out.

Number of decision trees

`nWeakClassifiers` indicates the number of decision trees (weak learners) that are combined for the classifier by boosting.

Runtime optimization

With `fWeightTrimRate`, the number of samples used for the individual learning steps can be limited using their automatically determined weightings. Values in the interval from 0 to 1 are permitted, whereby only samples with weightings > 1 - `fWeightTrimRate` are taken into account in the following learning step. The value 0 disables this optimization.

Application

A boost classifier model can be created like this, for example:

```
hr := F_VN_CreateBoostClassifierExp(
    ipMlModel           := ipMlModel,
    eType               := TCVN_BCT_REAL,
    nMaxDepth           := 1,
    nMinSamples         := 10,
    nWeakClassifiers    := 100,
    fWeightTrimRate     := 0.95,
    hrPrev              := hr);
```

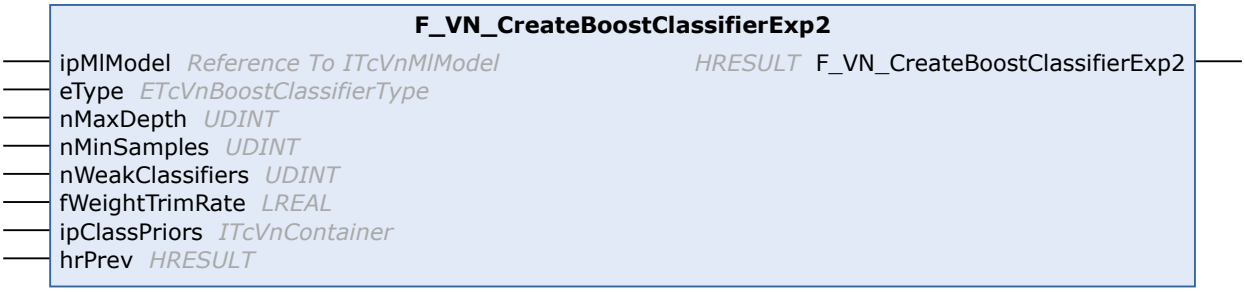
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 166]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateBoostClassifierExp2` is an expert variant of F_VN_CreateBoostClassifier [▶ 1385]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Learning technique

The learning algorithm can be selected via `eType`. Supported are: Discrete AdaBoost, Real AdaBoost, LogitBoost and Gentle AdaBoost.

Maximum tree depth

`nMaxDepth` indicates the maximum number of decision levels in a tree. Decision trees with a tree depth of 1 are usually used for boost classifiers. However, higher values are possible.

Minimum number of samples within a node

`nMinSamples` is the minimum number of samples that must remain in a node during tree formation. Splits that would fall below this number are not carried out.

Number of decision trees

`nWeakClassifiers` indicates the number of decision trees (weak learners) that are combined for the classifier by boosting.

Runtime optimization

With `fWeightTrimRate`, the number of samples used for the individual learning steps can be limited using their automatically determined weightings. Values in the interval from 0 to 1 are permitted, whereby only samples with weightings > 1 - `fWeightTrimRate` are taken into account in the following learning step. The value 0 disables this optimization.

Class-Priors

`ipClassPriors` is a container that can be used to specify the a priori probabilities of the individual classes.

Application

A boost classifier model can be created like this, for example:

```
hr := F_VN_CreateBoostClassifierExp2(
    ipMlModel           := ipMlModel,
    eType               := TCVN_BCT_REAL,
    nMaxDepth           := 1,
    nMinSamples         := 10,
    nWeakClassifiers    := 100,
    fWeightTrimRate     := 0.95,
    ipClassPriors       := 0,
    hrPrev              := hr);
```

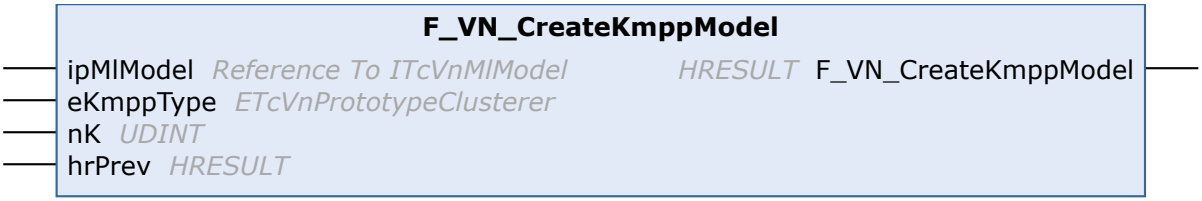
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 214]	k-means++ model type
nK	UDINT	Parameter k (number of clusters)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateKmppModel` creates a K-Means++ (KMPP) model.

K-Means++ models

K-Means++ can be used for clustering or anomaly detection. It finds clusters of data points. All samples are required for training at the same time (batch training) and post-training is not possible.

The assignment of cluster designations (as `DINT`s) may differ from training to training. If this assignment is to be deterministic, a fixed random seed must be set beforehand using F_VN_SetRngSeed [▶ 1546].

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eKmppType` specifies whether K-Means++ is used for clustering (`TCVN_PC_CLUSTERER`) or for anomaly detection (`TCVN_PC_NOVELTY_DETECTOR`).

Number of clusters

The number of clusters in the K-Means++ model is specified by `nK` and cannot be changed once it has been created. During training, the number of samples must be greater than or equal to the number of clusters.

Expert parameters

The expert variant F_VN_CreateKmppModelExp [▶ 1392] contains additional parameters.

Application

For example, a K-Means++ model for clustering with 5 clusters can be created like this:

```
hr := F_VN_CreateKmppModel(
    ipMlModel   := ipMlModel,
    eKmppType   := TCVN_PC_CLUSTERER,
    nK          := 5,
    hrPrev      := hr);
```

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 214]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateKmppModelExp` is the expert variant of F_VN_CreateKmppModel [▶ 1390]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eKmppType` specifies whether K-Means++ is used for clustering (`TCVN_PC_CLUSTERER`) or for anomaly detection (`TCVN_PC_NOVELTY_DETECTOR`).

Number of clusters

The number of clusters in the K-Means++ model is specified by `nK` and cannot be changed once it has been created. During training, the number of samples must be greater than or equal to the number of clusters.

Higher accuracy

If `bDoublePrecision` is `TRUE`, `LREAL` is used as the data type for the internal calculations of the model; if it is `FALSE`, `REAL` is used.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Application

For example, a K-Means++ model for clustering with 5 clusters can be created like this:

```
hr := F_VN_CreateKmppModelExp(
    ipMlModel           := ipMlModel,
    eKmppType           := TCVN_PC_CLUSTERER,
    nK                  := 5,
    bDoublePrecision    := FALSE,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    hrPrev              := hr);
```

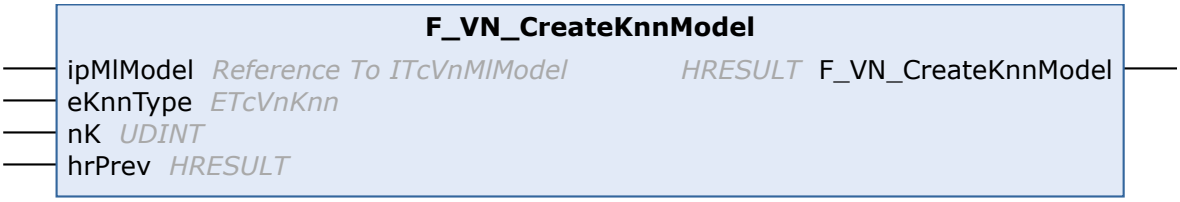
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eKnnType	ETcVnKnn [▶ 207]	k-nearest neighbors model type
nK	UDINT	Parameter k used for prediction (number of considered neighbors)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateKnnModel` creates a K-Nearest Neighbor (KNN) model.

K-Nearest Neighbor models

When training K-Nearest Neighbor models, all training samples are stored 1:1 in the model. The idea of data points within a multidimensional space (corresponding to the dimensionality of the samples) helps to understand this. A number `nK` of data points that are closest to the input sample (Euclidean distance) are used for prediction. These data points are called "nearest neighbors". All other data points are ignored in the prediction. The prediction result is calculated from all nearest neighbors, whereby these are weighted equally regardless of the actual distance to the input sample.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eKnnType` indicates whether the model is used as a classifier (`TCVN_KNN_CLASSIFIER`), as anomaly detection (`TCVN_KNN_NOVELTY_DETECTOR`) or as a regressor (`TCVN_KNN_REGRESSOR`).

Number of neighbors used

`nK` specifies how many data points are used as nearest neighbors in the prediction. For anomaly detection, `nK` is irrelevant because only the nearest neighbor is required. The value 1 should therefore be used in this case.

Application

For example, a K-nearest Neighbor model for classification using 5 nearest neighbors is created like this:

```
hr := F_VN_CreateKnnModel(
    ipMlModel   := ipMlModel,
    eKnnType    := TCVN_KNN_CLASSIFIER,
    nK          := 5,
    hrPrev      := hr);
```

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 214]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateLbgModel` creates a Linde-Buzo-Gray (LBG) model.

Linde-Buzo-Gray models

LBG models work similarly to K-Means++ models by iteratively adjusting the clusters to the samples. The main feature of LBG is that new clusters are created in the iteration process as soon as they are larger than `fMaxClusterRadius` (until the maximum number `nMaxClusters` is reached). With K-Means++, on the other hand, the number of clusters is set initially.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eLbgType` specifies whether LBG is used for clustering (`TCVN_PC_CLUSTERER`) or for anomaly detection (`TCVN_PC_NOVELTY_DETECTOR`).

Maximum number of clusters

If `nMaxClusters` exist in total due to the splitting of clusters that are too large, no more new clusters are created.

Maximum cluster radius

If the radius of a cluster becomes larger than `fMaxClusterRadius` during training, this cluster is split into two smaller clusters.

Optimization quality

Part of the iterative training process is a global optimization step. If `bSingleSplitSteps = TRUE`, this optimization is executed after each individual cluster split. This leads to a higher quality of clusters, but also to a longer training time. If `bSingleSplitSteps = FALSE`, the optimization is only executed after several clusters have been split.

Expert parameters

The expert variant F_VN_CreateLbgModelExp [▶ 1397] contains additional parameters.

Application

For example, an LBG model for clustering with `5` clusters, a maximum cluster size of `3.6` and high optimization quality can be created in this way:

```
hr := F_VN_CreateLbgModel(
    ipMlModel           := ipMlModel,
    eLbgType            := TCVN_PC_CLUSTERER,
    nMaxClusters        := 5,
    fMaxClusterRadius   := 3.6,
    bSingleSplitSteps   := TRUE,
    hrPrev              := hr);
```

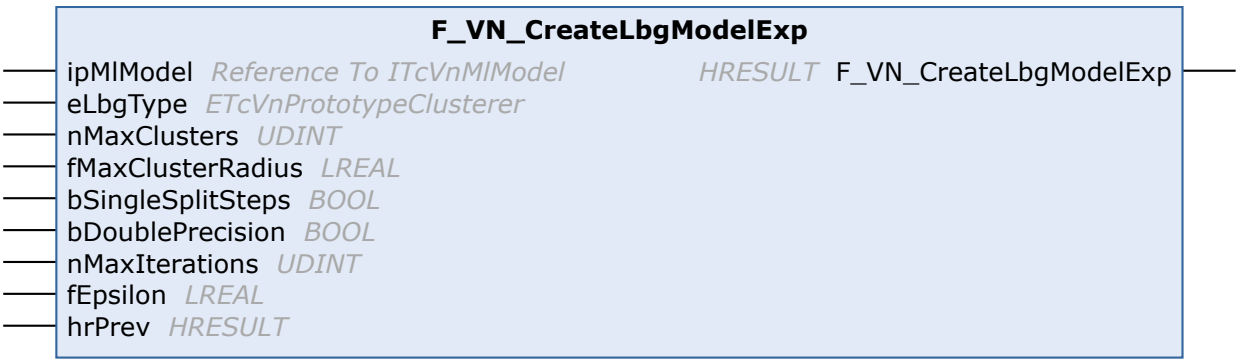
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 214]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateLbgModelExp` is the expert variant of F_VN_CreateLbgModel [▶ 1395]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eLbgType` specifies whether LBG is used for clustering (`TCVN_PC_CLUSTERER`) or for anomaly detection (`TCVN_PC_NOVELTY_DETECTOR`).

Maximum number of clusters

If `nMaxClusters` exist in total due to the splitting of clusters that are too large, no more new clusters are created.

Maximum cluster radius

If the radius of a cluster becomes larger than `fMaxClusterRadius` during training, this cluster is split into two smaller clusters.

Optimization quality

Part of the iterative training process is a global optimization step. If `bSingleSplitSteps = TRUE`, this optimization is executed after each individual cluster split. This leads to a higher quality of clusters, but also to a longer training time. If `bSingleSplitSteps = FALSE`, the optimization is only executed after several clusters have been split.

Higher accuracy

If `bDoublePrecision` is `TRUE`, `LREAL` is used as the data type for the internal calculations of the model; if it is `FALSE`, `REAL` is used.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Application

For example, an LBG model for clustering with `5` clusters, a maximum cluster size of `3.6` and high optimization quality can be created in this way:

```
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);
```

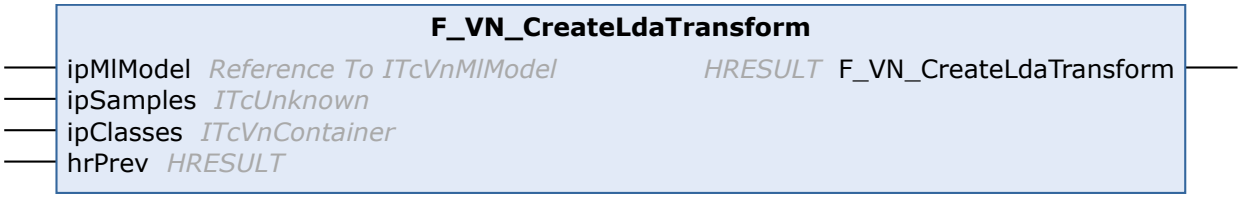
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer [▶ 370]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateLdaTransform` creates a Linear Discriminant Analysis (LDA) model for feature transformation.

Linear Discriminant Analysis models

Linear Discriminant Analysis creates a feature transformation with the aim of making the classes as easy to separate as possible using the transformed features. The model can be used to compress the samples effectively.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Samples

All samples are sent to the model in a container via `ipSamples`.

Classes

The class assignments of all samples in a container are given to the model via `ipClasses`.

Application

For example, an LDA model for feature reduction can be created like this:

```
hr := F_VN_CreateLdaTransform(
    ipMlModel   := ipMlModel,
    ipSamples   := ipSamples,
    ipClasses   := ipClasses,
    hrPrev      := hr);
```

Related functions

- F_VN_CreateLdaTransformViaComponentNum [▶ 1401]
- F_VN_CreatePcaTransform [▶ 1404]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1405]
- F_VN_CreatePcaTransformViaVariance [▶ 1407]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateLdaTransformViaComponentNum` is a variant of F_VN_CreateLdaTransform [▶ 1399]. It contains the additional parameter `nComponentNum` to specify the number of components for the feature reduction in advance. This allows the level of data compression to be set.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Samples

All samples are sent to the model in a container via `ipSamples`.

Classes

The class assignments of all samples in a container are given to the model via `ipClasses`.

Number of components

`nComponentNum` components are used for feature reduction. At 0, the maximum number of components is automatically calculated.

Application

For example, an LDA model with 25 components can be created like this:

```
hr := F_VN_CreateLdaTransformViaComponentNum(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
    ipClasses       := ipClasses,
    nComponentNum   := 25,
    hrPrev          := hr);
```

Related functions

- F_VN_CreateLdaTransform [▶ 1399]
- F_VN_CreatePcaTransform [▶ 1404]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1405]
- F_VN_CreatePcaTransformViaVariance [▶ 1407]

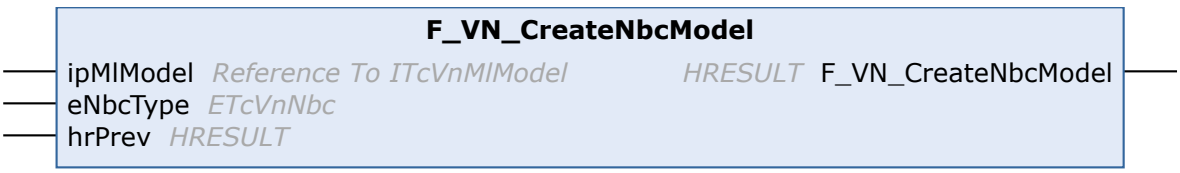
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eNbcType	ETcVnNbc [▶ 209]	Normal Bayes classifier type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateNbcModel` creates a Normal Bayes Classifier (NBC) model.

Normal Bayes Classifier models

The Normal Bayes Classifier creates a mixture model in which each class is represented by a Gaussian distribution. The parameters of these Gaussian functions are determined during training.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eNbcType` specifies whether NBC is used for classification (`TCVN_NBC_CLASSIFIER`) or for anomaly detection (`TCVN_NBC_NOVELTY_DETECTOR`).

Application

For example, an NBC model for classification can be created like this:

```
hr := F_VN_CreateNbcModel(
    ipMlModel   := ipMlModel,
    eNbcType    := TCVN_NBC_CLASSIFIER,
    hrPrev      := hr);
```

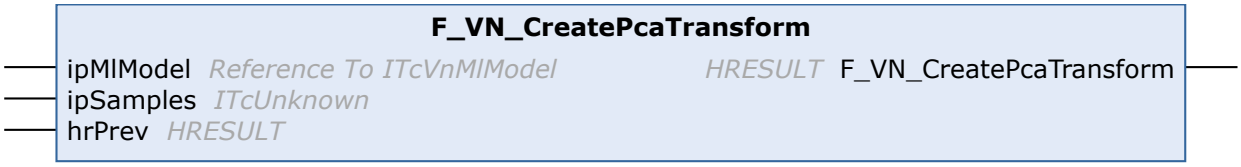
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreatePcaTransform` creates a Principal Component Analysis (PCA) model.

Principal Component Analysis models

In Principal Component Analysis, the principal components of the sample distribution are calculated. The principal components are the directions within the feature space that show the greatest variance in the samples. The dimension of the feature space is reduced by projecting the data onto selected principal components, in which the variance of the data is retained as much as possible.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Samples

All samples are sent to the model in a container via `ipSamples`.

Application

For example, a PCA model for feature reduction can be created like this:

```
hr := F_VN_CreatePcaTransform(
    ipMlModel   := ipMlModel,
    ipSamples   := ipSamples,
    hrPrev      := hr);
```

Related functions

- F_VN_CreatePcaTransformViaComponentNum [▶ 1405]
- F_VN_CreatePcaTransformViaVariance [▶ 1407]
- F_VN_CreateLdaTransform [▶ 1399]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1401]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 428]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreatePcaTransformViaComponentNum` is a variant of F_VN_CreatePcaTransform [▶ 1404]. It contains the additional parameter `nComponentNum` to define the number of principal components for the feature reduction in advance.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Samples

All samples are sent to the model in a container via `ipSamples`.

Number of components

The PCA calculates `nComponentNum` principal components. At 0, the maximum number of components is automatically calculated.

Application

For example, a PCA model with 5 principal components can be created like this:

```
hr := F_VN_CreatePcaTransformViaComponentNum(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
    nComponentNum   := 5,
    hrPrev          := hr);
```

Related functions

- F_VN_CreatePcaTransform [▶ 1404]
- F_VN_CreatePcaTransformViaVariance [▶ 1407]
- F_VN_CreateLdaTransform [▶ 1399]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1401]

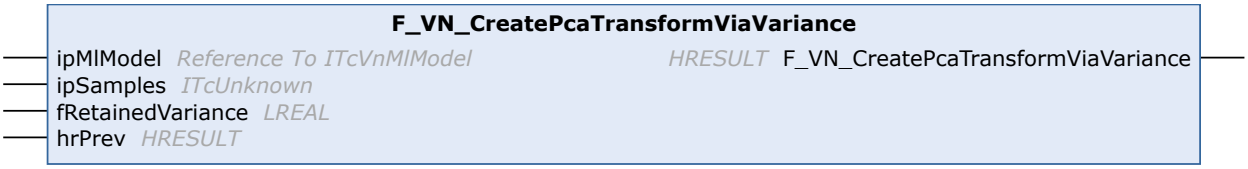
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 428]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreatePcaTransformViaVariance` is a variant of F_VN_CreatePcaTransform [▶ 1404]. It contains the additional parameter `fRetainedVariance` to define the retained variance.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Samples

All samples are sent to the model in a container via `ipSamples`.

Retained variance

In PCA, the principal components are chosen so that the proportion `fRetainedVariance` of variance remains in the samples. With `1`, the entire variance remains, so that the number of features is not reduced.

Application

For example, a PCA model with 65% retained variance can be created like this:

```
hr := F_VN_CreatePcaTransformViaComponentNum(
    ipMlModel           := ipMlModel,
    ipSamples           := ipSamples,
    fRetainedVariance   := 0.65,
    hrPrev              := hr);
```

Related functions

- F_VN_CreatePcaTransform [▶ 1404]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1405]
- F_VN_CreateLdaTransform [▶ 1399]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1401]

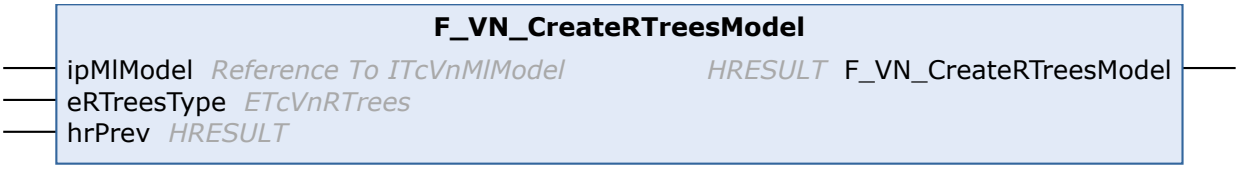
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 215]	Random trees model type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateRTreesModel` creates a Random Forest (RTrees) model.

Random Forest models

The Random Forest model uses decision trees whose branches are determined by a random process. The prediction of the RTrees models is determined from the results of the individual decision trees.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eRTreesType` specifies whether the Random Forest model is used for classification (`TCVN_RT_CLASSIFIER`) or for regression (`TCVN_RT_REGRESSOR`).

Expert parameters

The expert variants F_VN_CreateRTreesModelExp [▶ 1410] and F_VN_CreateRTreesModelExp2 [▶ 1412] contain additional parameters.

Application

For example, an RTrees model for regression can be created like this:

```
hr := F_VN_CreateRTreesModel(
    ipMlModel   := ipMlModel,
    eRTreesType := TCVN_RT_REGRESSOR,
    hrPrev      := hr);
```

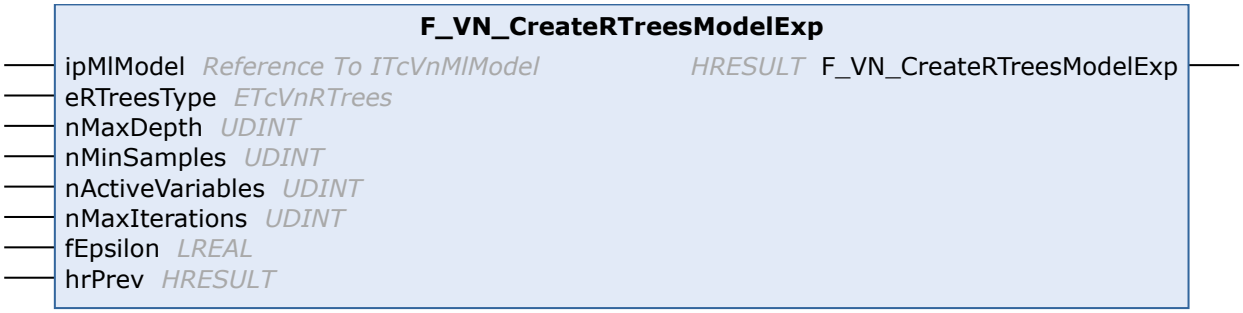
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 215]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateRTReesModelExp` is an expert variant of F_VN_CreateRTReesModel [▶ 1408]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eRTreesType` specifies whether the Random Forest model is used for classification (`TCVN_RT_CLASSIFIER`) or for regression (`TCVN_RT_REGRESSOR`).

Maximum tree depth

`nMaxDepth` indicates the maximum number of decision levels in a tree.

Minimum number of samples within a node

`nMinSamples` is the minimum number of samples that must remain in a node during tree formation. Splits that would fall below this number are not carried out.

Splitting features

`nActiveVariables` is the number of features used to split the tree. With `0`, `sqrt(N)` features are used with the number `N` of all available features.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Application

For example, an RTrees model for regression can be created like this:

```
hr := F_VN_CreateRTreesModelExp(
    ipMlModel           := ipMlModel,
    eRTreesType         := TCVN_RT_REGRESSOR,
    nMaxDepth           := 8,
    nMinSamples         := 4,
    nActiveVariables    := 0,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    hrPrev              := hr);
```

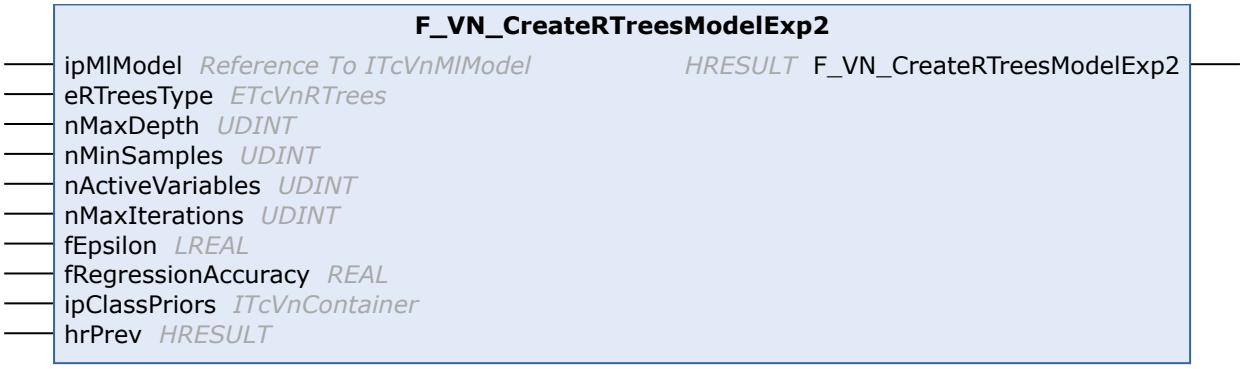
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 215]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateRTReesModelExp2` is an expert variant of F_VN_CreateRTreesModel [▶ 1408]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eRTreesType` specifies whether the Random Forest model is used for classification (`TCVN_RT_CLASSIFIER`) or for regression (`TCVN_RT_REGRESSOR`).

Maximum tree depth

`nMaxDepth` indicates the maximum number of decision levels in a tree.

Minimum number of samples within a node

`nMinSamples` is the minimum number of samples that must remain in a node during tree formation. Splits that would fall below this number are not carried out.

Splitting features

`nActiveVariables` is the number of features used to split the tree. With `0`, `sqrt(N)` features are used with the number `N` of all available features.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Regression accuracy

`fRegressionAccuracy` is the abort criterion for regression optimization. For a classification, the value must be set to `0`.

Class-Priors

`ipClassPriors` is a container that can be used to specify the a priori probabilities of the individual classes. For a regression, the value should be set to `0`.

Application

For example, an RTrees model for regression can be created like this:

```
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);
```

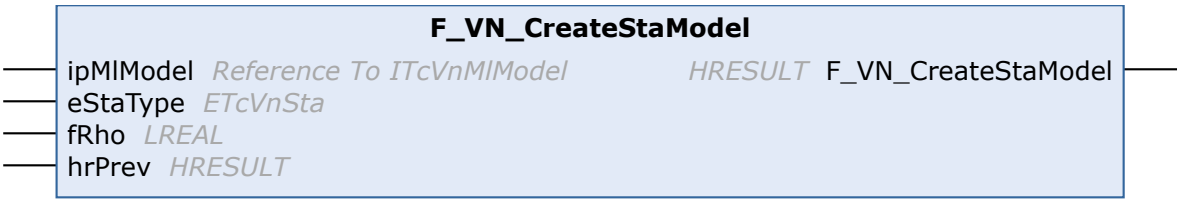
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 217]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateStaModel` creates a Simplified TopoART (STA) model.

Simplified TopoART models

The Simplified TopoART network learns sub-areas of the feature space. Depending on the application, these are grouped into clusters, assigned to classes or used for prediction. Due to the incremental architecture, post-training is possible, whereby the information already learned is retained in the best possible way. Only values in the interval of 0.0 and 1.0 are accepted as inputs (with the exception of class names). It is therefore advisable to use a feature normalization.

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eStaType` specifies whether the STA model is used for classification (`TCVN_STA_CLASSIFIER`), regression (`TCVN_STA_REGRESSOR`), clustering (`TCVN_STA_CLUSTERER`) or anomaly detection (`TCVN_STA_NOVELTY_DETECTOR`).

Vigilance parameters

This parameter is crucial for adapting the model to the respective task, as it controls the maximum size of the sub-areas to be learned.

Expert parameters

The expert variants F_VN_CreateStaModelExp [▶ 1415] and F_VN_CreateStaModelExp2 [▶ 1417] contain additional parameters.

Application

For example, an STA model for classification can be created like this:

```
hr := F_VN_CreateStaModel(
    ipMlModel   := ipMlModel,
    eStaType    := TCVN_STA_CLASSIFIER,
    fRho        := 0.9,
    hrPrev      := hr);
```

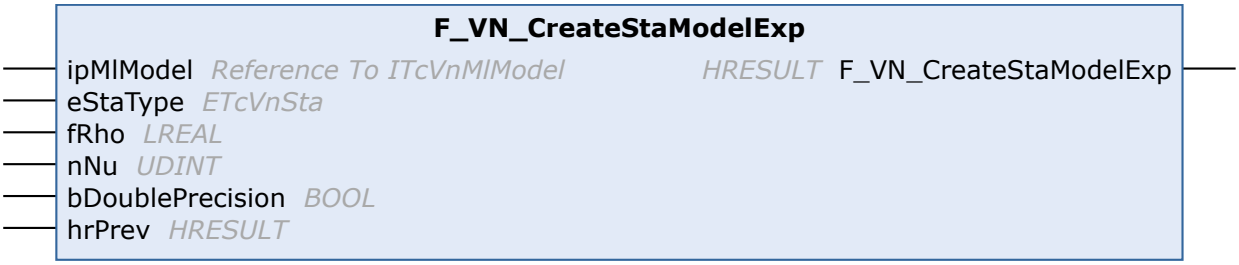
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 217]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateStaModelExp` is an expert variant of F_VN_CreateStaModel [▶ 1414]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eStaType` specifies whether the STA model is used for classification (`TCVN_STA_CLASSIFIER`), regression (`TCVN_STA_REGRESSOR`), clustering (`TCVN_STA_CLUSTERER`) or anomaly detection (`TCVN_STA_NOVELTY_DETECTOR`).

Vigilance parameters

This parameter is crucial for adapting the model to the respective task, as it controls the maximum size of the sub-areas to be learned.

Number of neighbors used

`nNu` specifies the number of neighboring sub-areas used for classification and prediction and must be >= 1.

Higher accuracy

If `bDoublePrecision` is `TRUE`, `LREAL` is used as the data type for the internal calculations of the model; if it is `FALSE`, `REAL` is used.

Application

For example, an STA model for classification can be created like this:

```
hr := F_VN_CreateStaModel(
    ipMlModel        := ipMlModel,
    eStaType         := TCVN_STA_CLASSIFIER,
    fRho             := 0.9,
    nNu              := 3,
    bDoublePrecision := FALSE,
    hrPrev           := hr);
```

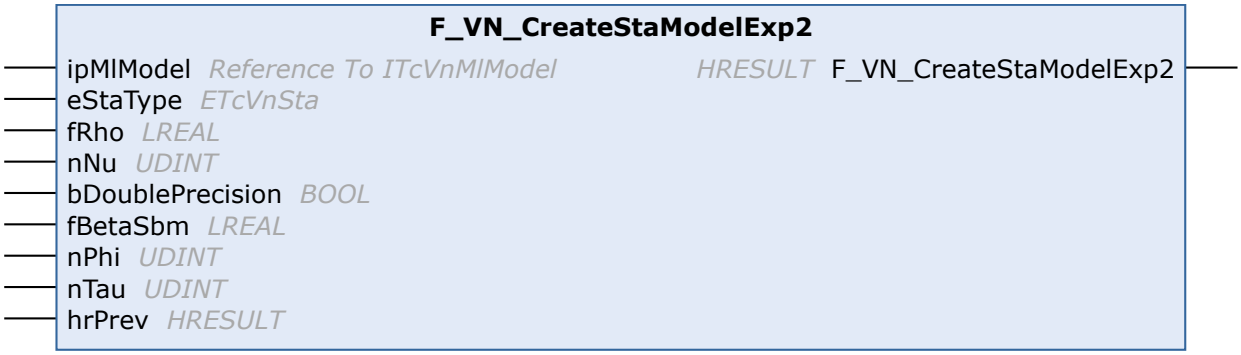
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 217]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateStaModelExp2` is an expert variant of F_VN_CreateStaModel [▶ 1414]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eStaType` specifies whether the STA model is used for classification (`TCVN_STA_CLASSIFIER`), regression (`TCVN_STA_REGRESSOR`), clustering (`TCVN_STA_CLUSTERER`) or anomaly detection (`TCVN_STA_NOVELTY_DETECTOR`).

Vigilance parameters

This parameter is crucial for adapting the model to the respective task, as it controls the maximum size of the sub-areas to be learned.

Number of neighbors used

`nNu` specifies the number of neighboring sub-areas used for classification and prediction and must be >= 1.

Higher accuracy

If `bDoublePrecision` is `TRUE`, `LREAL` is used as the data type for the internal calculations of the model; if it is `FALSE`, `REAL` is used.

Adjustment of the distribution

`fBetaSbm` supplements the control for adapting the sub-areas to the underlying sample distribution.

Noise suppression

`nPhi` and `nTau` are used to control the suppression of very rare samples (noise, outliers) in the learning process.

`nPhi` specifies the number of samples that must be learned in a sub-area so that this sub-area is permanently taken into account.

`nTau` specifies after how many learning steps the condition of `nPhi` is checked and sub-areas that do not fulfill this condition are removed.

Increasing `nPhi` and decreasing `nTau` increases the noise suppression, whereby the ratio of `nPhi` to `nTau` can be used as a measure to estimate the strength.

Application

For example, an STA model for classification can be created like this:

```
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);
```

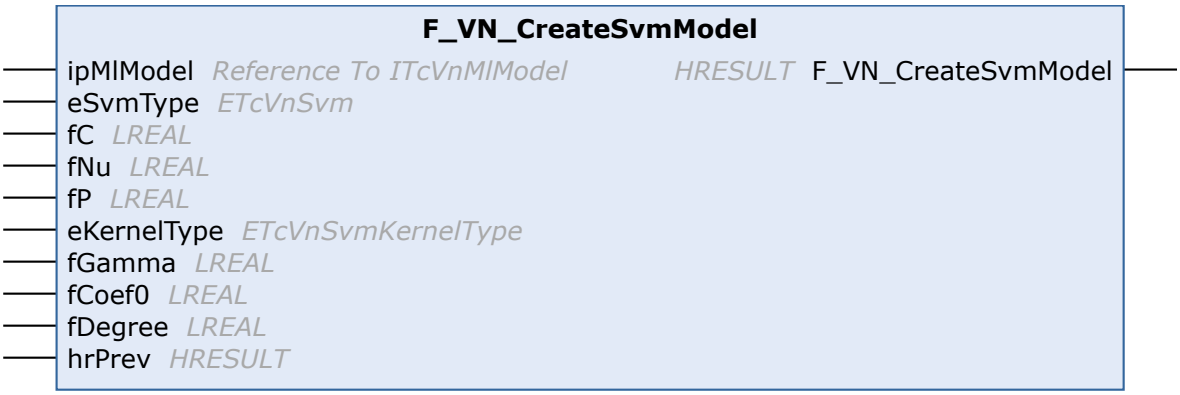
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 218]	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 [▶ 218]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateSvmModel` creates a Support Vector Machine (SVM) model.

Support Vector Machine models

Support Vector Machines can also be used for sophisticated non-linear problems. The prediction is based exclusively on the so-called support vectors, which is why SVM models are comparatively memory-efficient. All samples are required for training at the same time (batch training) and post-training is not possible.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eSvmType` specifies whether the SVM model is used for classification, regression or anomaly detection:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER
- TCVN_SVM_NOVELTY_DETECTOR
- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Model parameters

The use and meaning of the parameters `fC`, `fNu` and `fP` depends on the selected `eSvmType`.

Kernel type

The kernel type used for the calculation is defined with `eKernelType`. The kernel type depends on the task / data distribution and must be adapted accordingly.

Kernel parameters

The use and meaning of the parameters `fGamma`, `fCoef0` and `fDegree` depends on the selected `eKernelType`.

Expert parameters

The expert variants F_VN_CreateSvmModelExp [▶ 1422] and F_VN_CreateSvmModelExp2 [▶ 1425] contain additional parameters.

Application

For example, an SVM model for classification can be created like this:

```
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);
```

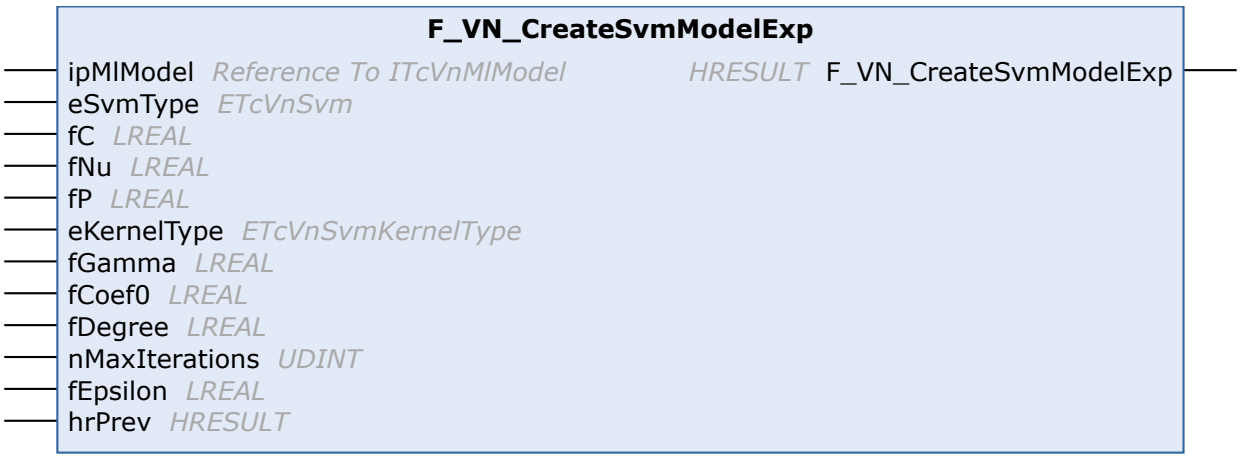
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 218]	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 [▶ 218]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateSvmModelExp` is an expert variant of F_VN_CreateSvmModel [▶ 1419]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eSvmType` specifies whether the SVM model is used for classification, regression or anomaly detection:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER
- TCVN_SVM_NOVELTY_DETECTOR
- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Model parameters

The use and meaning of the parameters `fC`, `fNu` and `fP` depends on the selected `eSvmType`.

Kernel type

The kernel type used for the calculation is defined with `eKernelType`. The kernel type depends on the task / data distribution and must be adapted accordingly.

Kernel parameters

The use and meaning of the parameters `fGamma`, `fCoef0` and `fDegree` depends on the selected `eKernelType`.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Application

For example, an SVM model for classification can be created like this:

```
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);
```

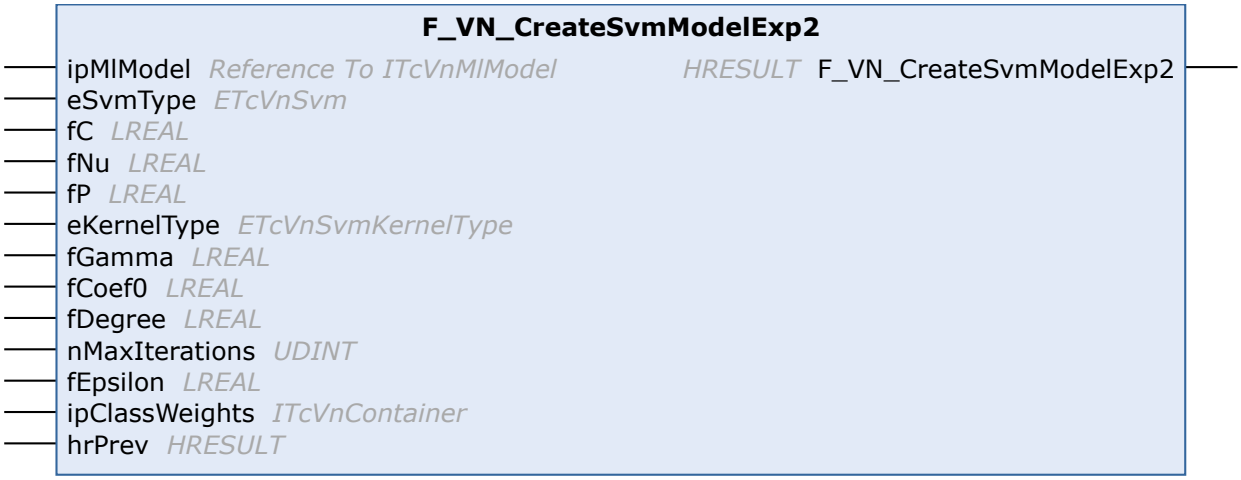
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 218]	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 [▶ 218]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateSvmModelExp2` is an expert variant of F_VN_CreateSvmModel [▶ 1419]. It contains additional parameters that are only relevant for models of type `TCVN_SVM_C_CLASSIFIER`.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Model type

`eSvmType` specifies whether the SVM model is used for classification, regression or anomaly detection:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER
- TCVN_SVM_NOVELTY_DETECTOR

- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Model parameters

The use and meaning of the parameters `fC`, `fNu` and `fP` depends on the selected `eSvmType`.

Kernel type

The kernel type used for the calculation is defined with `eKernelType`. The kernel type depends on the task / data distribution and must be adapted accordingly.

Kernel parameters

The use and meaning of the parameters `fGamma`, `fCoef0` and `fDegree` depends on the selected `eKernelType`.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Class-Weights

`ipClassWeights` is a container that can be used to specify the weighting of the individual classes.

Application

For example, an SVM model for classification parameterized with Nu can be created like this:

```
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 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateSvmSgdClassifier` creates a Support Vector Machine model with a linear kernel that is used in the Stochastic Gradient Descent (SVM-SGD) training.

SVM-SGD models

This SVM classifier model is only applicable to binary classification problems. It learns a separating hyperplane between a class with the label -1 and a class with the label 1. All samples are required simultaneously for the training (batch training) and post-training is not possible.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Expert parameters

The expert variant F_VN_CreateSvmSgdClassifierExp [▶ 1429] contains additional parameters.

Application

For example, an SVM-SGD model for classification can be created like this:

```
hr := F_VN_CreateSvmSgdClassifier(
    ipMlModel   := ipMlModel,
    hrPrev      := hr);
```

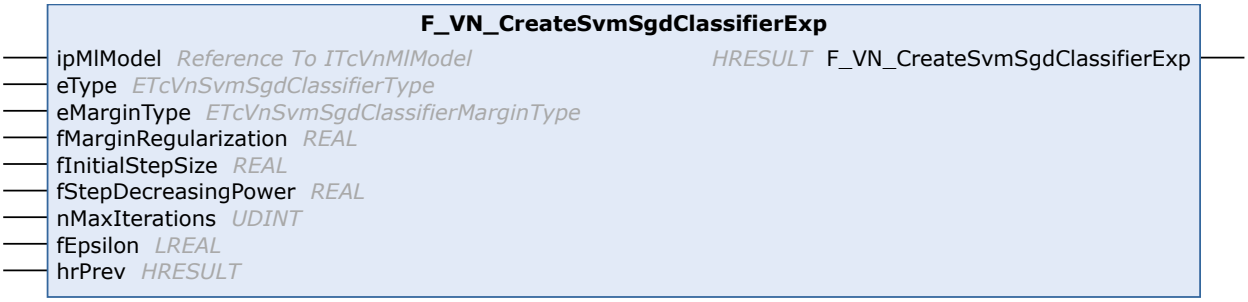
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnSvmSgdClassifierType [▶ 219]	Learning algorithm type (default: TCVN_SSCT_ASGD)
eMarginType	ETcVnSvmSgdClassifierMarginType [▶ 219]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_CreateSvmSgdClassifierExp` is an expert variant of F_VN_CreateSvmSgdClassifier [▶ 1427]. It contains additional parameters.

Parameter

Model

The created model is returned in the interface pointer `ipMlModel`.

Algorithm

`eType` specifies which optimization method is used in the training:

- TCVN_SSCT_SGD Stochastic Gradient Descent
- TCVN_SSCT_ASGD Average Stochastic Gradient Descent

Delimitation of the classes

`eMarginType` specifies whether the delimitation of the classes is to be carried out strictly (`TCVN_SSCMT_HARD_MARGIN`) or with outliers (`TCVN_SSCMT_SOFT_MARGIN`).

Regularization

`fMarginRegularization` is responsible for the weights decreasing at each step and controls the extent to which the influence of outliers is limited. The smaller the parameter, the lower the probability that an outlier will be ignored.

Step size

`fInitialStepSize` indicates the step size at the beginning of the training.

Decrease in step size

`fStepDecreasingPower` specifies the exponent for reducing the step size.

Maximum iterations

A maximum of as many iterations as specified in `nMaxIterations` are used for the optimization. If the value is `0`, the respective default value is used.

Termination limit

The optimization is aborted as soon as the error between two iterations does not change more than specified in `fEpsilon`. If the value is `0`, the respective default value is used.

Application

For example, an SVM-SGD model for classification can be created like this:

```
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);
```

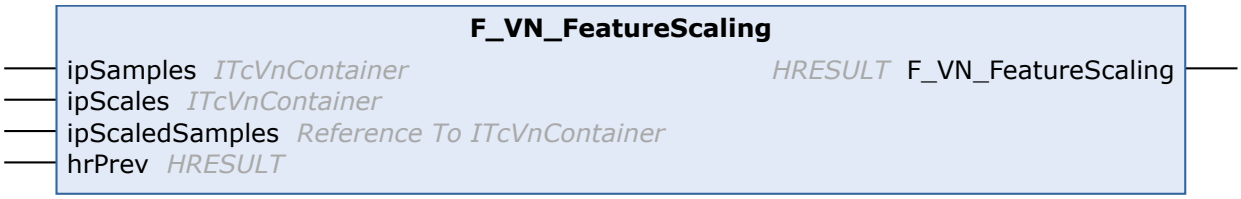
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FeatureScaling` scales features from several samples. The function F_VN_InverseFeatureScaling [▶ 1449] can be used for inverse scaling.

Parameter

Samples

One or more samples must be transferred as a container to `ipSamples`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled samples

The scaled samples are returned via the reference `ipScaledSamples`.

Expert parameters

The expert variant F_VN_FeatureScalingExp [▶ 1433] contains additional parameters.

Application

For example, a feature scaling in which the scaling parameters are calculated beforehand by F_VN_GetFeatureScales [▶ 1443] using the `MINMAX` method looks like this:

```
hr := F_VN_GetFeatureScales(ipSamples, ipScales, TCVN_FST_MINMAX, hr);

hr := F_VN_FeatureScaling(
    ipSamples       := ipSamples,
    ipScales        := ipScales,
    ipScaledSamples := ipScaledSamples,
    hrPrev          := hr);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_InverseFeatureScaling [▶ 1449]
- F_VN_InverseFeatureScaling_REAL [▶ 1451]

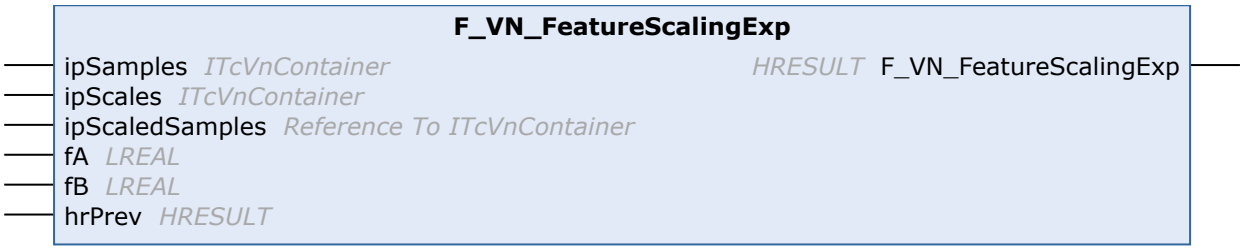
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FeatureScalingExp` is the expert variant of F_VN_FeatureScaling [▶ 1431]. It contains additional parameters.

Parameter

Samples

One or more samples must be transferred as a container to `ipSamples`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled samples

The scaled samples are returned via the reference `ipScaledSamples`.

Defining the output range

If the scaling type `TCVN_FST1_MINMAX` is used, the value range of the output values can also be specified via `fA` and `fB`.

Application

For example, a feature scaling with an output value range of 0.1 to 0.9 looks like this:

```
hr := F_VN_FeatureScalingExp(
    ipSamples       := ipSamples,
    ipScales        := ipScales,
    ipScaledSamples := ipScaledSamples,
    fA              := 0.1,
    fB              := 0.9,
    hrPrev          := hr);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_InverseFeatureScaling [▶ 1449]
- F_VN_InverseFeatureScaling_REAL [▶ 1451]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Feature transform instance to be used
ipSamples	ITcUnknown [▶ 428]	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 [▶ 428]	Returns the transformed sample(s) using the same type like ipSamples
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_FeatureTransform` transforms the features of one or more samples using a transformation model. An inverse transform can be carried out with the F_VN_InverseFeatureTransform [▶ 1456] function.

Parameter

Transformation model

A previously created model must be transferred to `ipFeatureTransform` for feature transformation.

Samples

A container with the samples to be transformed is transferred to `ipSamples`. The following container types are permitted:

- `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL` for a single sample
- `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL` for a container with several samples.

Transformed samples

A container with the transformed samples is returned via the pointer `pipTransformed`. The container type is taken from `ipSamples`.

Application

For example, a feature transformation is carried out as follows:

```
hr := F_VN_FeatureTransform(
    ipFeatureTransform  := ipMlModel,
    ipSamples           := ipSamples,
    pipTransformed      := ADR(ipTransformed),
    hrPrev              := hr);
```

Related functions

- F_VN_InverseFeatureTransform [▶ 1456]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 370]	Returns the cluster IDs (ContainerType_Vector_DINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetBatchClusters` determines the best matching cluster for all samples in a container and returns its ID.

Parameter

Clusterer model

A previously trained model must be transferred to `ipClusterer` for assignment to clusters.

Samples

A container with the samples is transferred to `ipSamples`. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Cluster IDs

A container with the determined IDs is returned via the reference `ipClusters`.

Expert parameters

The expert variant F_VN_GetBatchClustersExp [▶ 1437] contains additional parameters.

Application

The assignment of samples to clusters can be calculated in this way:

```
hr := F_VN_GetBatchClusters(
    ipClusterer := ipClusterer,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    hrPrev      := hr);
```

Related functions

- F_VN_GetSampleCluster [▶ 1445]

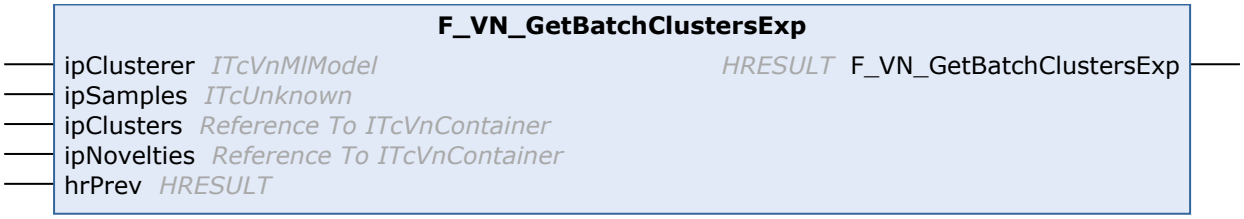
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 370]	Returns the cluster IDs (ContainerType_Vector_DINT)
ipNovelties	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetBatchClustersExp` is the expert variant of F_VN_GetBatchClusters [▶ 1436]. It contains additional parameters.

Parameter

Clusterer model

A previously trained model must be transferred to `ipClusterer` for assignment to clusters.

Samples

A container with the samples is transferred to `ipSamples`. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Cluster IDs

A container with the determined IDs is returned via the reference `ipClusters`.

Degrees of novelty

A container with the degrees of novelty of each sample is returned via the reference `ipNovelties`. The degree of novelty indicates how strongly a sample differs from the clusters. A degree of novelty of `0` indicates that a sample lies exactly in the center of a cluster.

Application

The assignment of samples to clusters, including the degrees of novelty, can be calculated as follows:

```
hr := F_VN_GetBatchClustersExp(
    ipClusterer := ipClusterer,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    ipNovelties := ipNovelties,
    hrPrev      := hr);
```

Related functions

- F_VN_GetSampleCluster [▶ 1445]
- F_VN_GetSampleNovelty [▶ 1448]
- F_VN_GetBatchNovelties [▶ 1439]

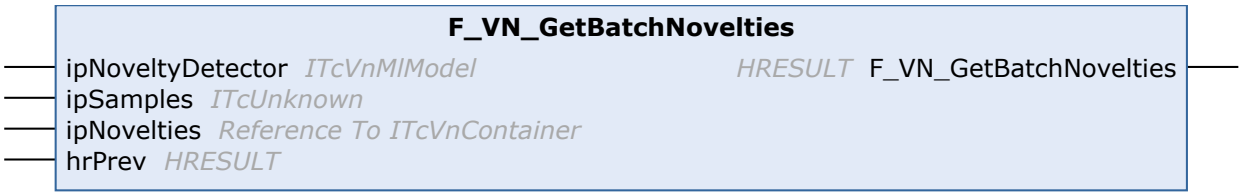
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Novelty detector to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipNovelties	Reference To ITcVnContainer [▶ 370]	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetBatchNovelties` determines the degree of novelty for all samples in a container.

Parameter

Anomaly model

To calculate the degree of novelty, the previously trained model must be transferred to `ipNoveltyDetector`.

Samples

A container with the samples is transferred to `ipSamples`.

Degrees of novelty

A container with the calculated degrees of novelty is returned via the reference `ipNovelties`.

Application

The degree of novelty of several samples can be calculated as follows:

```
hr := F_VN_GetBatchNovelties(
    ipNoveltyDetector   := ipNoveltyDetector,
    ipSamples           := ipSamples,
    ipNovelties         := ipNovelties,
    hrPrev              := hr);
```

Related functions

- F_VN_GetSampleNovelty [▶ 1448]

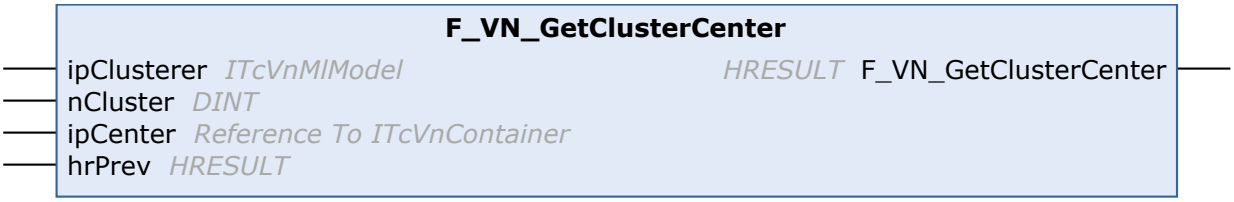
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
nCluster	DINT	Cluster ID of the cluster the center of which is requested
ipCenter	Reference To ITcVnContainer [▶ 370]	Returns the cluster center (ContainerType_Vector_REAL or ContainerType_Vector_LREAL depending on the floating point type internally used by the clusterer)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_GetClusterCenter` returns the center of a cluster.

Parameter

Clusterer model

The cluster whose center is to be calculated must be contained in the previously trained model `ipClusterer`.

Cluster ID

The ID of the corresponding cluster is transferred to `nCluster`.

Cluster center

The center of the cluster is returned via the reference `ipCenter`. The type of container returned is `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`, depending on the data type used internally in `ipClusterer`.

Application

The center of the third cluster in a Clusterer model can be determined as follows:

```
hr := F_VN_GetClusterCenter(
    ipClusterer := ipClusterer,
    nCluster    := 2,
    ipCenter    := ipCenter,
    hrPrev      := hr);
```

Related functions

- F_VN_GetClusterNum [▶ 1442]

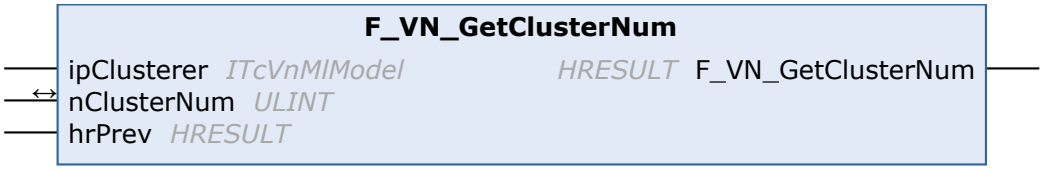
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_GetClusterNum` returns the number of clusters in a Clusterer model.

Parameter

Clusterer model

The previously trained Clusterer model must be transferred to `ipClusterer`.

Number of clusters

The number of clusters is returned via `nClusterNum`.

Application

For example, the number of clusters can be determined as follows:

```
hr := F_VN_GetClusterNum(
    ipClusterer   := ipClusterer,
    nClusterNum   := nClusterNum,
    hrPrev        := hr);
```

Related functions

- F_VN_GetClusterCenter [▶ 1440]

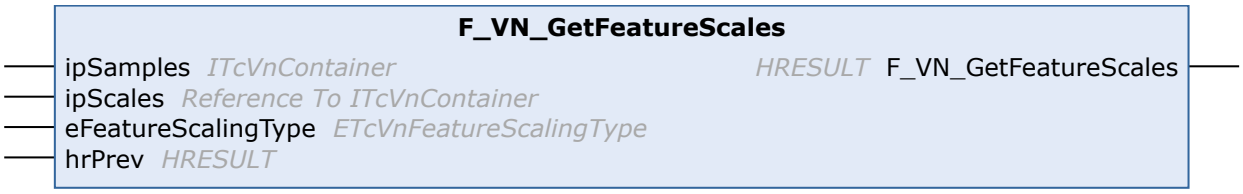
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container holding input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipScales	Reference To ITcVnContainer [▶ 370]	Returns a container with the scaling parameters for each feature (Vector_REAL or Vector_LREAL depending on the type of ipSamples).
eFeatureScalingType	ETcVnFeatureScalingType [▶ 202]	Feature scaling type
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function F_VN_GetFeatureScales calculates scaling parameters for each feature value of the samples transferred. The corresponding scaling can then be applied using the F_VN_FeatureScaling [▶ 1431] function for feature normalization. The same scaling can also be used with F_VN_InverseFeatureScaling [▶ 1449] to invert the scaling of the prediction results of a regression.

Parameter

Samples

The samples used to calculate the scaling parameters are transferred as containers to ipSamples. The number of feature values must be the same for each sample.

Scaling

The calculated scaling parameters are returned as a container via the reference ipScales. The number and order of the elements in the container depend on the scaling type, which is always found in the first element. With MAXABS there is always one scaling value per feature. For the other types, there are always two scaling values per feature.

Scaling type

The desired scaling type is transferred to eFeatureScalingType as the enum ETcVnFeatureScalingType [▶ 202]. The options are:

- TCVN_FST1_MAXABS: Uses the maximum of the absolute feature values to obtain a normalization in the range from -1 to 1. If all values are positive, normalization is performed from 0 to 1.
- TCVN_FST1_MINMAX: Uses the minimum and maximum values of the feature values to obtain a normalization in the range from 0 to 1.
- TCVN_FST1_STANDARDIZATION: Uses the mean value and the standard deviation of the feature values to obtain a normalization of the vector mean values equal to 0 and the vector standard deviation equal to 1.

Application

For example, the calculation of scaling parameters using the MINMAX method looks like this:

```
hr := F_VN_GetFeatureScales(
    ipSamples           := ipSamples,
    ipScales            := ipScales,
    eFeatureScalingType := TCVN_FST1_MINMAX,
    hrPrev              := hr);
```

If the number of feature values per sample in ipSamples is 12, for example, ipScales will have exactly 1 + 2 * 12 = 25 elements for this scaling type.

Related functions

- F_VN_FeatureScaling [▶ 1431]
- F_VN_InverseFeatureScaling [▶ 1449]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_GetSampleCluster` determines the best matching cluster for a sample and returns its ID.

Parameter

Clusterer model

A previously trained model must be transferred to `ipClusterer` for assignment to clusters.

Sample

A container with a sample is transferred to `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Cluster ID

The ID determined is returned via `nCluster`.

Expert parameters

The expert variant F_VN_GetSampleClusterExp [▶ 1446] contains additional parameters.

Application

The assignment of a sample to a cluster can be calculated as follows:

```
hr := F_VN_GetSampleCluster(
    ipClusterer := ipClusterer,
    ipSample    := ipSample,
    nCluster    := nCluster,
    hrPrev      := hr);
```

Related functions

- F_VN_GetBatchClusters [▶ 1436]

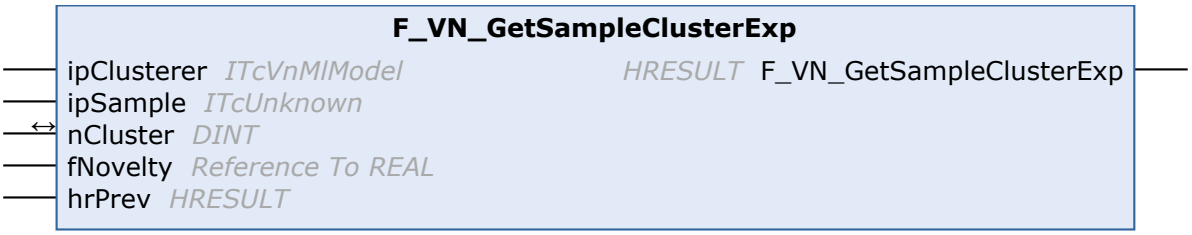
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSample	ITcUnknown [▶ 428]	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 [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_GetSampleClusterExp` is the expert variant of F_VN_GetSampleCluster [▶ 1445]. It contains additional parameters.

Parameter

Clusterer model

A previously trained model must be transferred to `ipClusterer` for assignment to clusters.

Sample

A container with a sample is transferred to `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Cluster ID

The ID determined is returned via `nCluster`.

Degree of novelty

The degree of novelty of the sample is returned via the reference `fNovelty`. The degree of novelty indicates how strongly a sample differs from the clusters. A degree of novelty of 0 indicates that a sample is known to the model.

Application

The calculation of the cluster and the degree of novelty of a sample can be calculated as follows:

```
hr := F_VN_GetSampleClusterExp(
    ipClusterer := ipClusterer,
    ipSample    := ipSample,
    nCluster    := nCluster,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Related functions

- F_VN_GetSampleNovelty [▶ 1448]

- F_VN_GetBatchClusters [▶ 1436]
- F_VN_GetBatchNovelties [▶ 1439]

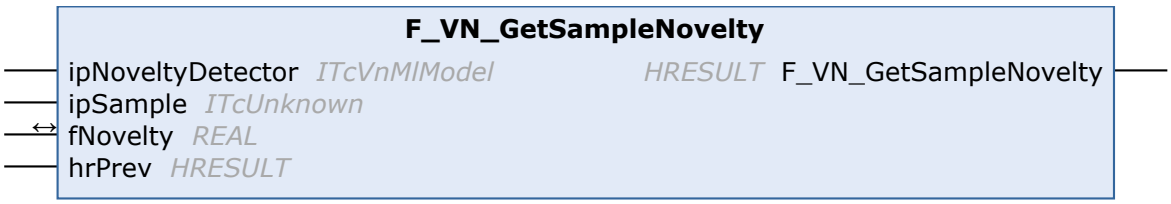
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Novelty detector to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_GetSampleNovelty` determines the degree of novelty for a sample in a container.

Parameter

Anomaly model

To calculate the degree of novelty, the previously trained model must be transferred to `ipNoveltyDetector`.

Sample

A container with a sample is transferred to `ipSample`

Degree of novelty

The degree of novelty is returned via the reference `fNovelty`. The degree of novelty indicates how much a sample differs from the learned data. A degree of novelty of 0 indicates that a sample is known to the model.

Application

The degree of novelty of a sample can be calculated as follows:

```
hr := F_VN_GetSampleNovelty(
    ipNoveltyDetector   := ipNoveltyDetector,
    ipSample            := ipSample,
    fNovelty            := fNovelty,
    hrPrev              := hr);
```

Related functions

- F_VN_GetBatchNovelties [▶ 1439]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_InverseFeatureScaling` inverts the scaling of one or more samples. The function F_VN_FeatureScaling [▶ 1431] can be used for scaling.

Parameter

Samples

One or more samples must be transferred as a container to `ipSamples`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled samples

The inversely scaled samples are returned via the reference `ipInverseSamples`.

Expert parameters

The expert variant F_VN_InverseFeatureScalingExp [▶ 1453] contains additional parameters.

Application

For example, an inverse feature scaling where the scaling was previously calculated using F_VN_FeatureScaling [▶ 1431] looks like this:

```
hr := F_VN_FeatureScaling(ipSamples, ipScales, ipSamplesScaled, hr);

hr := F_VN_InverseFeatureScaling(
    ipSamples       := ipSamplesScaled,
    ipScales        := ipScales,
    ipInverseSamples:= ipSamplesInvertedScaling,
    hrPrev          := hr);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_FeatureScaling [▶ 1431]
- F_VN_InverseFeatureScaling_REAL [▶ 1451]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_InverseFeatureScaling_REAL` inverts the scaling of a sample with only one feature or the prediction value of a scalar regressor. The function F_VN_FeatureScaling [▶ 1431] can be used for scaling.

Parameter

Samples

A sample of the type `REAL` must be transferred to `fSample`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled sample

The inversely scaled feature is returned via `fInverseSample`.

Expert parameters

The expert variant F_VN_InverseFeatureScalingExp_REAL [▶ 1454] contains additional parameters.

Application

For example, an inverse feature scaling for a sample looks like this:

```
hr := F_VN_InverseFeatureScaling_REAL(
    fSample        := fSample,
    ipScales       := ipScales,
    fInverseSample := fSampleInvertedScaling,
    hrPrev         := hr);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_FeatureScaling [▶ 1431]
- F_VN_InverseFeatureScaling [▶ 1449]

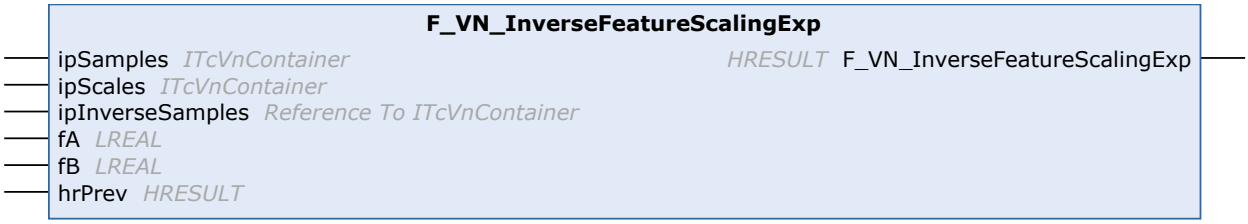
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 370]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_InverseFeatureScalingExp` is the expert variant of F_VN_InverseFeatureScaling [▶ 1449]. It contains additional parameters.

Parameter

Samples

One or more samples must be transferred as a container to `ipSamples`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled samples

The inversely scaled samples are returned via the reference `ipInverseSamples`.

Defining the output range

If the scaling type `TCVN_FST1_MINMAX` is used, the value range of the output values can also be specified via `fA` and `fB`.

Application

For example, an inverse feature scaling where the scaling was previously calculated by F_VN_FeatureScalingExp [▶ 1433] with an output value range of 0.1 to 0.9 looks like this:

```
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);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_FeatureScaling [▶ 1431]
- F_VN_InverseFeatureScaling_REAL [▶ 1451]

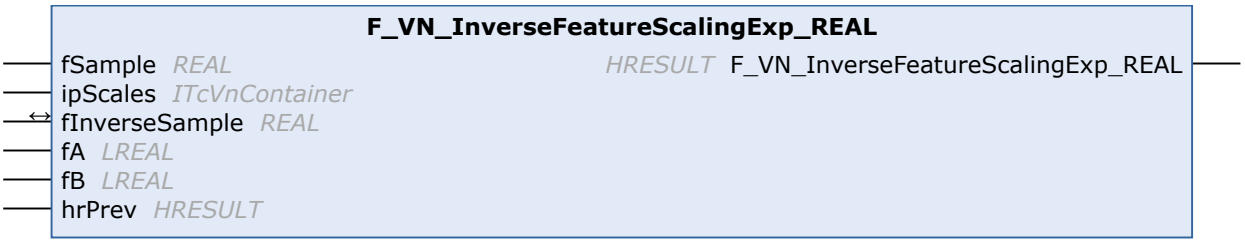
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]	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 [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_InverseFeatureScalingExp_REAL` is the expert variant of F_VN_InverseFeatureScaling_REAL [▶ 1451]. It contains additional parameters.

Parameter

Samples

A sample of the type `REAL` must be transferred to `fSample`.

Scaling

A container with the scaling type and the corresponding scaling parameters must be transferred to `ipScales`. The container can be calculated using the function F_VN_GetFeatureScales [▶ 1443] or created using the structure described there.

Scaled sample

The inversely scaled feature is returned via `fInverseSample`.

Defining the output range

If the scaling type `TCVN_FST1_MINMAX` is used, the value range of the output values can also be specified via `fA` and `fB`.

Application

For example, an inverse feature scaling for a sample looks like this:

```
hr := F_VN_InverseFeatureScalingExp_REAL(
    fSample        := fSample,
    ipScales       := ipScales,
    fInverseSample := fSampleInvertedScaling,
    fA             := 0.1,
    fB             := 0.9,
    hrPrev         := hr);
```

Related functions

- F_VN_GetFeatureScales [▶ 1443]
- F_VN_FeatureScaling [▶ 1431]
- F_VN_InverseFeatureScaling_REAL [▶ 1451]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Feature transform instance to be used
ipSamples	ITcUnknown [▶ 428]	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 [▶ 428]	Returns the transformed sample(s) using the same type like ipSamples
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_InverseFeatureTransform` inverts the feature transformation of one or more samples using a transformation model. The direct transformation can be carried out with the same model using the function F_VN_FeatureTransform [▶ 1434].

Parameter

Transformation model

A previously created model must be transferred to `ipFeatureTransform` for feature transformation.

Samples

A container with the samples to be transformed is transferred to `ipSamples`. The following container types are permitted:

- `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL` for a single sample
- `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL` for a container with several samples.

Transformed samples

A container with the transformed samples is returned via the pointer `pipTransformed`. The container type is taken from ipSamples.

Application

For example, an inverse feature transformation is carried out as follows:

```
hr := F_VN_InverseFeatureTransform(
    ipFeatureTransform  := ipMlModel,
    ipSamples           := ipSamplesTransformed,
    pipTransformed      := ADR(ipTransformed),
    hrPrev              := hr);
```

Related functions

- F_VN_FeatureTransform [▶ 1434]

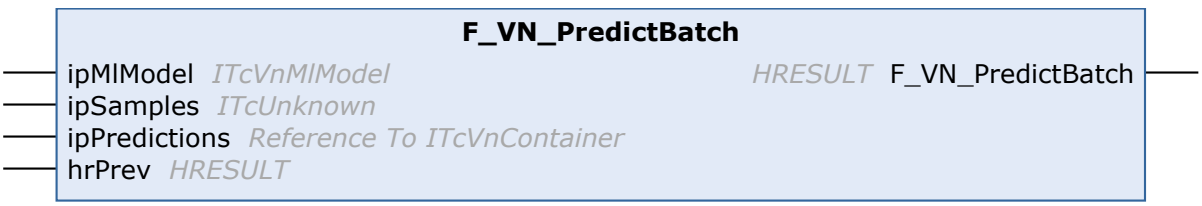
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier or regressor to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_PredictBatch` calculates the prediction of a model for several samples. Depending on the type of model, the predictions have a different data type (see `ipPredictions`).

Parameter

Model

For prediction, the previously trained model must be transferred to `ipMlModel`.

Samples

The samples are transferred to `ipSamples` in a container. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Predictions

The predictions are returned in a container via the reference `ipPredictions`. Depending on the type of model, the container has a different type:

- Class label for a classification: `ContainerType_Vector_DINT`
- Numerical prediction values in a scalar regression: `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`, depending on `ipSamples`
- Numerical prediction values in a vector regression: `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`, depending on `ipSamples`

Expert parameters

The expert variant F_VN_PredictBatchExp [▶ 1459] contains additional parameters.

Application

For example, several samples can be classified at the same time:

```
hr := F_VN_PredictBatch(
    ipMlModel       :=  ipMlModel,
    ipSamples       :=  ipSamples,
    ipPredictions   :=  ipPredictions,
    hrPrev          :=  hr);

hr := F_VN_GetAt_DINT(ipPredictions, nClassOfThirdSample, 2, hr);
```

The call of `F_VN_GetAt_DINT` shows how the class label of the 3rd sample can be get from the predictions.

Related functions

- F_VN_PredictSampleClass [▶ 1461]
- F_VN_PredictSampleScalar [▶ 1464]
- F_VN_PredictSampleVector [▶ 1467]

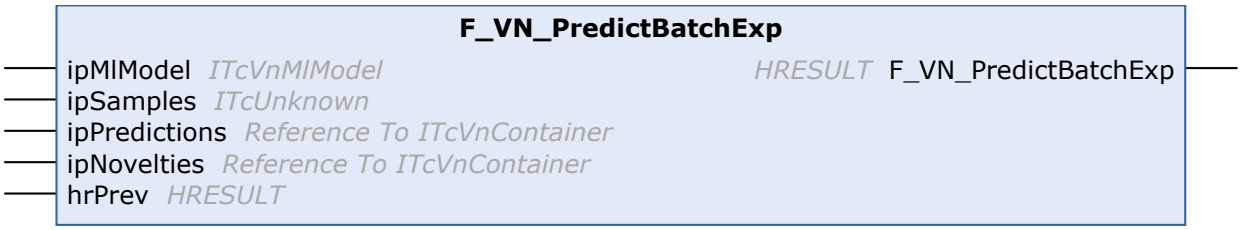
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier or regressor to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_PredictBatchExp` is the expert variant of F_VN_PredictBatch [▶ 1457]. It contains additional parameters.

Parameter

Model

For prediction, the previously trained model must be transferred to `ipMlModel`.

Samples

The samples are transferred to `ipSamples` in a container. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Predictions

The predictions are returned in a container via the reference `ipPredictions`. Depending on the type of model, the container has a different type:

- Class label for a classification: `ContainerType_Vector_DINT`
- Numerical prediction values in a scalar regression: `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`, depending on `ipSamples`
- Numerical prediction values in a vector regression: `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`, depending on `ipSamples`

Degrees of novelty

The degrees of novelty of the samples are returned as containers via the reference `ipNovelties`.

Application

For example, several samples can be classified at the same time:

```
hr := F_VN_PredictBatchExp(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
```

```
    ipPredictions   := ipPredictions,
    ipNovelties     := ipNovelties,
    hrPrev          := hr);

hr := F_VN_GetAt_DINT(ipPredictions, nClassOfThirdSample, 2, hr);
```

The call of `F_VN_GetAt_DINT` shows how the class label of the 3rd sample can be get from the predictions.

Related functions

- F_VN_PredictSampleClass [▶ 1461]
- F_VN_PredictSampleScalar [▶ 1464]
- F_VN_PredictSampleVector [▶ 1467]
- F_VN_GetBatchNovelties [▶ 1448]

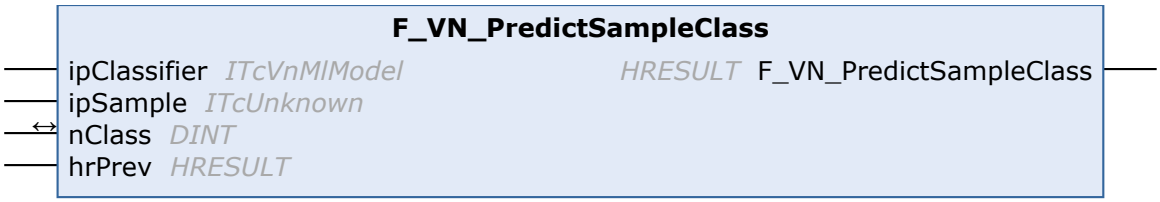
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_PredictSampleClass` classifies a sample based on a classification model.

Parameter

Classification model

The previously trained classification model must be transferred to `ipClassifier`.

Sample

The samples are transferred to `ipSample` in a container. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Class

The class of the sample is returned as the classification result via `nClass`.

Expert parameters

The expert variant F_VN_PredictSampleClassExp [▶ 1463] contains additional parameters.

Application

For example, a sample can be classified as follows:

```
hr := F_VN_PredictSampleClass(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClassResult,
    hrPrev          := hr);
```

Related functions

- F_VN_PredictBatch [▶ 1457]

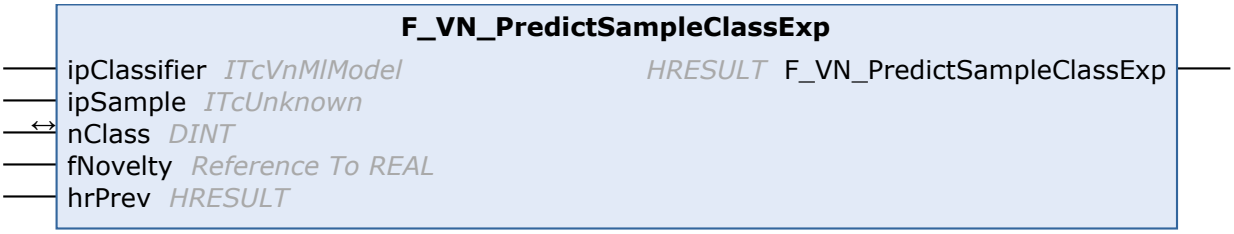
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier to be used
ipSample	ITcUnknown [▶ 428]	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 [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_PredictSampleClassExp` is the expert variant of F_VN_PredictSampleClass [▶ 1461]. It contains additional parameters.

Parameter

Classification model

The previously trained classification model must be transferred to `ipClassifier`.

Sample

The samples are transferred to `ipSample` in a container. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Class

The class of the sample is returned as the classification result via `nClass`.

Degree of novelty

The degree of novelty of the sample is returned via `fNovelty`.

Application

For example, a sample can be classified as follows:

```
hr := F_VN_PredictSampleClassExp(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClassResult,
    fNovelty        := fNovelty,
    hrPrev          := hr);
```

Related functions

- F_VN_GetSampleNovelty [▶ 1448]
- F_VN_PredictBatch [▶ 1457]

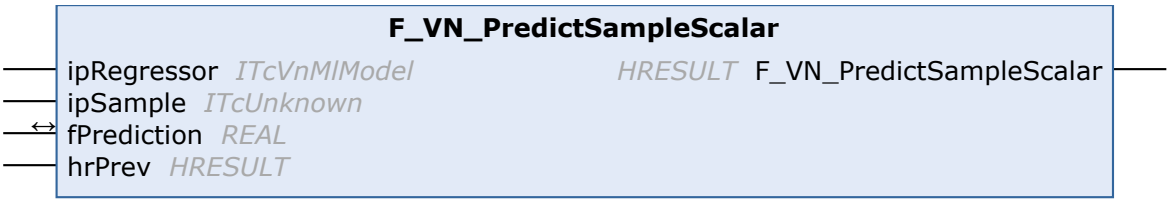
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_PredictSampleScalar` calculates the scalar prediction value of a sample based on a regression model.

Parameter

Regression model

The previously trained regression model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Prediction

The calculated prediction value is returned via `fPrediction`.

Expert parameters

The expert variant F_VN_PredictSampleScalarExp [▶ 1466] contains additional parameters.

Application

For example, the prediction value of a sample can be calculated as follows:

```
hr := F_VN_PredictSampleScalar(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    fPrediction := fPrediction,
    hrPrev      := hr);
```

Related functions

- F_VN_PredictBatch [▶ 1457]

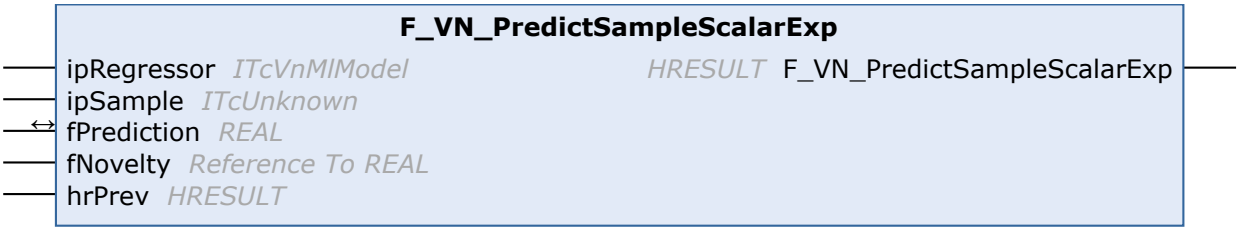
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	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 [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_PredictSampleScalarExp` is the expert variant of F_VN_PredictSampleScalar [▶ 1464]. It contains additional parameters.

Parameter

Regression model

The previously trained regression model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Prediction

The calculated prediction value is returned via `fPrediction`.

Degree of novelty

The degree of novelty of the sample is returned via `fNovelty`.

Application

For example, the regression value of a sample can be calculated as follows:

```
hr := F_VN_PredictSampleScalarExp(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    fPrediction := fPrediction,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Related functions

- F_VN_PredictBatch [▶ 1457]
- F_VN_GetBatchNovelties [▶ 1448]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	Reference To ITcVnContainer [▶ 370]	Returns the predicted output (ContainerType_Vector_REAL or ContainerType_Vector_LREAL, depending on ipSample)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_PredictSampleVector` calculates the prediction vector of a sample based on a regression model.

Parameter

Regression model

The previously trained regression model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Prediction

The calculated prediction vector is returned as a container via the reference `ipPrediction`. The container type is taken from `ipSamples`.

Expert parameters

The expert variant F_VN_PredictSampleVectorExp [▶ 1469] contains additional parameters.

Application

For example, the prediction vector of a sample can be calculated as follows:

```
hr := F_VN_PredictSampleVector(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    ipPrediction:= ipPrediction,
    hrPrev      := hr);
```

Related functions

- F_VN_PredictBatch [▶ 1457]

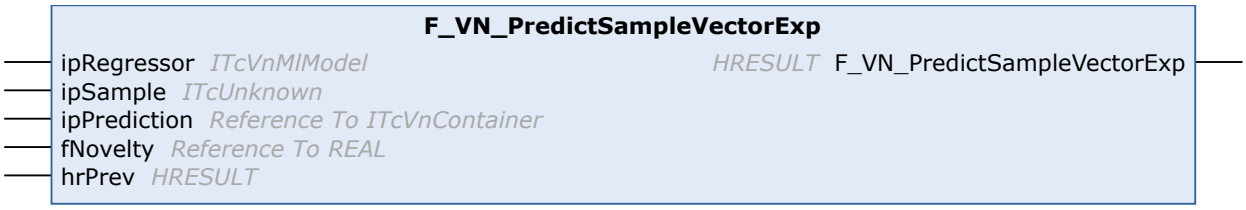
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_PredictSampleVectorExp` is the expert variant of F_VN_PredictSampleVector [▶ 1467]. It contains additional parameters.

Parameter

Regression model

The previously trained regression model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The container type must be either `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Prediction

The calculated prediction vector is returned as a container via the reference `ipPrediction`. The container type is taken from `ipSamples`.

Degree of novelty

The degree of novelty of the sample is returned via `fNovelty`.

Application

For example, the prediction vector of a sample can be calculated as follows:

```
hr := F_VN_PredictSampleVectorExp(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    ipPrediction:= ipPrediction,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Related functions

- F_VN_PredictBatch [▶ 1457]
- F_VN_GetBatchNovelties [▶ 1448]

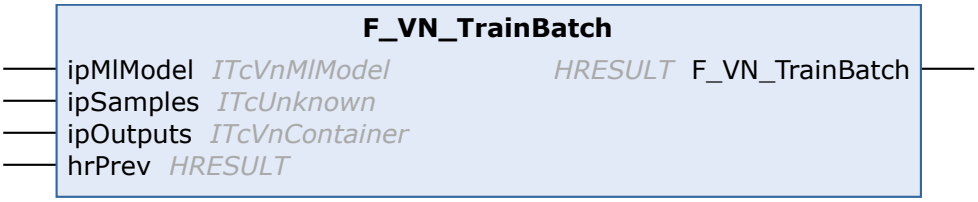
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier, clusterer, novelty detector, or regressor to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipOutputs	ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainBatch` trains a model with several samples. Depending on the type of model, the result values must have a different type (see `ipOutputs`). Classifier, clusterer, novelty detector and regressor are supported as model types.

Parameter

Model

The previously created model must be transferred to `ipMlModel`.

Samples

The samples are transferred to `ipSamples` in a container. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Results

The known results for the samples are transferred to `ipOutputs` in a container. Depending on the type of model, the container must have a different type:

- Class label for a classification: `ContainerType_Vector_DINT`
- Numerical result values for a scalar regression: `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`
- Numerical result values for a vector regression: `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`

If it is a Clusterer model or a novelty detector, it must be `ipOutputs := 0`.

Application

For example, an ML model for classification can be trained like this:

```
hr := F_VN_TrainBatch(
    ipMlModel    := ipMlModel,
    ipSamples    := ipSamples,
    ipOutputs    := ipClassLabels,
    hrPrev       := hr);
```

Related functions

- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

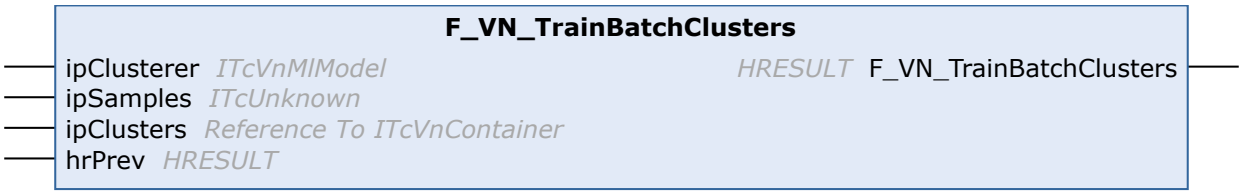
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSamples	ITcUnknown [▶ 428]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 370]	Returns the IDs of the clusters the samples have been assigned to (ContainerType_Vector_DINT; optional, set to 0 if not required)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainBatchClusters` trains a Clusterer model with several samples. Compared to F_VN_TrainBatch [▶ 1470], the assigned clusters can also be returned for each sample.

Parameter

Clusterer model

The previously created Clusterer model must be transferred to `ipClusterer`.

Samples

The samples are transferred to `ipSamples` in a container. The type of container must be `ContainerType_Vector_Vector_REAL` or `ContainerType_Vector_Vector_LREAL`.

Clusters

The IDs of the assigned clusters are returned in a container of the type `ContainerType_Vector_DINT` via the reference `ipClusters`.

Application

For example, a Clusterer model can be trained like this:

```
hr := F_VN_TrainBatchClusters(
    ipClusterer := ipClusterModel,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    hrPrev      := hr);

hr := F_VN_GetAt_DINT(ipClusters, nClusterOfThirdSample, 2, hr);
```

The call of `F_VN_GetAt_DINT` shows how the cluster assignment of the 3rd sample can be queried directly after training.

Related functions

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

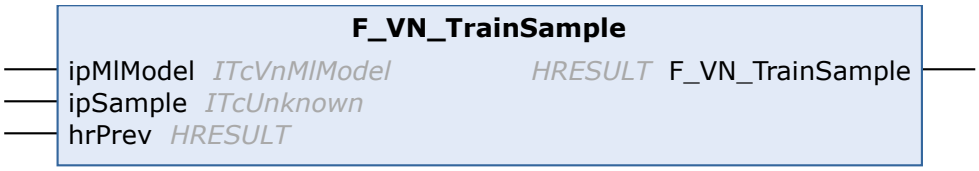
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer or novelty detector to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainSample` trains a clusterer or novelty detection model with a sample.

Parameter

Model

The previously created model must be transferred to `ipMlModel`.

Sample

The sample container is transferred to `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Application

For example, an ML model can be trained with a sample as follows:

```
hr := F_VN_TrainSample(
    ipMlModel   := ipMlModel,
    ipSample    := ipSample,
    hrPrev      := hr);
```

Related functions

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Classifier to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nClass	DINT	Class label to be learnt
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainSampleClass` trains a classifier model with a sample.

Parameter

Classifier model

The previously created classifier model must be transferred to `ipClassifier`.

Sample

The sample container is transferred to `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Class

The target class for the sample is transferred as `nClass`.

Application

For example, a classifier model can be trained with single samples as follows:

```
hr := F_VN_TrainSampleClass(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClass,
    hrPrev          := hr);
```

Related functions

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Clusterer to be used
ipSample	ITcUnknown [▶ 428]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainSampleCluster` trains a clusterer model with a sample and returns the corresponding cluster ID of the sample.

Parameter

Clusterer model

The previously created Clusterer model must be transferred to `ipClusterer`.

Sample

The sample container is transferred as `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Cluster

The cluster ID associated with the sample is returned via `nCluster`.

Application

For example, a Clusterer model can be trained with a single sample as follows:

```
hr := F_VN_TrainSampleCluster(
    ipClusterer     := ipClusterModel,
    ipSample        := ipSample,
    nCluster        := nCluster,
    hrPrev          := hr);
```

Related functions

- F_VN_TrainBatch [▶ 1470]

- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleScalar [▶ 1478]
- F_VN_TrainSampleVector [▶ 1479]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fOutput	REAL	Scalar output to be learnt
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainSampleScalar` trains a scalar regressor model with a sample.

Parameter

Regressor model

The previously created regressor model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Training value

The result value of the sample is transferred to `fOutput`.

Application

For example, a scalar regressor model can be trained with a single sample as follows:

```
hr := F_VN_TrainSampleScalar(
    ipRegressor     := ipRegressorModel,
    ipSample        := ipSample,
    fOutput         := fOutput,
    hrPrev          := hr);
```

Related functions

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleVector [▶ 1479]

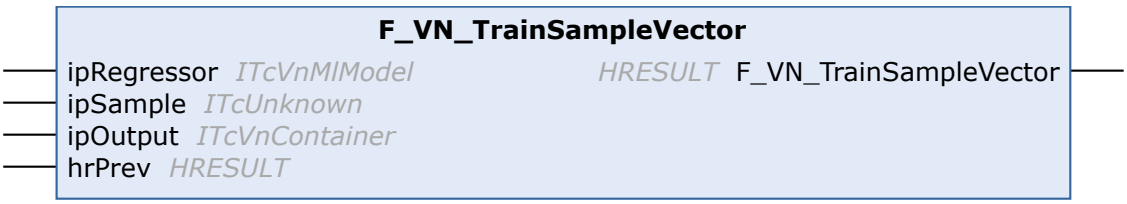
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]	Regressor to be used
ipSample	ITcUnknown [▶ 428]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipOutput	ITcVnContainer [▶ 370]	Vectorial output to be learnt (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_TrainSampleVector` trains a vector regressor model with a sample.

Parameter

Regressor model

The previously created regressor model must be transferred to `ipRegressor`.

Sample

The sample container is transferred as `ipSample`. The type of container must be `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`.

Training vector

The result vector of the sample is transferred to `ipOutput`.

Application

For example, a vector regressor model can be trained with a single sample as follows:

```
hr := F_VN_TrainSampleVector(
    ipRegressor     := ipRegressorModel,
    ipSample        := ipSample,
    ipOutput        := ipVector,
    hrPrev          := hr);
```

Related functions

- F_VN_TrainBatch [▶ 1470]
- F_VN_TrainBatchClusters [▶ 1472]
- F_VN_TrainSample [▶ 1474]
- F_VN_TrainSampleClass [▶ 1475]
- F_VN_TrainSampleCluster [▶ 1476]
- F_VN_TrainSampleScalar [▶ 1478]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20 Measurement

The group contains functions for the measurement of objects in images.

Functions

Edge localization

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

Geometrical features

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

Miscellaneous

- F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1486] for aligning the search field to an edge

Parameters for edge localization

The borders of an image object often don't lie exactly on the border between two pixels. Instead, they run independently of pixel grids. Therefore edge localization with sub-pixel accuracy is necessary. This not only considers the transition between two pixels, but also determines the edge position within a pixel on the basis of the overall intensity gradient of an object border.

NOTICE

Don't process the image before measuring

Don't process the image before calling the measurement function. Processing can lead to loss of information and lower accuracy accordingly.

This basic functionality of edge localization is used by all measurement functions. The functions differ mainly by the geometric arrangement in which they perform the edge localization (e.g. F_VN_LocateEllipse [▶ 1507] for round objects) and the properties that you calculate from them (e.g. F_VN_MeasureAngleBetweenEdges [▶ 1513] for angle measurement).

Certain parameters must be defined for each function that uses edge localization. These include:

- Definition of the search area: where is the edge sought?
- Definition of edge contrast: What is detected as an edge?
- Definition of the measurement algorithm: How to determine the exact position of the edge.

These parameters are described below. In the case of expert parameters, standard values are specified in addition.

Search area

Parameter	Description
Position of the search lines	The position of the search lines is defined according to the feature to be localized. The position of search lines for a linear edge is defined as a single line with the start point `aStartPoint` and end point `aEndPoint`. The position of search lines for an ellipse is defined by a center point `aCenter` and a radius `fSearchRadius`. The position of search lines for angles is defined by an inner point `aInnerPoint` and two outer points `aOuterPoint1` and `aOuterPoint2`. The position of the search lines should always be set such that the search lines cut the edges to be localized orthogonally. A rectangular search area can be adjusted orthogonally to an edge with the function F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1486].
Number of search lines `nSearchLines : UDINT`	This specifies how many search lines are to be used for the edge localization. The simplest edge localization has one search line on which the edge must be found. Depending on the geometric feature to be determined, multiple search lines are distributed locally so that several edge transitions are localized. This can be necessary in the following cases: • Not only the position, but also the gradient or the rotation of the edge are required. • The geometrical feature (e.g. ellipse) necessitates several search lines. • An average value is required. When localizing linear edges, the search lines are arranged symmetrically on both sides of the central search line, the central search line being included in `nSearchLines`. Thus, an odd number of search lines is always required.
Distance between the search lines `fSearchlineDist : REAL` *(Expert)* Default: `1`	The search lines distance defines the distance between neighboring search lines in pixels (> 0, default 1). This allows flexible balancing between accuracy and computing time.
Inversion of the search direction `bInvertSearchDirection : BOOL` *(Expert)* Default: `FALSE`	The localization algorithm always searches for the edge from one end of the search line to the other. e.g. searching takes place from aStartPoint to aEndPoint. If it is necessary to invert this search direction due to the image properties (i.e. to start from the other end of the search lines), this parameter helps.

Edge contrast

Parameter	Description
Edge direction `eEdgeDirection : ETcVnEdgeDirection`	The edge direction determines whether an edge transition from light to dark or from dark to light should be found. The following values are available for this: • `TCVN_ED_DARK_TO_LIGHT` • `TCVN_ED_LIGHT_TO_DARK`
Minimum strength `fMinStrength : LREAL`	The minimum strength defines the minimum difference in intensity required within an edge for it to be found. This makes it possible to control how much contrast an edge must have in order to be found.
Maximum thickness `nMaxThickness : UDINT` *(Expert)* Default: `10`	The maximum thickness specifies the number of pixels within which `fMinStrength` must be reached. This allows you to control whether only sharp edges or also very blurred edges are to be found.

Algorithm

Parameter	Description
Algorithm `eAlgorithm : ETcVnEdgeDetectionAlgorithm` *(Expert)* Default: `TCVN_EDA_INTERPOLATION`	This parameter defines the localization algorithm used. The available options are: • `TCVN_EDA_INTERPOLATION` • `TCVN_EDA_APPROX_ERF` • `TCVN_EDA_APPROX_GAUSSIAN` `TCVN_EDA_APPROX_ERF` and `TCVN_EDA_APPROX_GAUSSIAN` are slower than the standard algorithm `TCVN_EDA_INTERPOLATION`, but can achieve higher accuracy. The prerequisite for this, however, is that the intensity gradient of the edge also corresponds as well as possible to the specified model. The error function `erf` or the Gaussian function is used as a model. The `erf` function is well suited for edges if the transition from bright to dark extends over several pixels and remains approximately constant on both sides of the light or dark area respectively.  A Gaussian function is well suited for lines, i.e. a dark-light-dark transition or vice versa. However, the brightness plateau may only be a few pixels wide, otherwise the erf function should be used. 
Subpixels / Iterations `nSubpixelsIterations : UDINT` *(Expert)* Default: 10	The meaning of this parameter depends on the localization algorithm used. At `TCVN_EDA_INTERPOLATION`: the number of subpixels considered defines the fine granularity with which the search line evaluates the underlying pixel intensities. A value of 10, for example, specifies that 10 measured values are taken over a length of 1 pixel. Recommended: `5` - `10` At `TCVN_EDA_APPROX_ERF` and `TCVN_EDA_APPROX_GAUSSIAN`: the number of iterations is an abort criterion that defines the maximum number of iterations with which the edge model is to be approximated for the pixel intensities. More iterations increase the runtime; if this is too high the interpolation method should be used. Recommended: `50` - `100`
Approximation accuracy `fApproxPrecision : REAL` *(Expert)* Default: `0.001`	The approximation accuracy is a further abort criterion of the edge localization. It is only observed if the localization algorithm is `TCVN_EDA_APPROX_ERF` or `TCVN_EDA_APPROX_GAUSSIAN`. The approximation of the edge or line is aborted if either the number of iterations of `nSubPixelIterations` is reached or the relative

Parameter	Description
	deviation in pixels is less than `fApproximationPrecision` for two consecutive iterations. Recommended: `0.01` - `0.0001`

Interpretation of the HRESULT

The HRESULT [▶ 141] return values have the following meaning:

Code	Name	Description
16#000	S_OK	Function was successfully executed and an edge was located on all search lines.
16#001	S_FALSE	Function was successfully executed, but edges were not located on each search line.
16#256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. In this case, all measurement functions can return partial results.
16#70C	NOTFOUND	Function was executed successfully, but no edge could be found on any search line. It is assumed that a measurement function is only called if it is ensured that the image contains an object with the parameterized edge properties. If none of the search lines are found, this indicates incorrect parameterization or an unsuitable call of the function. Therefore, an error code is output in this case.
16#7xx	All error codes except NOTFOUND	Function was not executed successfully. See also: ADS Return Codes [▶ 2879]

The following query, for example, can be used for the processing of the return value:

```
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
```

Reducing execution time

The execution time for a fixed search window varies slightly depending on the position of the component in the image. If the edge that is found is closer to the starting point, the execution time is shorter than for an edge further away.

A larger fluctuation can occur with the two approximation algorithms, since these require different iterations in order to approximate the model parameters using the surrounding pixel intensities.

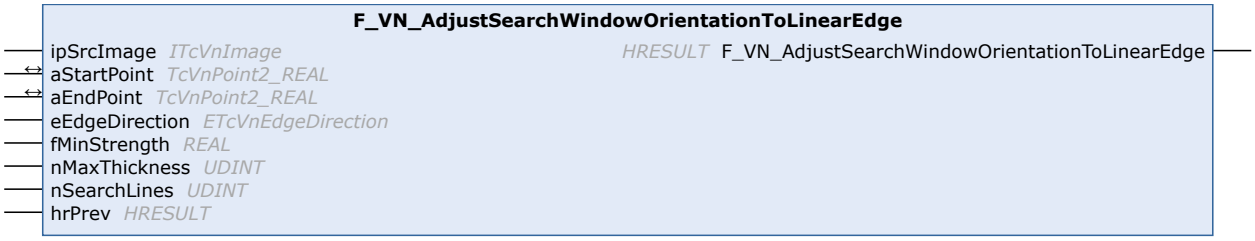
In general, fewer iterations are required if the search lines of the search window hit the edge as orthogonally as possible. If the position and orientation of the objects in the image are subject to little or no fluctuation, the search window can be adjusted accordingly to achieve shorter execution times. In addition, the maximum time required can be reduced by reducing the maximum number of iterations, but this may lead to less accurate results.

In addition, the time can be limited with an external watchdog, which terminates the execution of the function if necessary and returns the existing partial results.

Samples

Measurement [▶ 2813]

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 [▶ 411]	Source image
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Position from which to start the search process (in the direction of aEndPoint), which is not changed
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends, which is adjusted by this function

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_AdjustSearchWindowOrientationToLinearEdge` rotates the search field for measurement functions in such a way that it lies orthogonal to a linear edge.

Parameter

Input image

The input image `ipSrcImage` must contain an edge in accordance with the parameters for edge detection.

Search field (Return value)

The search field to be adjusted is transferred as a reference so that it can be adjusted. The search field consists of the central search line, which consists of the start point `aStartPoint` and end point `aEndPoint`, as well as the number of search lines `nSearchLines`.

Following the function call, `aStartPoint` and `aEndPoint` contain the adjusted central search line. This is now orthogonal to the linear edge on the input image `ipSrcImage`.

Parameters for edge localization

The parameters `eEdgeDirection`, `fMinStrength` and `nMaxThickness` determine the search features for the linear edge in the input image `ipSrcImage`. These work analogously to the edge localization in F_VN_LocateEdge [▶ 1495].

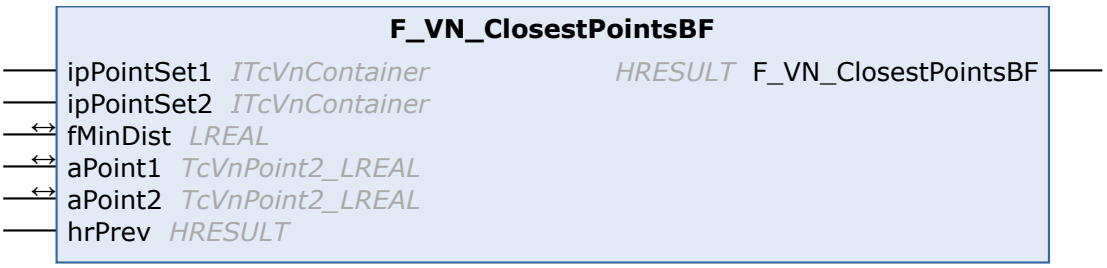
Required License

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 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 [▶ 370]	Container with 1st set of 2d points
ipPointSet2	ITcVnContainer [▶ 370]	Container with 2nd set of 2d points
hrPrev	HRESULT [▶ 141]	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 [▶ 160]	Returns the point out of ipPointSet1, for which fMinDist is achieved
aPoint2	TcVnPoint2_LREAL [▶ 160]	Returns the point out of ipPointSet2, for which fMinDist is achieved

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_ClosestPointsBF` determines which points from two point clouds are closest to each other.

Parameter

Point clouds

The closest two points are sought in the point clouds `ipPointSet1` and `ipPointSet2`.

Minimum distance (Return value)

The parameter `fMinDist` returns the distance between the two points that are closest to each other. This corresponds to the minimum distance between the two point clouds.

Closest points (Return value)

The parameters `aPoint1` and `aPoint2` return the two points that are closest to each other in the point clouds.

Application

The determination of the minimum distance between two lines looks like this, for example:

```
hr := F_VN_ClosestPointsBF(
    ipPointSet1 :=  ipLine1,
    ipPointSet2 :=  ipLine2,
    fMinDist    :=  fDistance,
    aPoint1     :=  aPoint1,
    aPoint2     :=  aPoint2,
    hrPrev      :=  hr
);
```

Related functions

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

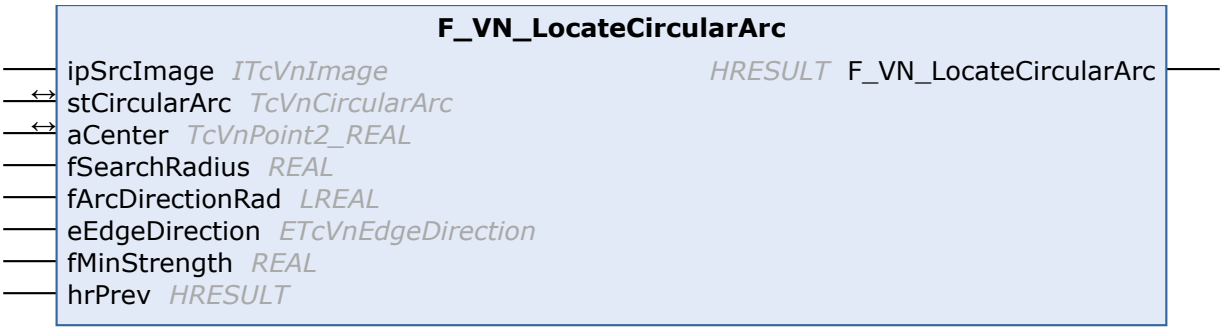
Required License

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 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 [▶ 411]	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 [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 227]	Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 160]	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 [▶ 141]

Further information

The function `F_VN_LocateCircularArc` localizes and measures a proportional circular arc. It is based on the edge localization [▶ 1481] and uses normal parameters. It returns the localized circular arc as a structure of the type TcVnCircularArc (center point, radius, angle range).

If full circles are to be measured instead of a circular arc, use the function F_VN_LocateEllipse [▶ 1507] instead.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Circular arc (Return value)

The parameter `stCircularArc` returns the localized circular arc as a structure of the type TcVnCircularArc [▶ 227].

Estimated center point

The estimated center point `aCenter` of the circular arc must be specified in order to define the position of the search lines.

If the position of the center point of the circle is unknown, it may need to be determined beforehand by means of an object detection [▶ 1176].

Search radius

The search radius `fSearchRadius` defines the length of the search lines from the center point of the circle in pixels. The search radius should be defined somewhat larger than the actual radius of the circle so that the edge localization can be carried out unambiguously.

Search direction

The search direction `fArcDirectionRad` specifies the circular direction in which the edge search should begin (starting from the center point of the circle). The value is to be specified in radians and has a value range of [-π, π]. Right corresponds to 0, top with –π/2 and bottom with π/2. Starting from `fArcDirectionRad`, the circular arc must be present at least in the range of +- 0.4 in order to be detected. It is recommended to aim for the center of the circular arc.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_LocateCircularArcExp [▶ 1492] contains additional parameters.

Application

The localization and measurement of a proportional circular arc with a radius of `270px` and a search angle of `120°` looks like this, for example:

```
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
);
```

The properties of the localized circular arc can then be retrieved via the variables `stArc.aCenter`, `stArc.fRadius`, `stArc.fStartAngle` and `stArc.fEndAngle`. In addition, the circular arc can be visualized with the function F_VN_DrawCircularArc [▶ 1004]:

```
F_VN_DrawCircularArc(stArc, ipImageRes, aColorGreen, 5, hr);
```

Samples

- Locate Circular Arc [▶ 2813]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

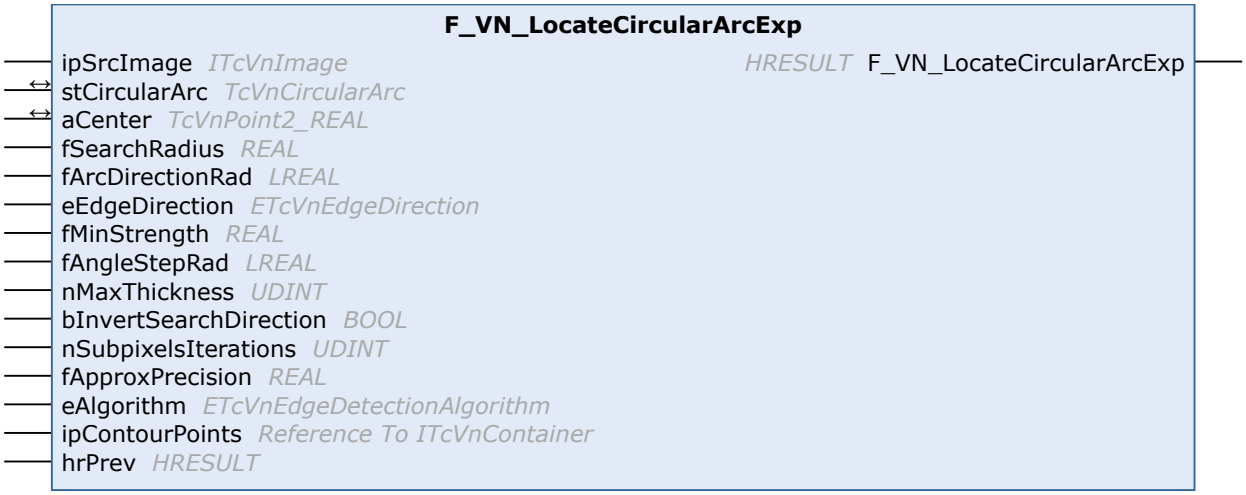
Required License

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 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 [▶ 411]	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 [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
ipContourPoints	Reference To ITcVnContainer [▶ 370]	Returns the subpixel accurate contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 227]	Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 160]	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 [▶ 141]

Further information

The function `F_VN_LocateCircularArcExp` is the expert version of F_VN_LocateCircularArc [▶ 1489]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Circular arc (Return value)

The parameter `stCircularArc` returns the localized circular arc as a structure of the type TcVnCircularArc [▶ 227].

Estimated center point

The estimated center point `aCenter` of the circular arc must be specified in order to define the position of the search lines.

If the position of the center point of the circle is unknown, it may need to be determined beforehand by means of an object detection [▶ 1176].

Search radius

The search radius `fSearchRadius` defines the length of the search lines from the center point of the circle in pixels. The search radius should be defined somewhat larger than the actual radius of the circle so that the edge localization can be carried out unambiguously.

Search direction

The search direction `fArcDirectionRad` specifies the circular direction in which the edge search should begin (starting from the center point of the circle). The value is to be specified in radians and has a value range of [-π, π]. Right corresponds to 0, top with –π/2 and bottom with π/2. Starting from `fArcDirectionRad`, the circular arc must be present at least in the range of +- 0.4 in order to be detected. It is recommended to aim for the center of the circular arc.

Radial search line distance (Expert)

The radial search line distance `fAngleStepRad` defines how large the angle steps between the search lines are. It is specified in radians. The distance should be selected so that the complete circular arc is divided into about 20 – 60 steps. In most cases `0.1 rad (5.7°)` is a suitable value.

Search direction inversion (Expert)

The search direction can be inverted with the Boolean `bInvertSearchDirection`. In this case, that means the following:

- `FALSE`: Search from the center point of the circle to outside the circular arc.
- `TRUE`: Search from outside the circular arc to the center point of the circle

Contour points (Expert, return value)

The container `ipContourPoints` returns the edge points of the circular arc found and is therefore of the type `ContainerType_Vector_TcVnPoint2_REAL`. The container contains one point for each search line successfully found.

The parameter can be set to `0` if the return of the edge points is not required.

Circular arc stCircularArc

The circular arc stCircularArc is approximated from the set of these points.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The localization of a circular arc with expert parameters looks like this, for example:

```
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
);
```

The localized points can be visualized with the function F_VN_DrawContours [▶ 1009].

```
hr := F_VN_DrawContours(ipEllipseContour, -1, ipImageRes, aColorGreen, 5, hr);
```

Samples

- Locate Circular Arc [▶ 2813]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

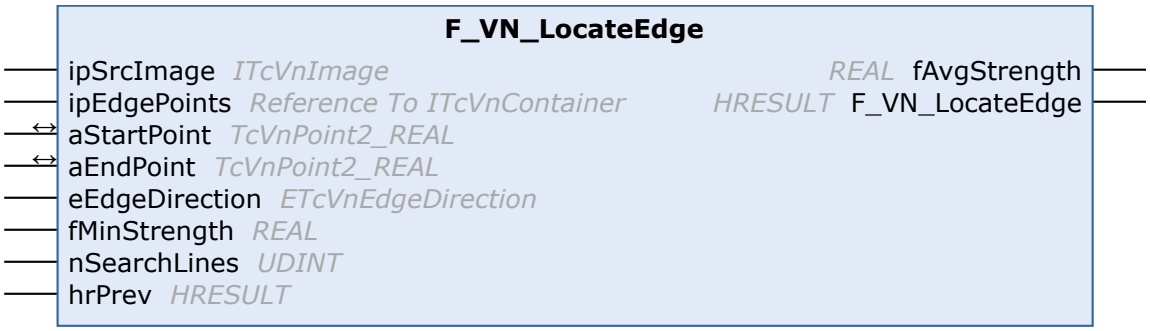
Required License

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 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 [▶ 411]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edge

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocateEdge` localizes an edge in a defined search window.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Localized points (Return value)

The container `ipEdgePoints` returns the localized points of the linear edge.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

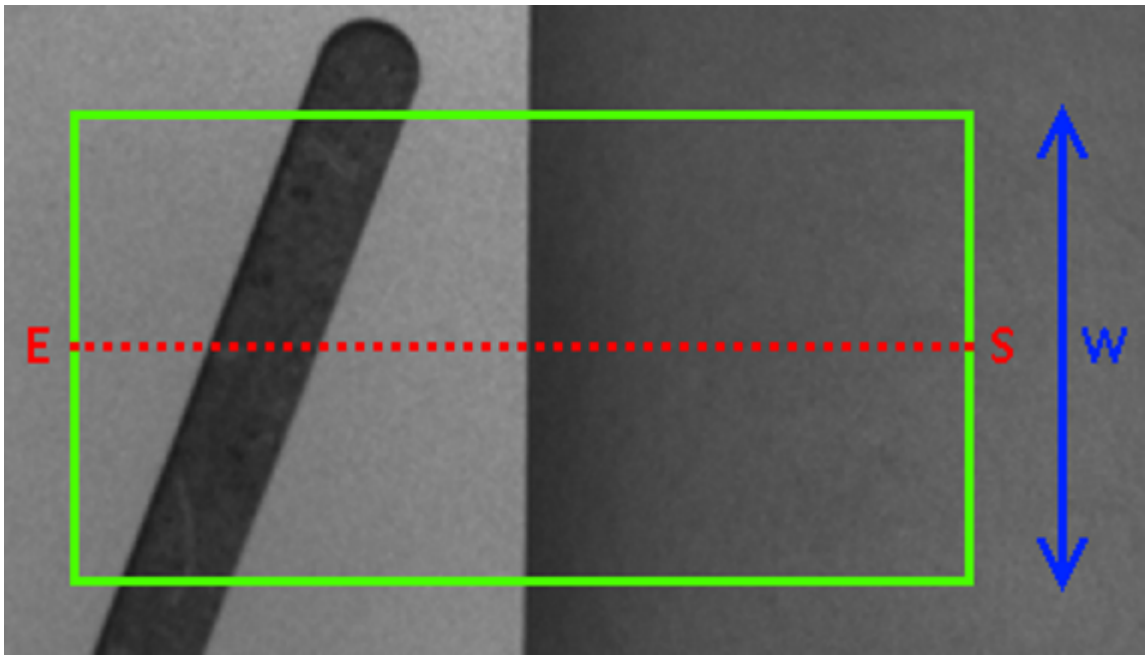


Fig. 15: S: StartPoint, E: EndPoint, W: WindowWidth

Edge strength (Return value)

The return value `fAvgStrength` displays the actual contrast of the localized edge (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_LocateEdgeExp contains additional parameters.

Application

The localization of a linear edge looks like this, for example:

```
hr := F_VN_LocateEdge(
    ipSrcImage              :=  ipImageIn,
    ipEdgePoints            :=  ipEdgePoints,
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    eEdgeDirection          :=  eDirection,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
    hrPrev                  :=  hr,
    fAvgStrength            =>  fAvgStrength
);
```

The localized edge points can be visualized with the function F_VN_DrawContours [▶ 1009].

```
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Samples

- Locate Edge [▶ 2817]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

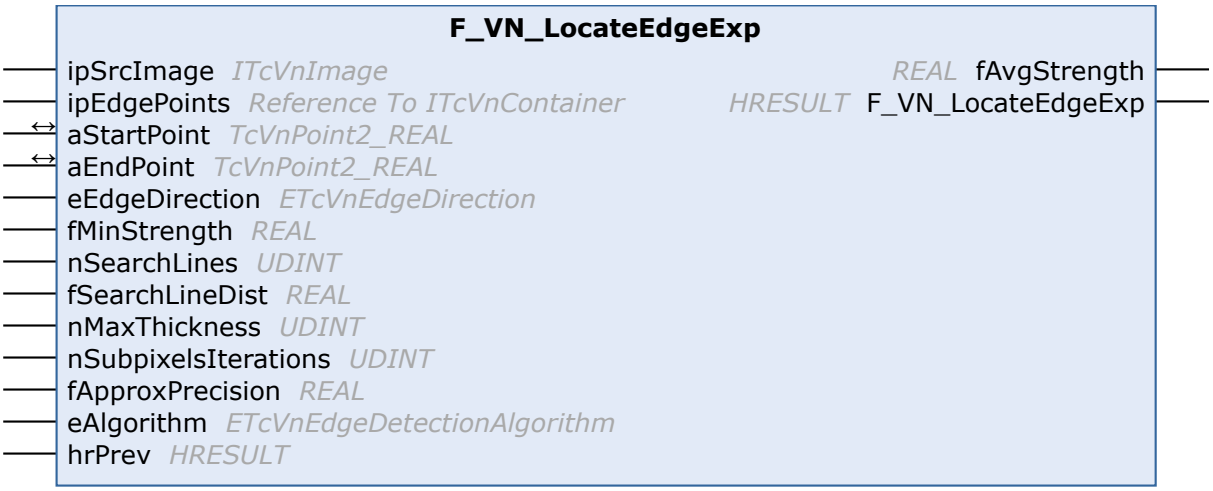
Required License

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 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 [▶ 411]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edge

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocateEdgeExp` is the expert version of F_VN_LocateEdge [▶ 1495]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Localized points (Return value)

The container `ipEdgePoints` returns the localized points of the linear edge.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

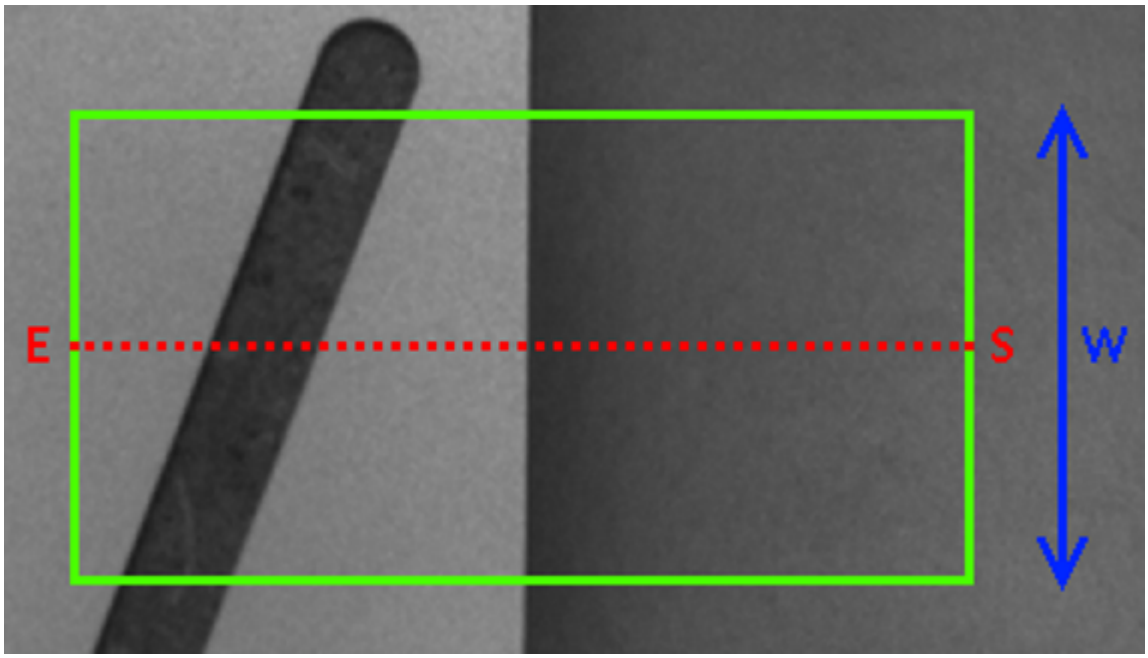


Fig. 16: S: StartPoint, E: EndPoint, W: WindowWidth

Edge strength (Return value)

The return value `fAvgStrength` displays the actual contrast of the localized edge (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The localization of a linear edge with expert parameters looks like this, for example:

```
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
);
```

Samples

- Locate Edge [▶ 2817]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

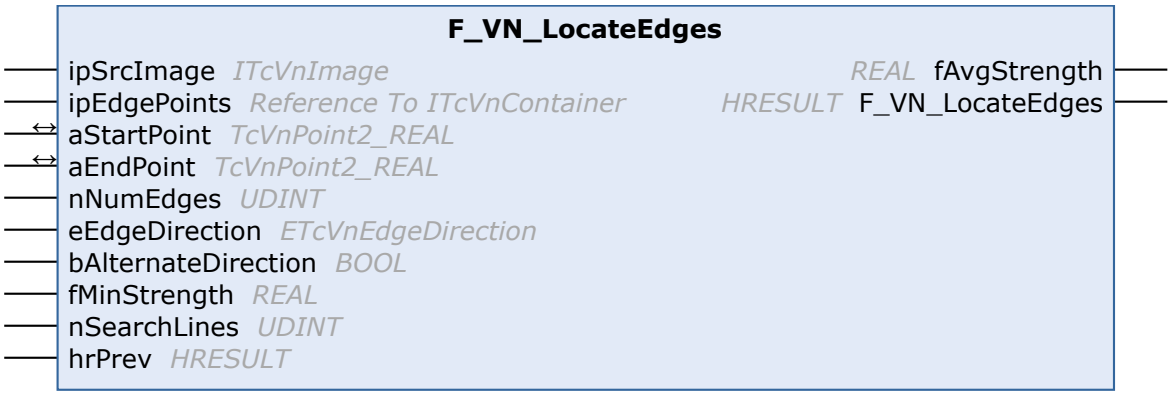
Required License

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 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 [▶ 411]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
nNumEdges	UDINT	The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edges

Return value

HRESULT [▶ 141]

Further information

The function F_VN_LocateEdges localizes several edges in a defined search window.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Localized points (Return value)

The container `ipEdgePoints` returns the localized points of the linear edges.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

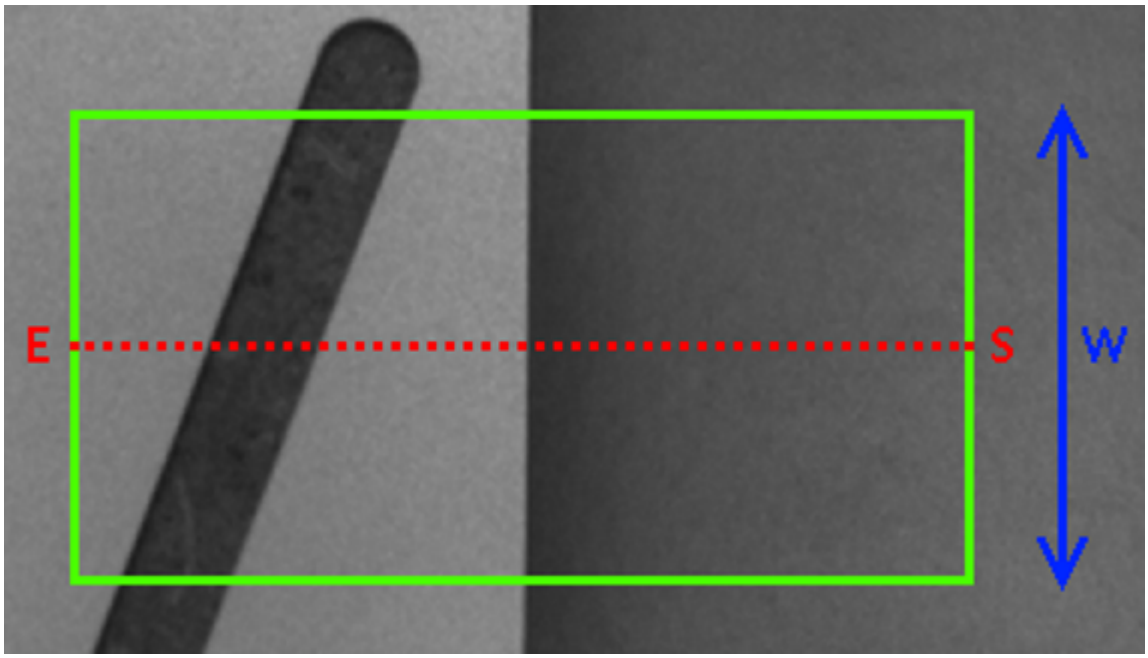


Fig. 17: S: StartPoint, E: EndPoint, W: WindowWidth

Edge strength (Return value)

The return value `fAvgStrength` displays the actual contrast of the localized edges (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_LocateEdgeExp contains additional parameters.

Application

The localization of a linear edge looks like this, for example:

```
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
);
```

The localized edge points can be visualized with the function F_VN_DrawContours [▶ 1009].

```
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

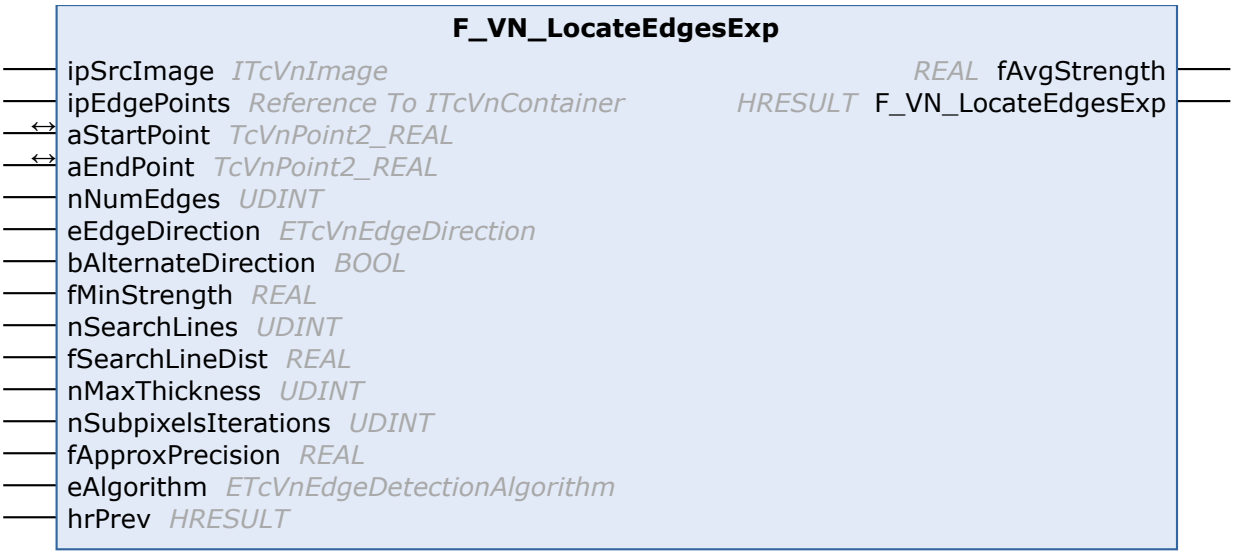
Required License

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 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 [▶ 411]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
nNumEdges	UDINT	The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edges

Return value

HRESULT [▶ 141]

Further information

The function F_VN_LocateEdgesExp is the expert version of F_VN_LocateEdges [▶ 1501]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Localized points (Return value)

The container `ipEdgePoints` returns the localized points of the linear edges.

The search window

First, a search window has to be defined to determine where and in which direction to search for an edge. This is done with the parameters aStartPoint, aEndPoint and nSearchLines. The search window results from the search line from aStartPoint to aEndPoint and parallel to it (nSearchLines - 1) / 2 search lines in each direction, each with 1 pixel distance. Thus nSearchLines defines the number of search lines and must always have an odd value.

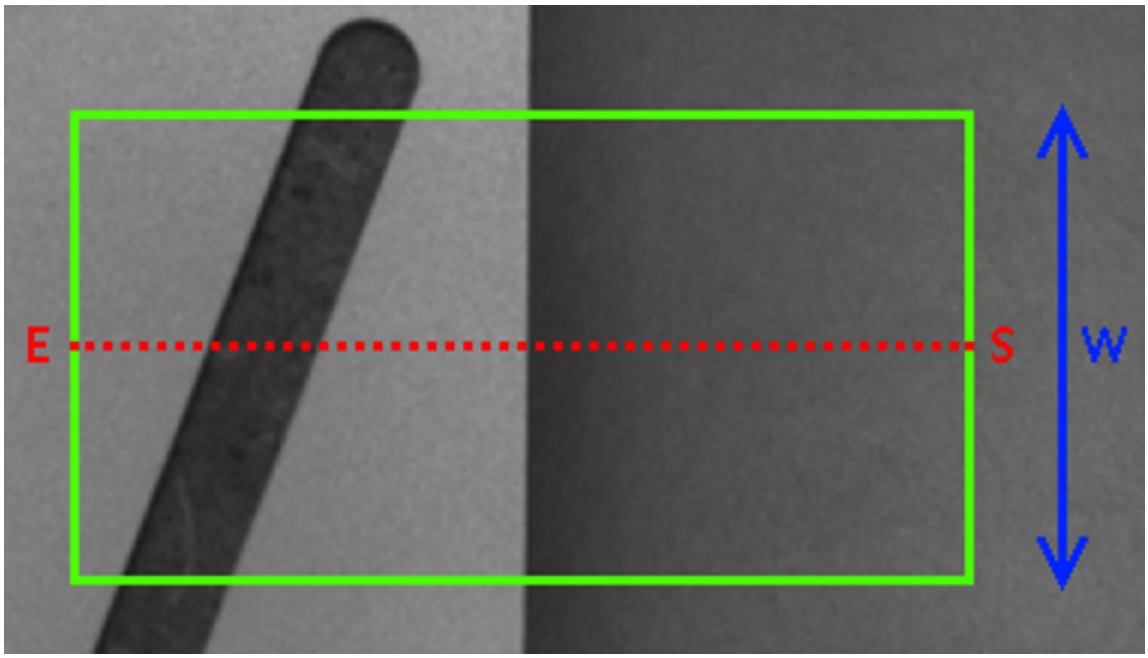


Fig. 18: S: StartPoint, E: EndPoint, W: WindowWidth

Edge strength (Return value)

The return value `fAvgStrength` displays the actual contrast of the localized edges (averaged over all search lines).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The localization of a linear edge with expert parameters looks like this, for example:

```
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
);
```

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

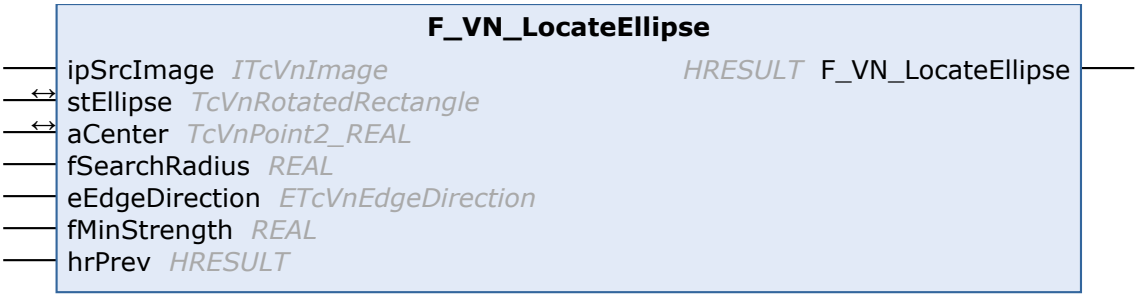
Required License

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 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 [▶ 411]	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 [▶ 199]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 160]	The expected ellipse center

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocateEllipse` localizes and measures an ellipse. It is based on the edge localization [▶ 1481] and uses normal parameters. It returns the localized ellipse as a structure of the type TcVnRotatedRectangle [▶ 245] (center point, size and angle).

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Ellipse (Return value)

The parameter stEllipse returns the localized ellipse as structure of the type TcVnRotatedRectangle [▶ 245] (center point, size and angle).

Search circle

The circular search area is defined via the expected center point `aCenter` and a search radius `fSearchRadius`. The search radius should be set somewhat larger than the expected ellipse radius in order to securely localize the ellipse.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_LocateEllipse(Exp) [▶ 1510] contains additional parameters.

Application

The localization and measurement of an ellipse with an estimated center point of `[360, 240]` and a radius of `100 px` looks like this, for example: for example:

```
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
);
```

The properties of the localized ellipse can then be retrieved via the variables `stEllipse.aCenter`, `stEllipse.stSize` and `stEllipse.fAngle`. In addition, the ellipse can be visualized with the function F_VN_DrawEllipse [▶ 1012]:

```
hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 5, hr);
```

Samples

- Locate Ellipse [▶ 2822]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

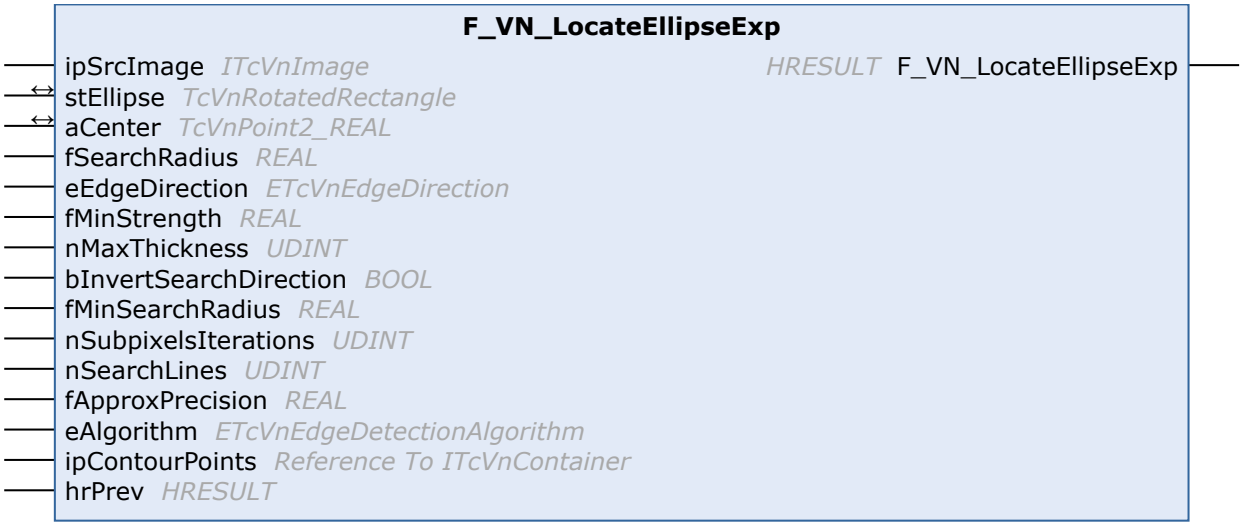
Required License

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 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 [▶ 411]	Source image (1 channel)
fSearchRadius	REAL	Radius around aCenter to search for edges
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
ipContourPoints	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 245]	Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 160]	The expected ellipse center

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_LocateEllipseExp` is the Expert version of F_VN_LocateEllipse [▶ 1507].

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Ellipse (Return value)

The parameter stEllipse returns the localized ellipse as structure of the type TcVnRotatedRectangle [▶ 245] (center point, size and angle).

Search circle

The circular search area is defined via the expected center point `aCenter` and a search radius `fSearchRadius`. The search radius should be set somewhat larger than the expected ellipse radius in order to securely localize the ellipse.

Localized points (Expert, return value)

The container `ipContourPoints` returns the localized points.

The parameter can be set to 0 if the return of the edge points is not required.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The localization and measurement of an ellipse with Expert parameters looks like this, for example:

```
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
);
```

The localized points of the ellipse can be visualized with the function F_VN_DrawContours [▶ 1009]:

```
hr := F_VN_DrawContours(ipContourPoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Samples

- Locate Ellipse [▶ 2822]

Related functions

- LocateCircularArc [▶ 1489] for localizing proportional circular arcs
- LocateEdge [▶ 1495] for localizing linear edges
- LocateEdges [▶ 1501] for localizing several linear edges
- LocateEllipse [▶ 1507] for localizing complete ellipses

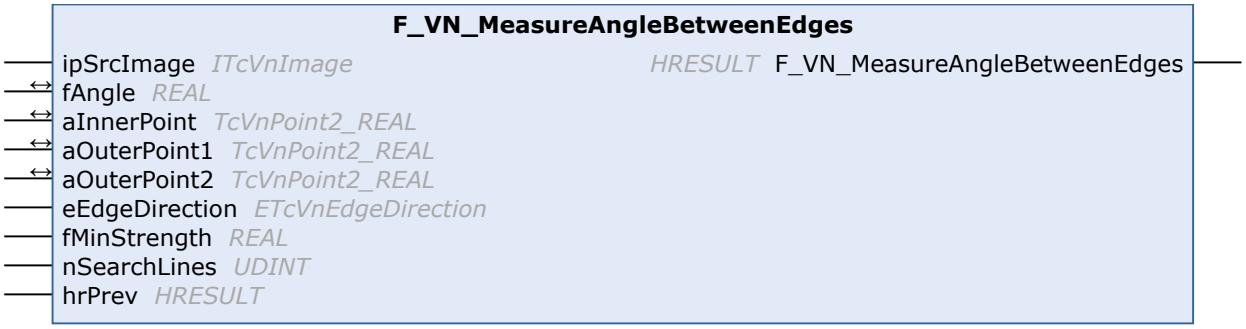
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	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 [▶ 160]	Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 160]	Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 160]	Position where the search process for edge 2 ends

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureAngleBetweenEdges` localizes two linear edges and measures the angle between them.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Angle (Return value)

The parameter `fAngle` returns the angle between the localized edges.

Depending on the parameter `bAngleInDegrees`, the angle `fAngle` is specified in radians (`FALSE`) or in degrees (`TRUE`).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_MeasureAngleBetweenEdgesExp [▶ 1515] contains additional parameters.

Application

The measurement of an angle between two linear edges looks like this, for example:

```
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
);
```

Samples

- Measure Angle Between Edges [▶ 2827]

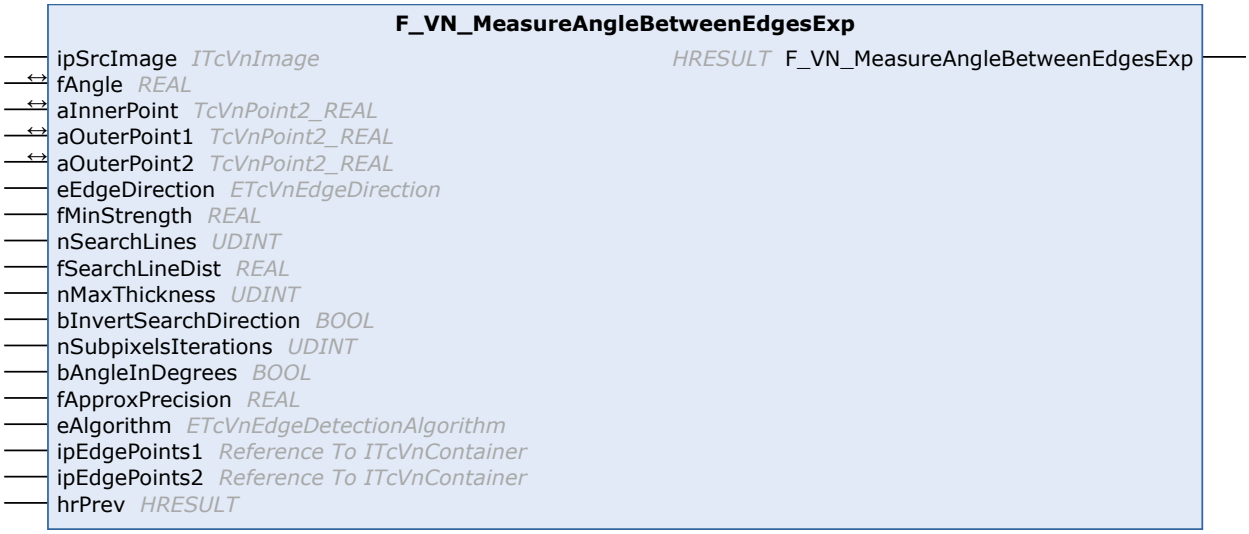
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points between aInnerPoint and aOuterPoint1 (ContainerType_Vector_TcVnPoint2_REAL)
ipEdgePoints2	Reference To ITcVnContainer [▶ 370]	Returns the detected edge points between aInnerPoint and aOuterPoint2 (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 160]	Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 160]	Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 160]	Position where the search process for edge 2 ends

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureAngleBetweenEdgesExp` is the Expert version of F_VN_MeasureAngleBetweenEdges [▶ 1513].

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Angle (Return value)

The parameter `fAngle` returns the angle between the localized edges.

Depending on the parameter `bAngleInDegrees`, the angle `fAngle` is specified in radians (`FALSE`) or in degrees (`TRUE`).

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Localized points (Expert, return value)

The containers `ipEdgePoints1` and `ipEdgePoints2` return the exact points that were localized on the basis of each search line. If not needed, the value `0` can be transferred instead of an interface pointer

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The measurement of an angle with Expert parameters looks like this, for example:

```
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
);
```

Samples

- Measure Angle Between Edges [▶ 2827]

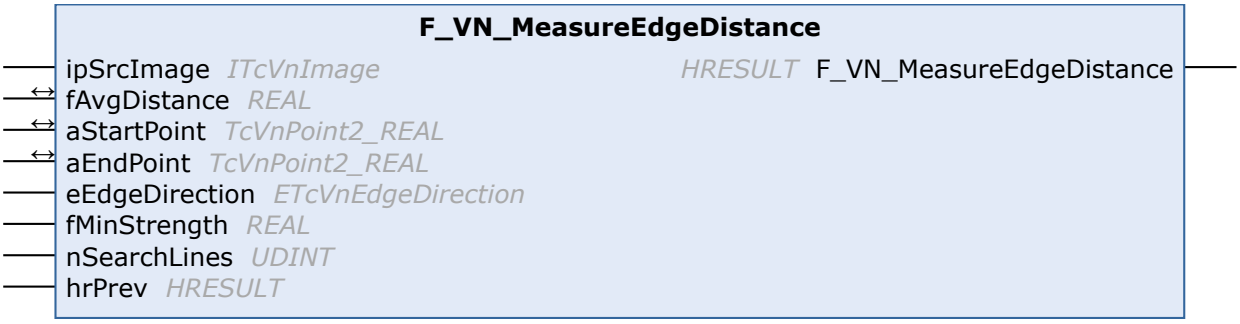
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	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 [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureEdgeDistance` localizes two linear edges and measures the average distance between them.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter `fAvgDistance` returns the minimum distance between the two localized edges.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_MeasureEdgeDistanceExp [▶ 1520] contains additional parameters.

Application

The measurement of an edge distance looks like this, for example:

```
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage              :=   ipImageIn,
    fAvgDistance            :=   fDistance
    aStartPoint             :=   aStartPoint,
    aEndPoint               :=   aEndPoint,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    hrPrev                  :=   hr
);
```

Related functions

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

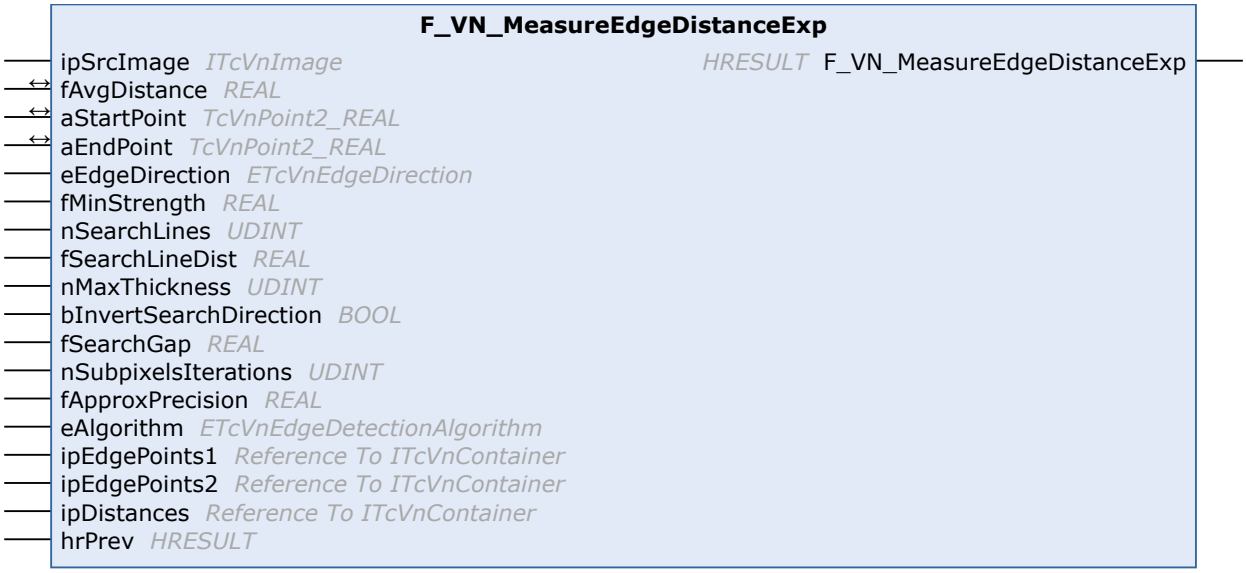
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 370]	Returns the distances between the detected edge points (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureEdgeDistanceExp` is the Expert version of F_VN_MeasureEdgeDistance [▶ 1518]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter `fAvgDistance` returns the minimum distance between the two localized edges.

Localized edge points (Expert, return value)

The containers `ipEdgePoints1` and `ipEdgePoints2` return the exact points that were localized on the basis of each search line. If not needed, the value `0` can be transferred instead of an interface pointer.

Measured distances (Expert, return value)

The container `ipDistances` returns all measured distances between opposite points. If not needed, the value `0` can be transferred instead of an interface pointer.

Search gap (Expert)

Area in the center of the defined search line in which the search is omitted. This can reduce the execution time.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The measurement of an edge distance with Expert parameters looks like this, for example:

```
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
);
```

The localized edge points can be visualized with the function F_VN_DrawContours [▶ 1009]:

```
hr := F_VN_DrawContours(ipEdgePoints1, -1, ipImageRes, aColorGreen, 5, hr);
hr := F_VN_DrawContours(ipEdgePoints2, -1, ipImageRes, aColorGreen, 5, hr);
```

Related functions

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

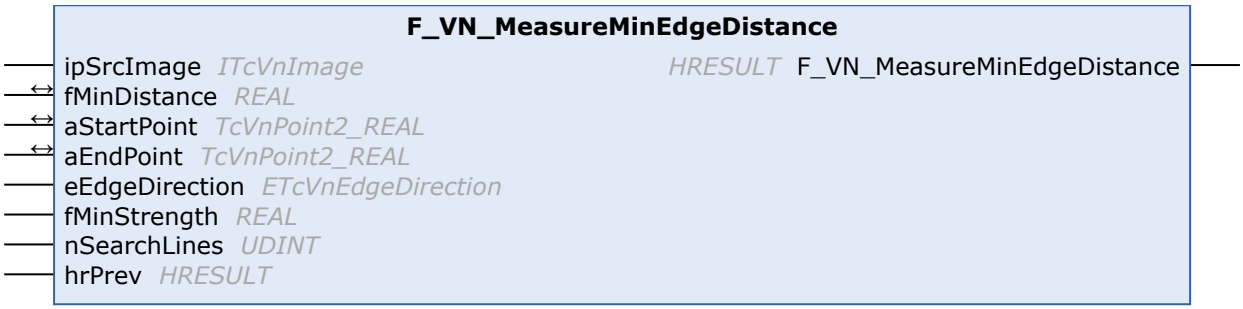
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 141]	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 [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureMinEdgeDistance` localizes two linear edges and measures the minimum distance between them. It returns the minimum distance as `REAL`.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter `fMinDistance` returns the minimum distance between the two localized edges.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Expert parameters

The expert version F_VN_MeasureMinEdgeDistanceExp [▶ 1525] contains additional parameters.

Application

The measurement of an edge distance looks like this, for example:

```
hr := F_VN_MeasureMinEdgeDistance(
    ipSrcImage              :=   ipImageIn,
    fMinDistance            :=   fDistance
    aStartPoint             :=   aStartPoint,
    aEndPoint               :=   aEndPoint,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    hrPrev                  :=   hr
);
```

Related functions

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

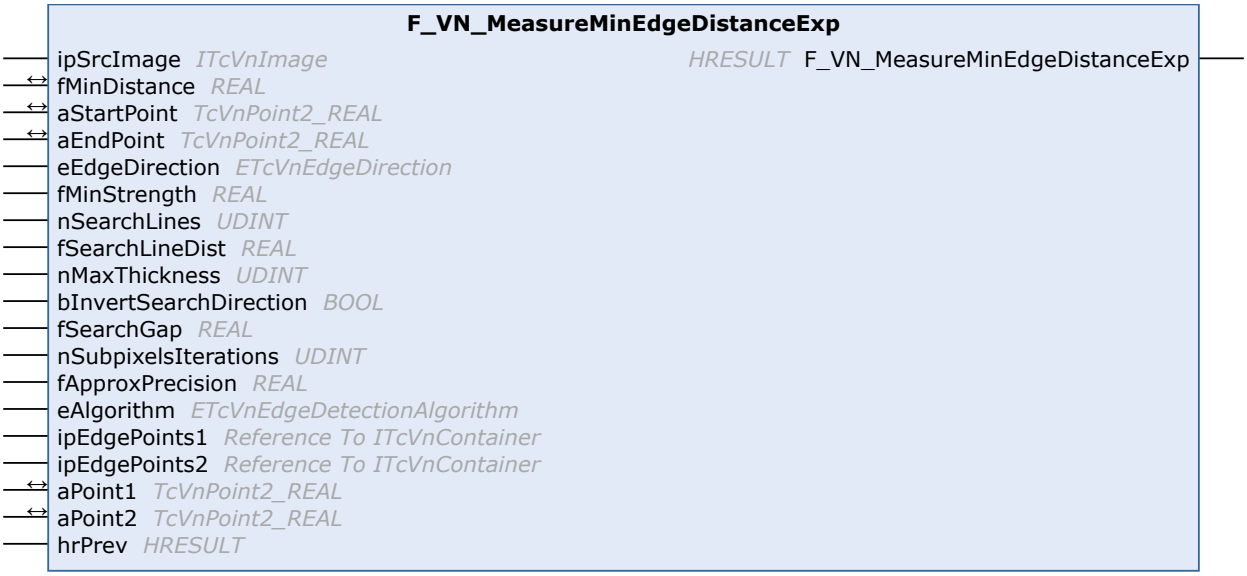
Required License

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 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 [▶ 411]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 199]	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 [▶ 198]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 141]	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 [▶ 160]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 160]	Position where the search process ends
aPoint1	TcVnPoint2_REAL [▶ 160]	Returns the point on the edge near aStartPoint, for which the minimum distance is achieved
aPoint2	TcVnPoint2_REAL [▶ 160]	Returns the point on the edge near aEndPoint, for which the minimum distance is achieved

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_MeasureMinEdgeDistanceExp` is the Expert version of F_VN_MeasureMinEdgeDistance [▶ 1523]. It contains additional parameters.

Parameter

Input image

The input image `ipSrcImage` must be a 1-channel grayscale image.

Minimum distance (Return value)

The parameter `fMinDistance` returns the minimum distance between the two localized edges.

Localized edge points (Return value)

The containers `ipEdgePoints1` and `ipEdgePoints2` return the exact points that were localized on the basis of each search line. If not needed, the value 0 can be transferred instead of an interface pointer.

In addition, the points `aPoint1` and `aPoint2` return the points that have the minimum distance between the two edges.

Search gap (Expert)

Area in the center of the defined search line in which the search is omitted. This can reduce the execution time.

Parameters for edge localization

The remaining parameters are explained in detail in the chapter Edge localization [▶ 1481].

Application

The measurement of the minimum edge distance with Expert parameters looks like this, for example:

```
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
);
```

The localized edge points can be visualized with the functions F_VN_DrawContours [▶ 1009] and F_VN_DrawPoint [▶ 1031]:

```
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);
```

Related functions

- ClosestPointsBF [▶ 1487] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 1513] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 1518] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 1523] for measuring the minimum distance between two edges

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21 Miscellaneous

This group contains further functions that can be used across groups.

Functions

Geometrical auxiliary functions

- F_VN_ClipLineToBoundary [▶ 1529]
- F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531]
- F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1532]
- F_VN_LineIntersectionPoint [▶ 1541]
- F_VN_LineIntersectionPointAndAngle [▶ 1542]
- F_VN_RotatedRectangleCorners [▶ 1544]
- F_VN_RotatedRectangleIntersection [▶ 1545]

Timestamp

- F_VN_GetTimestamp [▶ 1537]
- F_VN_UpdateTimestamp [▶ 1547]

Auxiliary functions for matrices

- F_VN_InitMatrixStruct [▶ 1539]
- F_VN_InvertMatrix3x3 [▶ 1540]
- F_VN_MultiplyMatrices [▶ 1543]

Converting interfaces

- F_VN_ConvertITcUnknownToITcVnBitmapExport [▶ 1533]
- F_VN_ConvertITcUnknownToITcVnContainer [▶ 1534]
- F_VN_ConvertITcUnknownToITcVnImage [▶ 1535]
- F_VN_ConvertITcUnknownToITcVnMlModel [▶ 1536]

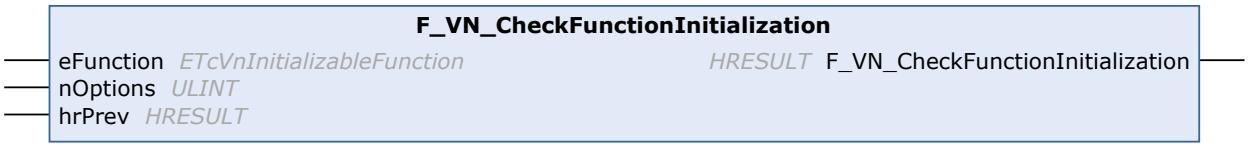
Initialization function

- F_VN_CheckFunctionInitialization [▶ 1529]
- F_VN_DeinitializeFunction [▶ 1536]

Miscellaneous

- F_VN_HuMomentInvariants [▶ 1538]
- F_VN_SetRngSeed [▶ 1546]

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 [▶ 205]	Initializable function
nOptions	ULINT	Initialization options for the function
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

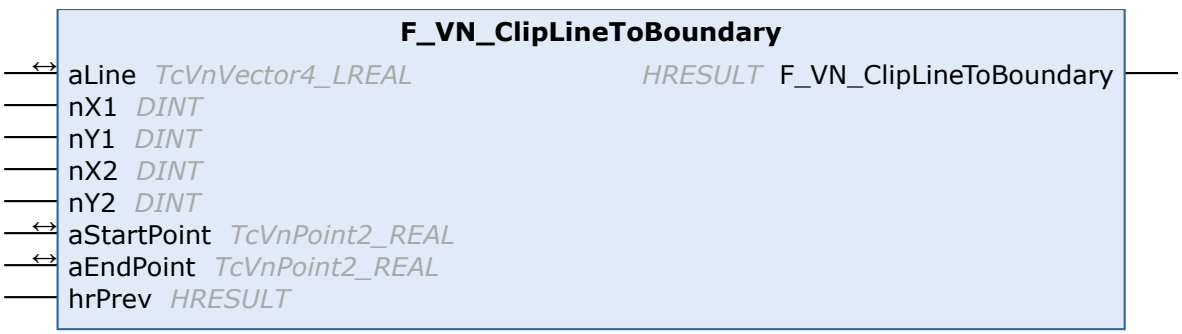
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 160]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 160]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Further information

This function adjusts a line to a specified area.

Different functions exist for various definition forms in this area:

Definition of the rectangular area	Function to be used
Specified by individual X/Y coordinates	F_VN_ClipLineToBoundary [▶ 1529]
By the size of an image	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531]
By a rectangular structure	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1532]

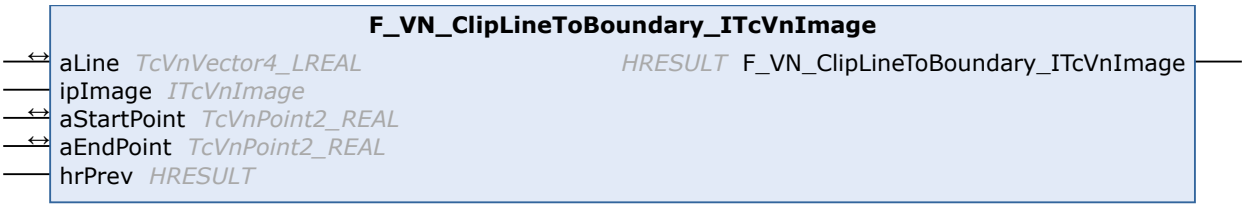
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]	Image, from which to derive the boundary
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 160]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 160]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Further information

This function adjusts a line to a specified area.

Different functions exist for various definition forms in this area:

Definition of the rectangular area	Function to be used
Specified by individual X/Y coordinates	F_VN_ClipLineToBoundary [▶ 1529]
By the size of an image	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531]
By a rectangular structure	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1532]

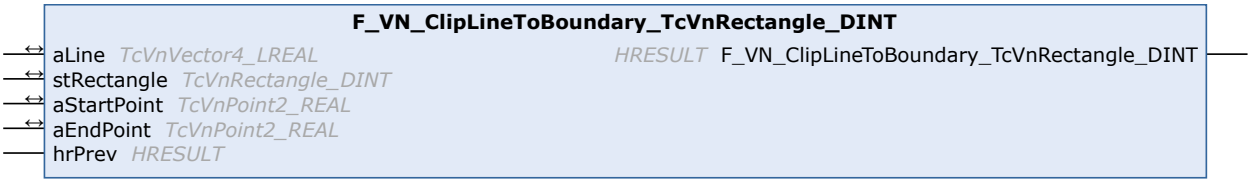
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 162]	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 [▶ 244]	Rectangular boundary
aStartPoint	TcVnPoint2_REAL [▶ 160]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 160]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Further information

This function adjusts a line to a specified area.

Different functions exist for various definition forms in this area:

Definition of the rectangular area	Function to be used
Specified by individual X/Y coordinates	F_VN_ClipLineToBoundary [▶ 1529]
By the size of an image	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1531]
By a rectangular structure	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1532]

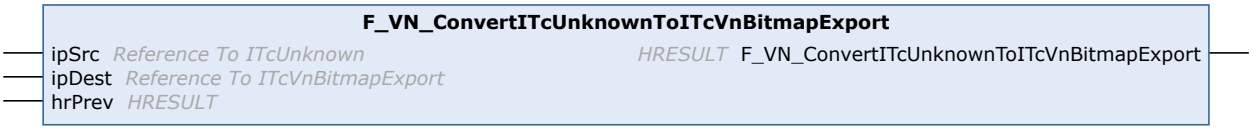
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 428]	Source pointer
ipDest	Reference To ITcVnBitmapExport [▶ 394]	Destination pointer
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

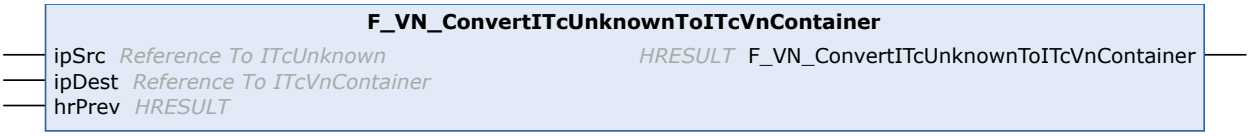
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 428]	Source pointer
ipDest	Reference To ITcVnContainer [▶ 370]	Destination pointer
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

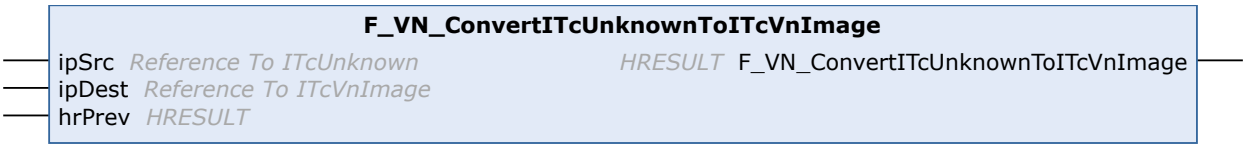
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 428]	Source pointer
ipDest	Reference To ITcVnImage [▶ 411]	Destination pointer
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

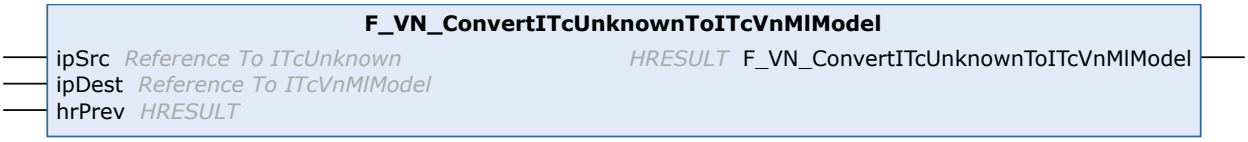
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 428]	Source pointer
ipDest	Reference To ITcVnMlModel [▶ 420]	Destination pointer
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 205]	Initializable function
nOptions	ULINT	Initialization options for the function
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

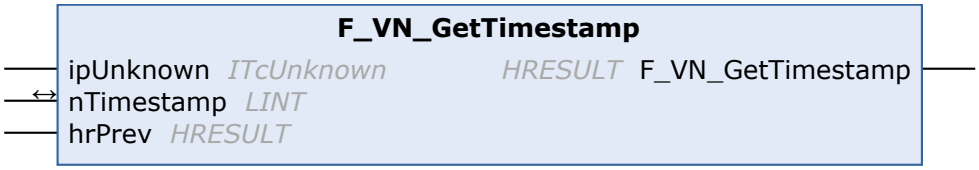
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 428]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

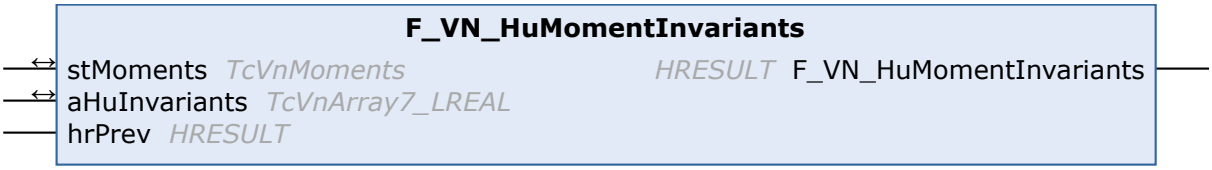
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 232]	Moments from which to compute the Hu invariants
aHuInvariants	TcVnArray7_LREAL [▶ 162]	Returns the Hu moment invariants

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 199]	Type of the matrix elements
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stDestMatrix	TcVnMatrix [▶ 231]	Returns completed matrix struct

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_InitMatrixStruct` creates a matrix structure of the type TcVnMatrix [▶ 231] from any memory area.

Parameter

Data buffer

The pointer `pSrcBuffer` must point to the memory area containing the data for the matrix. This could be an array, for example.

Matrix description

The matrix must be described by the number of rows `nRows` and columns `nCols` as well as by the data type of the element `eElementType`.

Result matrix

The matrix structure `stDestMatrix` of the type TcVnMatrix [▶ 231] is returned as the result.

Application

The creation of a `3×3` matrix looks like this, for example:

```
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);
```

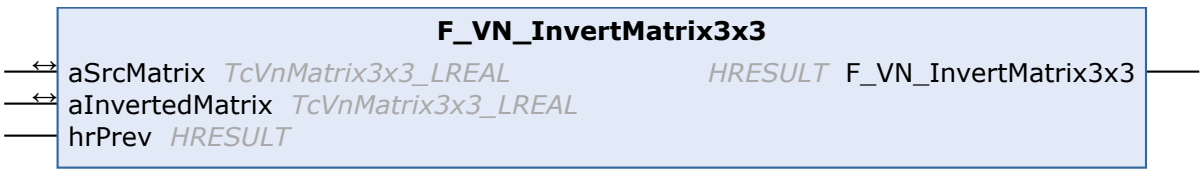
Required License

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Source matrix
aInvertedMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns inverted matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine1	TcVnVector4_LREAL [▶ 162]	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 [▶ 162]	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 [▶ 160]	Returns the intersection point of aLine1 and aLine2.

Return value

HRESULT [▶ 141]

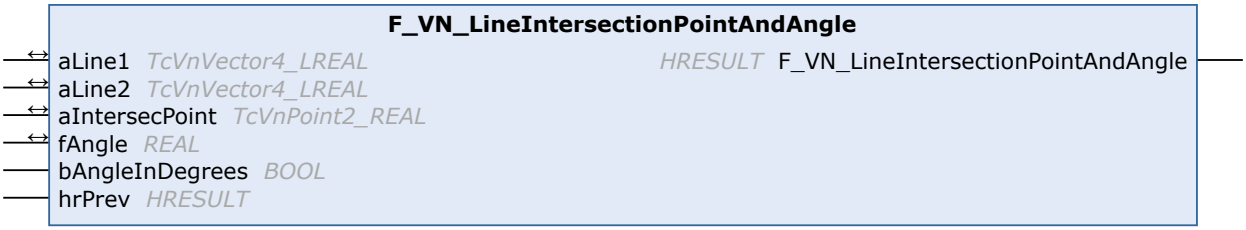
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine1	TcVnVector4_LREAL [▶ 162]	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 [▶ 162]	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 [▶ 160]	Returns the intersection point of aLine1 and aLine2.
fAngle	REAL	Returns the intersection angle of aLine1 and aLine2.

Return value

HRESULT [▶ 141]

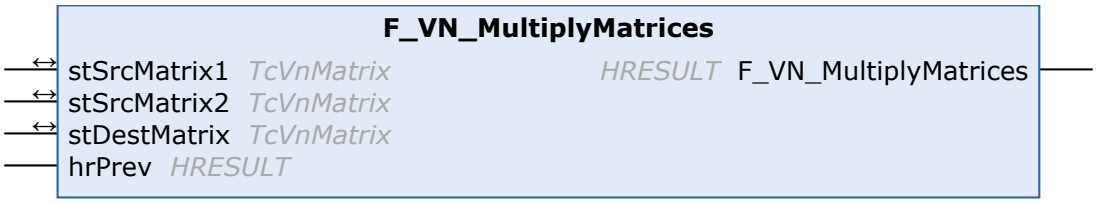
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stSrcMatrix1	TcVnMatrix [▶ 231]	First source matrix (A)
stSrcMatrix2	TcVnMatrix [▶ 231]	Second source matrix (B)
stDestMatrix	TcVnMatrix [▶ 231]	Destination matrix (C) (The destination matrix is filled by this function, but the required memory needs to be provided.)

Return value

HRESULT [▶ 141]

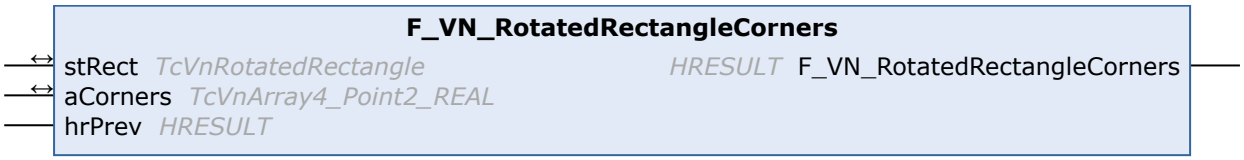
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.17 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRect	TcVnRotatedRectangle [▶ 245]	Rotated rectangle
aCorners	TcVnArray4_Point2_REAL [▶ 162]	Returns the 4 corner points

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.18 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 [▶ 370]	Returns the intersection points (ContainerType_Vector_TcVnPoint2_REAL, non-zero interface pointers are reused)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRect1	TcVnRotatedRectangle [▶ 245]	Rotated rectangle 1
stRect2	TcVnRotatedRectangle [▶ 245]	Rotated rectangle 2
eIntersection	ETcVnRectangleIntersection [▶ 214]	Returns the intersection type

Return value

HRESULT [▶ 141]

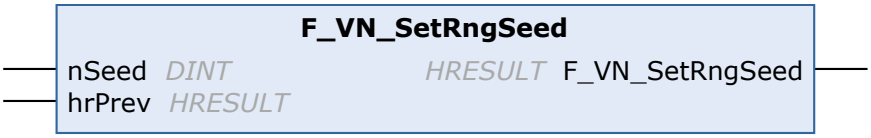
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.19 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

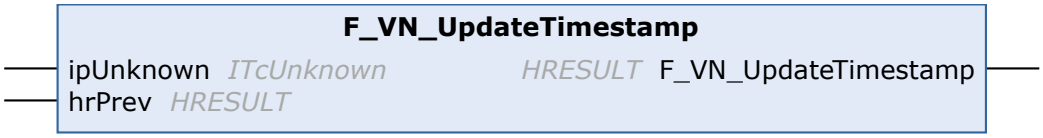
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.20 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 [▶ 428]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22 OCR

This group contains functions for **O**ptical **C**haracter **R**ecognition.

The OCR functions [▶ 1550] identify characters in an image and return the recognized characters as a string. The classification is based on classic Machine Learning models. The models are provided in a trained form so that the functions can be used directly. No additional settings or training are required. The supported characters and fonts depend on the data used in the training. For this reason, there are various models whose range of functions, as well as the general preconditions and requirements, are described below.

For all characters that are recognized in an image, the models assign each character to a known class with the highest match. The result therefore always consists of the known characters of the model used. Rejection or removal of unknown characters is not included.

The functions only support single-line character strings; several lines must be divided into individual ROIs and read with several function calls.

General requirements for recognizing characters:

- Character height min. 20 pixels
- Line width min. 3 pixels
- Points min. 3 x 3 pixels
- Lines min. 3 x 6 pixels
- Character spacing / separation min. 4 pixels
- Characters must not overlap
- Only horizontal alignment / arrangement of characters max. ± 6° deviation
- The line / contour of a character must not be interrupted
- The characters must not be mirrored or reversed

General requirements for the image:

- ROI only with the characters to be recognized and an interference-free zone around the characters
- Within the ROI, the characters must not be enclosed by another contour such as a rectangle
- Good contrast between character and background
- Homogeneous, non-noisy or disturbed, non-transparent background

Requirements for the fonts:

- Only proportional fonts with equal character widths
- Larger gaps are always recognized as a single space
- Only sans serif fonts such as Arial, Tahoma, Courier, Univers, Frutiger, Verdana, OCR-B
- No mixed fonts
- No dot print or italic fonts

Models:

The enum ETcVnOcrModelType [▶ 211] allows access to the following models:

- `TCVN_OMT_NUMBERS`
  - enables the classification of numbers
  - contains the characters of `0-9`
- `TCVN_OMT_NUMBERS_SC`
  - enables the classification of numbers and special characters
  - contains the characters from `0-9` and 6 special characters `. / - : = +`
- `TCVN_OMT_UCLETTERS`
  - enables the classification of capital letters
  - contains the characters of `A-Z`
- `TCVN_OMT_NUMBERS_SC_UCLETTERS`
  - enables the classification of numbers, special characters and capital letters
  - contains the characters from `0-9`, the 6 special characters `. / - : =` + and `A-Z`

When using the combined model (`TCVN_OMT_NUMBERS_SC_UCLETTERS`), confusion can occur due to the great similarity of certain characters. Examples of this are O and 0, S and 5 and B and 8.

If the position of a number or letter is known, the Expert function with the formatting default (`sPattern`) and the combination of the separate models should be used as an alternative. Alternatively, the characters of the result string (`ipCharacters`) can also be analyzed individually, so that you can decide individually whether, for example, a 0 is also accepted instead of O.

Initialize functions

In order to be able to use the OCR function with one or more models, the function must first be initialized with the respective models using the function block FB_VN_InitializeFunction [▶ 1590].

The time required to load the models depends on the performance of the IPC used and the model size and can take several hundred milliseconds or even several seconds. Therefore, cycle time overruns are to be expected with shorter task cycle times.

If the combined model `TCVN_OMT_NUMBERS_SC_UCLETTERS` is to be used, the router memory must be set to at least 512 MB due to the file size.

The exact size depends on the other use of the router memory and any existing fragmentation. If the function block `FB_VN_InitializeFunction` returns a return code `0x80004005`, for example, this indicates insufficient router memory. The router memory should therefore be checked first and increased if necessary. After loading a model, a large part of the memory is released again and is available for the application.

Interpretation of the HRESULT

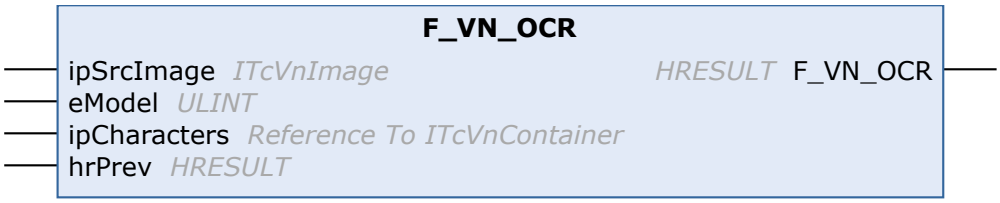
- The standard function returns `S_OK` if characters are recognized on the image. With the Exp function, it depends on whether a `sPattern` has been specified. If no formatting specification has been defined, the return corresponds to the standard function. If a formatting specification has been passed, `S_OK` is only returned if the recognized characters match the pattern specification.
- If the standard function was executed successfully and no characters were found or the characters in the Exp function do not match the pattern specification, `S_FALSE` is returned.
- If a `sPattern` entry is made in the Expert function for which the transferred model is missing, a `HRESULT = ADSERR_DEVICE_NOTINIT` is returned.
- If several models are passed to the standard function, a `HRESULT = ADSERR_DEVICE_INVALIDPARM` is always returned, as the function only supports one model per call. With the Expert function, it depends on whether a `sPattern` has been specified. If `sPattern` is empty, only the `TCVN_OMT_NUMBERS_SC_UCLETTERS` model is used. If this model has not been passed, a `HRESULT = ADSERR_DEVICE_INVALIDPARM` is returned.

Examples for the use of sPattern

The F_VN_OCRExp [▶ 1552] provides additional options with the parameters `sPattern` and `eOcrOptions`. Depending on the combination of characters on the input image, the information from `sPattern` and `eOcrOptions` and the recognized characters, `S_FALSE` is returned if there is a mismatch. The characters of the result string (`ipCharacters`) can then also be analyzed individually, as `S_FALSE` is also returned if the length differs, for example. The samples refer to the use of the `TCVN_OMT_NUMBERS_SC_UCLETTERS` model.

Characters on the ip-SrcImage	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

6.1.4.22.1 F_VN_OCR



Detects and recognizes characters in a binary image (white characters on black background).

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 [▶ 411]	Source image (USINT, 1 channel, binary)
eModel	ULINT	Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	Reference To ITcVnContainer [▶ 370]	Returns the recognized characters (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_OCR` detects and recognizes characters in a binary image and returns the recognized characters in a container.

Parameter

Input image

The input image that is transferred to `ipSrcImage` must be a 1-channel binary image. Only white characters on a black background are permitted.

Model

A model type of ETcVnOcrModelType [▶ 211], which is to be used in a function call to classify the characters, must be transferred to `eModel`. This function only supports the use of one model per function call.

To be able to use a model, it must first be initialized with the function block FB_VN_InitializeFunction [▶ 1590].

Recognized characters

The recognized characters are returned via the reference `ipCharacters`.

Expert parameters

The expert variant F_VN_OCRExp [▶ 1553] contains additional parameters.

Application

For example, a call of the OCR function, with a model for recognizing numbers, with subsequent string export looks like this:

```
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);
```

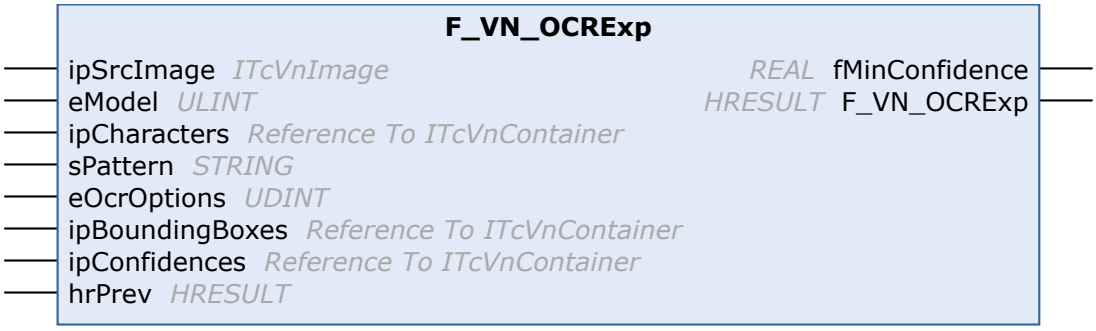
Required License

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.2 F_VN_OCRExp



Detects and recognizes characters in a binary image (white characters on black background). (expert function)

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 [▶ 411]	Source image (USINT, 1 channel, binary)
eModel	ULINT	Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	Returns the corresponding bounding boxes of the recognized characters (ContainerType_Vector_TcVnRectangle_DINT, optional, set to 0 if not required)
ipConfidences	Reference To ITcVnContainer [▶ 370]	Returns the corresponding classification confidences of the recognized characters (ContainerType_Vector_REAL, optional, set to 0 if not required)
hrPrev	HRESULT [▶ 141]	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 [▶ 141]

Further information

The function `F_VN_OCRExp` is the Expert version of F_VN_OCR [▶ 1550]. It contains additional parameters.

Parameter

Input image

The input image that is transferred to `ipSrcImage` must be a 1-channel binary image. Only white characters on a black background are permitted.

Model

The model types of ETcVnOcrModelType [▶ 211] that are to be used to classify the characters in a function call must be transferred to `eModel`. To be able to use several models with one function call, a string pattern must be specified via `sPattern`. The models can be linked with a `OR`.

To be able to use the individual models, they must first be initialized with the function block FB_VN_InitializeFunction [▶ 1590].

Recognized characters

The recognized characters are returned via the reference `ipCharacters`.

String pattern

A string pattern with the expected format of the character string to be recognized can be transferred via `sPattern`.

Supported formatting and its meaning:

- `!` (exclamation mark) to ignore characters (does not apply to spaces)
- `.` (point) for any character (does not apply to spaces)
  The use of the point is only supported with the TCVN_OMT_NUMBERS_SC_UCLETTERS model, otherwise the point has the same effect as an exclamation mark and an S_FALSE is output.
- `_` (underscore) for a space (independent of `eOcrOptions`)
- `d` (number) for an expected number of `0-9`
- `#` (special character) for an expected special character, supported are `. / - : = +`
- `u` (capital letter) for an expected capital letter of `A-Z`

Options

With `eOcrOptions`, options (ETcVnOcrOptions [▶ 211]) can be specified that influence the functionality and result output. If no options are required, a 0 must be passed.

Bounding Boxes

The reference `ipBoundingBoxes` can optionally be used to return the corresponding bounding boxes of the recognized characters. If no bounding boxes are required, the value 0 can be transferred.

Confidences

The reference `ipConfidences` can optionally be used to return the corresponding classification confidences of the recognized characters. If no confidences are required, the value 0 can be transferred.

Minimum confidence

The optional return value `fMinConfidence` returns the minimum confidence value of all characters found.

Application

For example, a call to the OCRExp function, with two models for recognizing numbers and capital letters, with subsequent string export looks like this:

```
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);
```

To evaluate the confidence values or to draw bounding boxes, individual elements of the containers can be accessed via `GetAt`. However, if you want to access several or all elements of a container, it is advisable to program a loop or export the `ipConfidences` container completely as an array.

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.23 Texture Analysis

This group contains functions for analyzing the texture of an image.

Functions

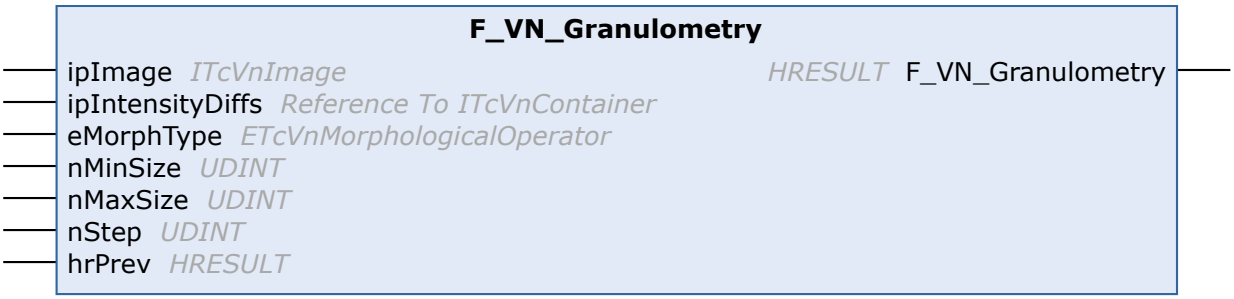
Granulometric properties

- F_VN_Granulometry [▶ 1554]

Haralick features

- F_VN_HaralickFeatures [▶ 1556]

6.1.4.23.1 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 [▶ 411]	Source image (1 channel)
ipIntensityDiffs	Reference To ITcVnContainer [▶ 370]	Returns the intensity differences between successive structuring element sizes (CTcVnContainer_Vector_LREAL), which can be used as features for texture description.
eMorphType	ETcVnMorphologicalOperator [▶ 208]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_Granulometry` calculates the granulometric properties of an image. These values provide an indication of the size distribution of different image areas.

Parameter

Image

The input image `ipImage`.

Features

The granulometric features are returned as a container via the reference `ipIntensityDiffs`.

Operator type

The type of morphological operator is defined via `eMorphType` as `TCVN_MO_OPENING` or as `TCVN_MO_CLOSING`.

Operator size

The size of the morphological operators is determined by the minimum `nMinSize`, the maximum `nMaxSize` and the step size `nStep`. The operators must have an odd size and the step size must be even.

Application

For example, the calculation of granulometric features with closing operators in the sizes three, seven and eleven looks like this:

```
hr := F_VN_Granulometry(
    ipImage             :=  ipImage,
    ipIntensityDiffs    :=  ipIntensityDiffs,
    eMorphType          :=  TCVN_MO_CLOSING,
    nMinSize            :=  3,
    nMaxSize            :=  11,
    nStep               :=  4,
    hrPrev              :=  hr);
```

Related functions

- F_VN_HaralickFeatures [▶ 1556]

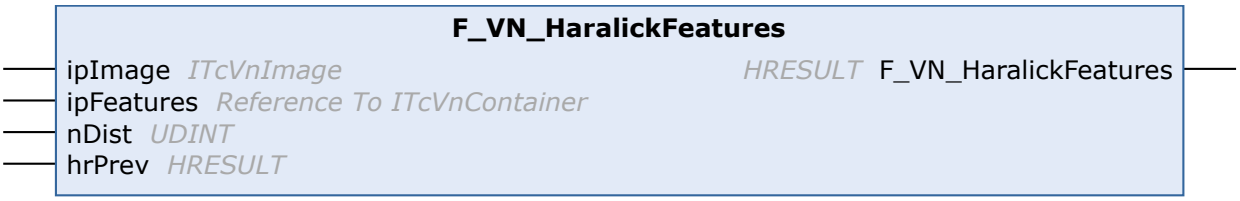
Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.23.2 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 [▶ 411]	Source image (1 channel, USINT)
ipFeatures	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Further information

The function `F_VN_HaralickFeatures` calculates the Haralick properties of an image. These values provide an indication of the structure of the image.

Parameter

Image

The input image `ipImage`.

Features

The Haralick features are returned as a container via the reference `ipFeatures`.

Distance

The distance between neighboring pixels in the Gray-Level Co-occurrence Matrix (GLCM) is specified as `nDist`.

Application

For example, the calculation of Haralick features with a distance of 5 looks like this:

```
hr := F_VN_HaralickFeatures(
    ipImage     :=  ipImage,
    ipFeatures  :=  ipFeatures,
    nDist       :=  5,
    hrPrev      :=  hr);
```

Related functions

- F_VN_Granulometry [▶ 1554]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24 Neural Network

This group contains functions for pre- and post-processing, as well as for the execution of neural network models.

Functions

Pre- & Postprocessing

- F_VN_ConvertDataLayout [▶ 1558]
- F_VN_ReshapeImage [▶ 1566]

Execution of neural networks

- F_VN_ExecuteNeuralNetwork [▶ 1559]
- F_VN_ExecuteNeuralNetwork_SingleLayerOutput [▶ 1561]

- F_VN_ExecuteNeuralNetwork_MultiLayerOutput [▶ 1560]

Postprocessing

- F_VN_Sigmoid(Exp) [▶ 1567]
- F_VN_SoftMax [▶ 1569]
- F_VN_NonMaxSuppression(Exp) [▶ 1562]
- F_VN_Reduce [▶ 1565]
- F_VN_ReduceArg [▶ 1566]

Get the layer names

- F_VN_GetNeuralNetworkLayerNames [▶ 1562]

6.1.4.24.1 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 [▶ 411]	Source image
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

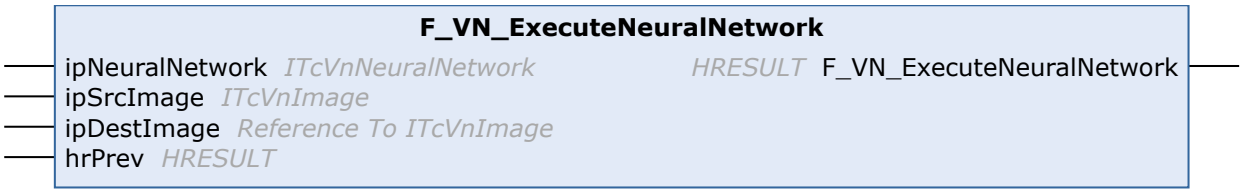
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.2 F_VN_ExecuteNeuralNetwork



Execute a neural network using the provided image as input and provide the output from the last output layer. The function doesn't perform any pre-processing or transformation.

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 [▶ 430]	Neural network model
ipSrcImage	ITcVnImage [▶ 411]	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 [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

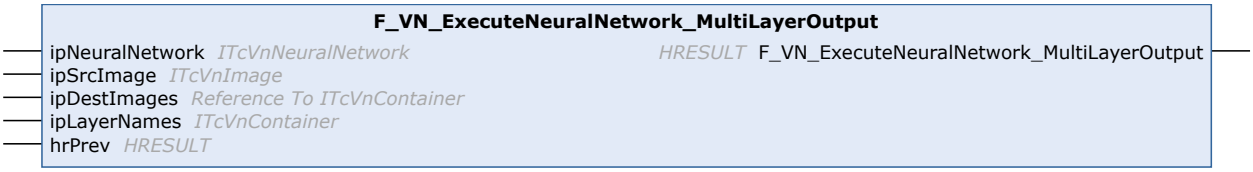
Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.3 F_VN_ExecuteNeuralNetwork_MultiLayerOutput



Execute a neural network using the provided image as input and provide the output from the specified output layers. The function doesn't perform any pre-processing or transformation.

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 [▶ 430]	Neural network model
ipSrcImage	ITcVnImage [▶ 411]	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 [▶ 370]	Container of images containing the output from the specified layers.
ipLayerNames	ITcVnContainer [▶ 370]	Container of layer names whose outputs are required (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

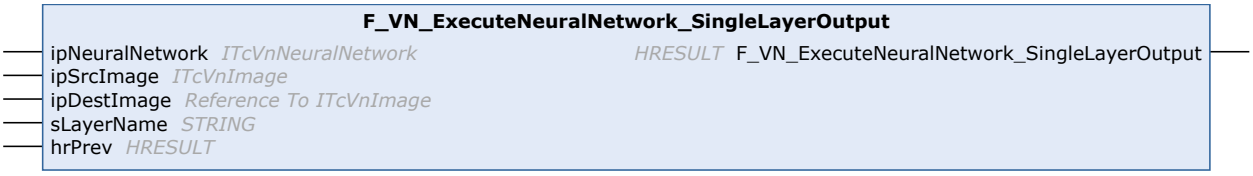
Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.4 F_VN_ExecuteNeuralNetwork_SingleLayerOutput



Execute a neural network using the provided image as input and provide the output from a specified output layer. The function doesn't perform any pre-processing or transformation.

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 [▶ 430]	Neural network model
ipSrcImage	ITcVnImage [▶ 411]	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 [▶ 411]	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 the last output layer is provided.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

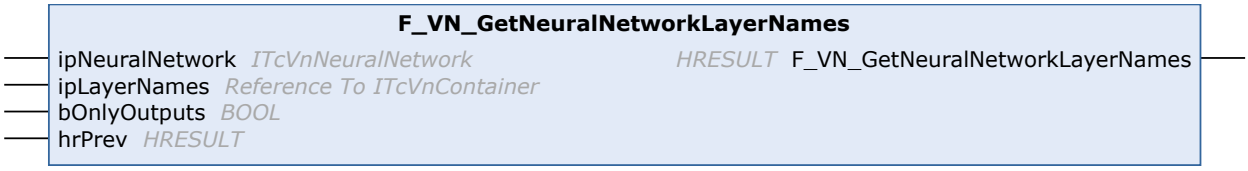
Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.5 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 [▶ 430]	Neural network model
ipLayerNames	Reference To ITcVnContainer [▶ 370]	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	BOOL	If true, only the unconnected output layer names are returned.
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

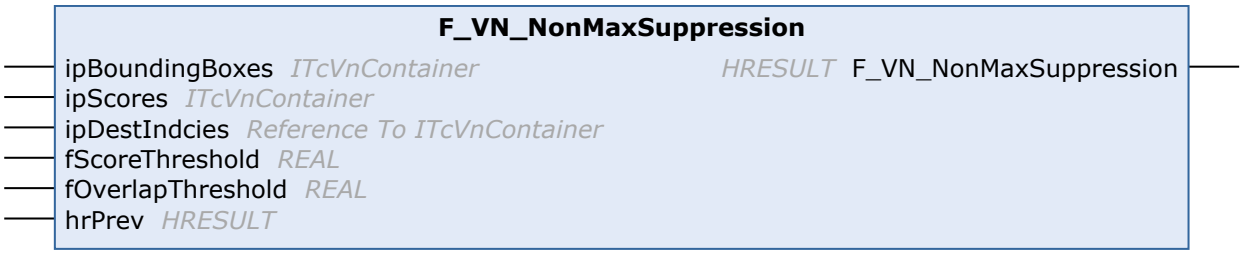
Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.6 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 [▶ 370]	Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer [▶ 370]	Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	Reference To ITcVnContainer [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

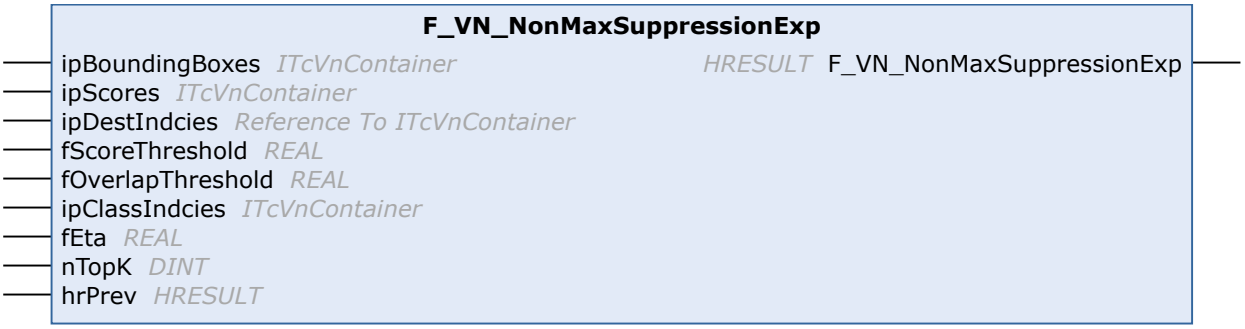
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.7 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 [▶ 370]	Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer [▶ 370]	Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	Reference To ITcVnContainer [▶ 370]	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 [▶ 370]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

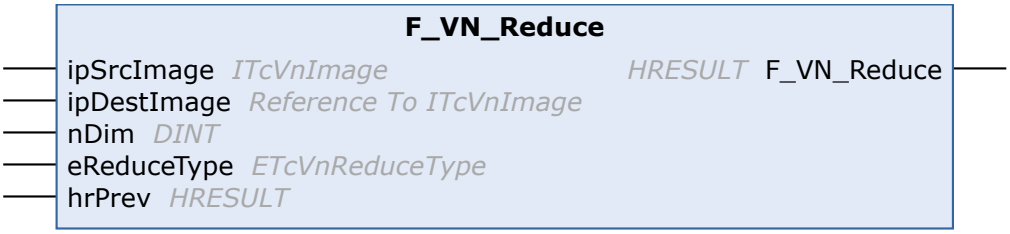
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.8 F_VN_Reduce



Reduce ipSrcImage by applying the specified operation across rows/columns to obtain a single row/column 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 [▶ 411]	Source image (USINT or REAL elements, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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
eReduceType	ETcVnReduceType [▶ 224]	Reduction operation
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.9 F_VN_ReduceArg



Finds indices of max/min elements along provided axis and reduce the ipSrcImage to a single row/column.

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 [▶ 411]	Source image (1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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
eReduceType	DINT	Reduction operation REDUCE_MAX or REDUCE_MIN
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.10 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 [▶ 411]	Source image (1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate single channel destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aNewShape	TcVnArray33_UDINT [▶ 162]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.11 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 [▶ 411]	Source image (REAL)
ipDestImage	Reference To ITcVnImage [▶ 411]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

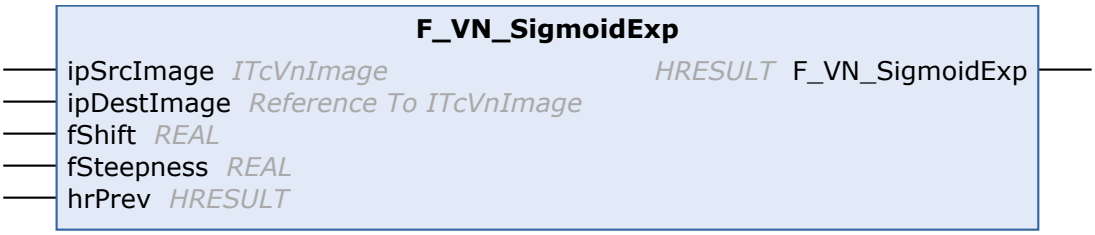
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.12 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 [▶ 411]	Source image (REAL)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.13 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 [▶ 411]	Source image (REAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 411]	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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5 Function Blocks

Communication with GigE Vision cameras or with the file system as well as complex image processing functions is mapped by function blocks. Function blocks are identified by the prefix "FB_VN_".

Grouping of function blocks

All function blocks of the TwinCAT Vision library are sorted thematically into the following groups:

- Camera Register Access [▶ 1571]
- File Access [▶ 1588]
- Image Acquisition [▶ 1611]
- Image Processing [▶ 1652]

6.1.5.1 License Overview

The following list shows the assignment of TwinCAT Vision function blocks to available licenses (see license model [▶ 12]). It shows which license is required for the application of certain function blocks.



The basic license TC3 Vision Base (TF7100) is always required.

TC3 Vision Base

FB_VN_FileSourceControl [▶ 1613]

FB_VN_GevCameraControl [▶ 1621]

FB_VN_InitializeFunction [▶ 1590]

FB_VN_ReadCalibrationPattern [▶ 1592]

FB_VN_ReadCalibrationResult [▶ 1594]

FB_VN_ReadContainer [▶ 1596]

FB_VN_ReadImage [▶ 1598]

FB_VN_ReadMemory [▶ 1572]

FB_VN_ReadMlModel [▶ 1600]

FB_VN_ReadNeuralNetwork [▶ 1609]

FB_VN_ReadRegister_REAL [▶ 1574]

FB_VN_ReadRegister_UDINT [▶ 1576]

FB_VN_ReadRegister_ULINT [▶ 1578]

FB_VN_SimpleCameraControl [▶ 1646]

FB_VN_WriteCalibrationResult [▶ 1601]

FB_VN_WriteContainer [▶ 1603]

FB_VN_WriteImage [▶ 1605]

FB_VN_WriteMemory [▶ 1580]

FB_VN_WriteMlModel [▶ 1607]

FB_VN_WriteRegister_REAL [▶ 1582]

FB_VN_WriteRegister_UDINT [▶ 1584]

FB_VN_WriteRegister_ULINT [▶ 1586]

TC3 Vision Matching

FB_VN_GeneralizedHoughBallard [▶ 1652]

FB_VN_SSIM [▶ 1657]

6.1.5.2 Camera Register Access

This group contains function blocks for reading and writing camera registers or memory areas.

Function blocks

UDINT-Register (4 Byte)

- FB_VN_ReadRegister_UDINT [▶ 1576]
- FB_VN_WriteRegister_UDINT [▶ 1584]

ULINT register (8 bytes)

- FB_VN_ReadRegister_ULINT [▶ 1578]
- FB_VN_WriteRegister_ULINT [▶ 1586]

REAL-Register (4 Byte)

- FB_VN_ReadRegister_REAL [▶ 1574]
- FB_VN_WriteRegister_REAL [▶ 1582]

Larger memory areas

- FB_VN_ReadMemory [▶ 1572]
- FB_VN_WriteMemory [▶ 1580]

Parameter

The following parameters work in a similar way with all function blocks in this group:

Address

The address `nAddress` specifies the register of the camera to be read or saved. The type GVCP_REGISTER_ADDRESS is merely an alias for UDINT; therefore, the address must be 4 bytes (or 32 bits) long. For larger memory areas of the camera (see FB_VN_ReadMemory [▶ 1572]) the address points to the beginning of the area to be read or saved.

Execute the function

The triggers for write and read commands are `bWrite` and `bRead` respectively. They are triggered by a rising edge (change from `FALSE` to `TRUE`). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks

Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout

So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with `nTimeout`. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs `bError` and `nErrorId`.

Working state

The output `bBusy` indicates whether the function block is busy. No new command can be triggered if `bBusy = TRUE`.

By means of a suitable `IF` query it is possible to determine whether the execution of the function block is completed:

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

Error state

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [▶ 2879]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING (16#71E)` if the function block is busy (`bBusy = TRUE`).

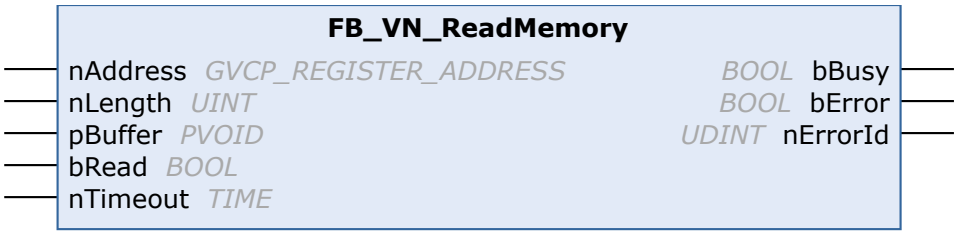
The hexadecimal value of the error code can be extracted as follows:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

Samples

- Reading/writing a register [▶ 2832]

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 [▶ 160]		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.

Further information

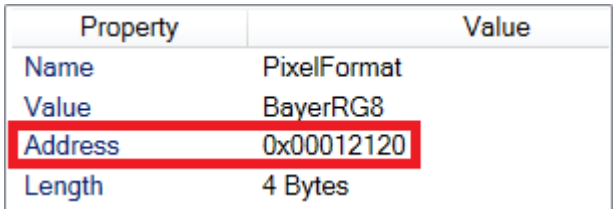
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



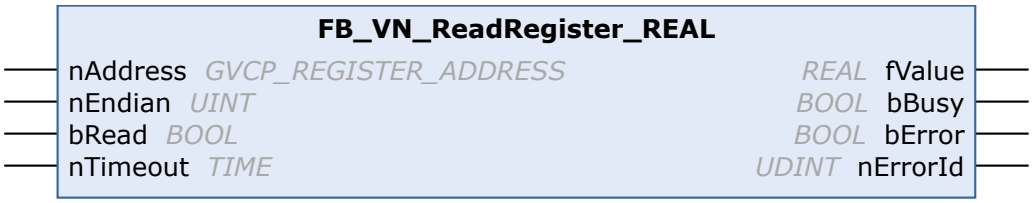
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

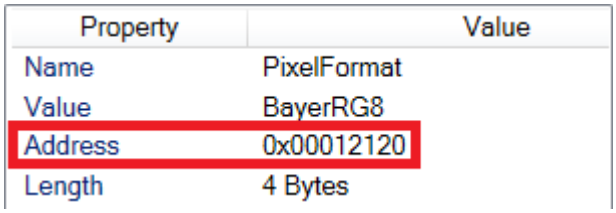
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

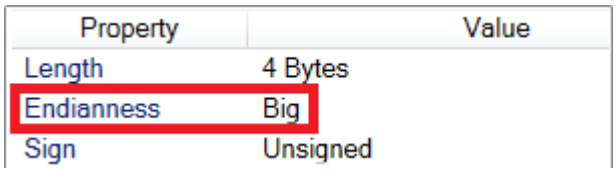
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Samples

- Reading/writing a register [▶ 2832]

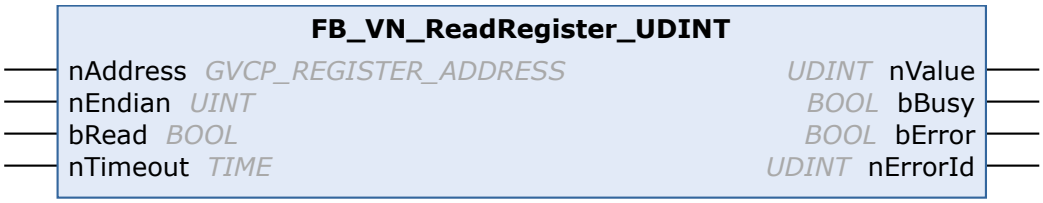
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

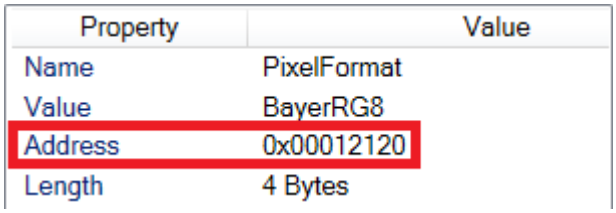
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

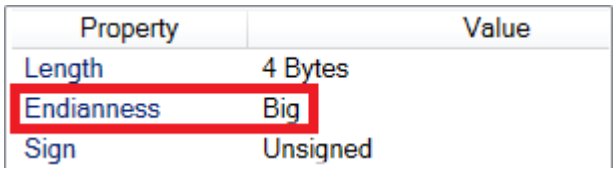
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Samples

- Reading/writing a register [▶ 2832]

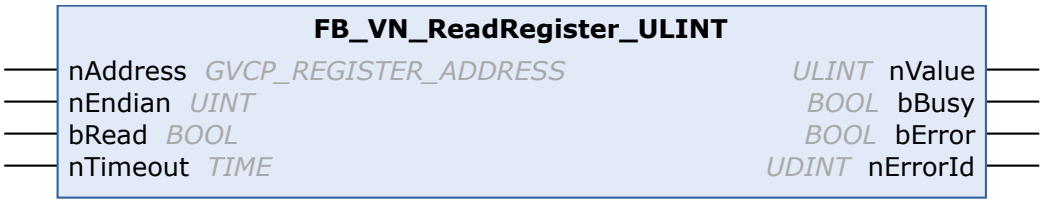
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

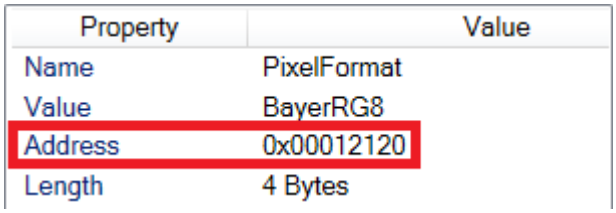
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

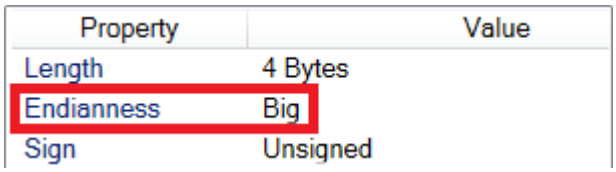
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Examples

Reading/writing a register [▶ 2832]

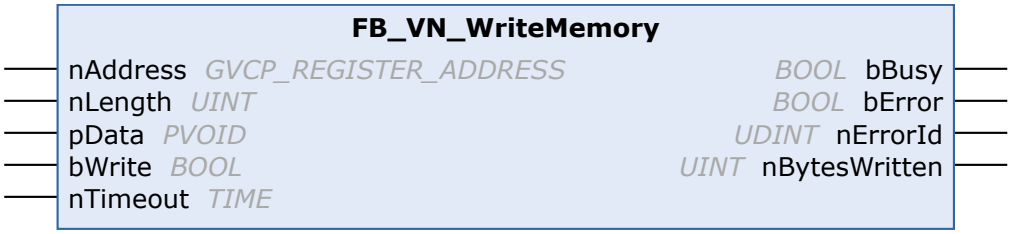
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

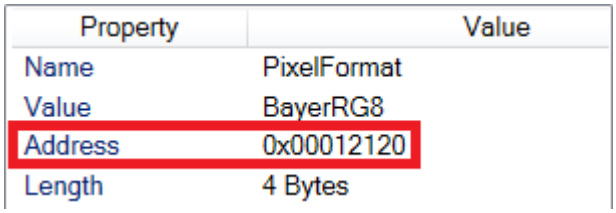
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



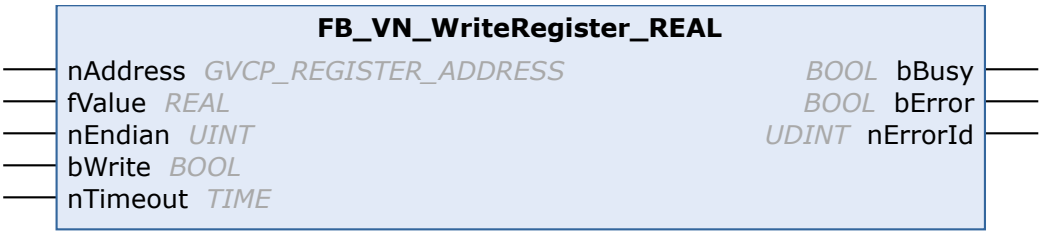
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

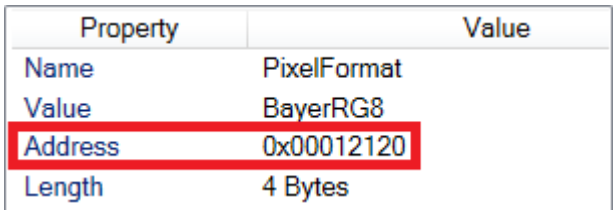
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

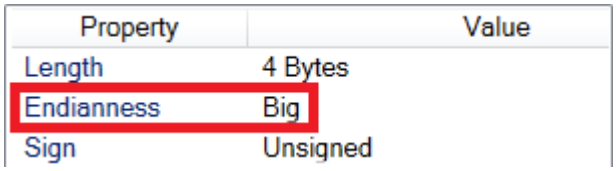
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Samples

- Reading/writing a register [▶ 2832]

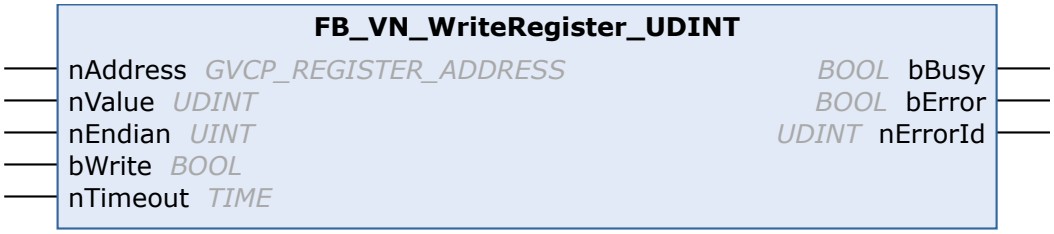
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

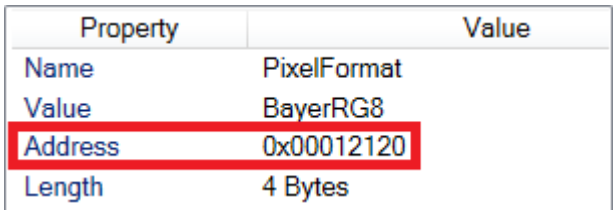
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

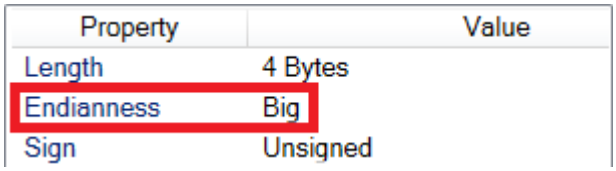
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Samples

- Reading/writing a register [▶ 2832]

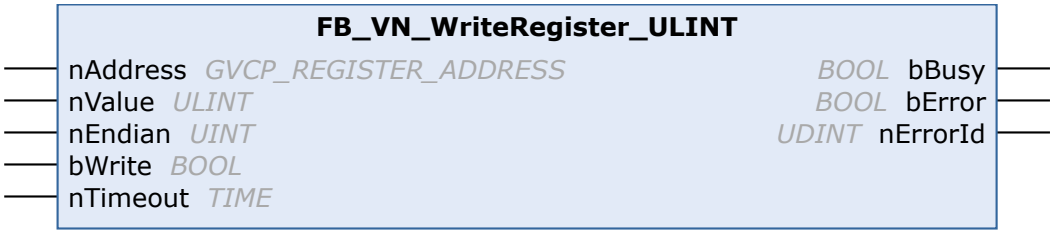
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 160]		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.

Further information

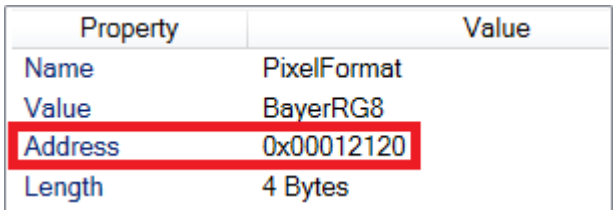
Open camera Control Channel needed!

The Control Channel of the GigE Vision camera must be open in order to be able to use this function block. In addition, some camera parameters can only be changed if the image acquisition is stopped. It is therefore recommended to place the camera in the state TCVN_CS_OPENED first.

Parameter

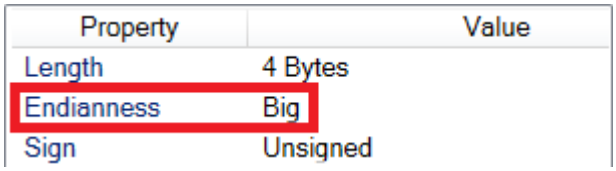
nAddress

The register address identifies the camera register to be addressed. It is of the type `GVCP_REGISTER_ADDRESS`, which is an alias of the type `UDINT`. Thus, all register addresses have a size of 4 bytes. The register address of a camera parameter can be seen at the respective point in the configuration tree [▶ 78] of the GigE Vision camera instance:



nEndian

Endianess describes the order in which the bytes of a camera register are arranged. This does not need to be considered any further in terms of the application and can be adopted as a property of the respective parameter of the configuration tree of the GigE Vision camera instance:



A Big Endian is represented by `nEndian := 0` and a Small Endian by `nEndian := 1`. Big Endian is the default.

Examples

Reading/writing a register [▶ 2832]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3 File Access

This group contains function blocks for reading and writing data on the file system.

Function blocks

Images

- FB_VN_ReadImage [▶ 1598]
- FB_VN_WriteImage [▶ 1605]

Container

- FB_VN_ReadContainer [▶ 1596]
- FB_VN_WriteContainer [▶ 1603]

Machine learning models

- FB_VN_ReadMlModel [▶ 1600]
- FB_VN_WriteMlModel [▶ 1607]

Calibration pattern

- FB_VN_ReadCalibrationPattern [▶ 1592]
- FB_VN_ReadCalibrationResult [▶ 1595]
- FB_VN_WriteCalibrationResult [▶ 1603]

Function initialization

- FB_VN_InitializeFunction [▶ 1590]

Neural Network

- FB_VN_ReadNeuralNetwork [▶ 1609]

General

These function blocks work asynchronously; the time from the first execution to the desired result extends over several cycles. The actual duration of each read/write operation depends on several factors, such as the size of the image and the behavior of the operating system. The following parameters are the same for all function blocks and are therefore described here once centrally.

Parameter

File path

The file path `sFilePath` specifies where to save a file or which file to load. Absolute and relative paths can be used. In the case of relative paths, the default paths set in the Service Configuration [▶ 66] are used as the basis.

Path length

The type of the `sFilePath` is specified as STRING, but paths with a length of up to 255 characters can be passed.

The file path is stored internally on a rising edge of the write command `bWrite` so that changes to the path in later cycles have no effect.

Execute the function

The triggers for write and read commands are `bWrite` and `bRead` respectively. They are triggered by a rising edge (change from `FALSE` to `TRUE`). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks

Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout

So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with `nTimeout`. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs `bError` and `nErrorId`.

Working state

The output `bBusy` indicates whether the function block is busy. No new command can be triggered if `bBusy = TRUE`.

By means of a suitable `IF` query it is possible to determine whether the execution of the function block is completed:

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

Error state

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [▶ 2879]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING (16#71E)` if the function block is busy (`bBusy = TRUE`).

The hexadecimal value of the error code can be extracted as follows:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

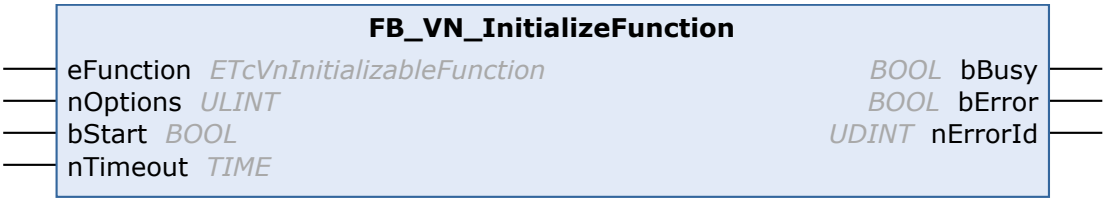
The most frequent error codes are:

Hex	Dec	Name	Meaning
16#006	6	TARGETPORTNOTFOUND	ADS server not reachable. Essentially occurs when the TwinCAT Vision Service has not started. See Vision Service doesn't start [▶ 2864].
16#700	1792	ERROR	General error if saving or loading fails. This usually indicates that the specified file format is not supported or that the file path is unreachable. The error usually occurs in the TwinCAT Vision Service [▶ 66].
16#70B	1803	INVALIDPARM	Invalid parameter. This usually indicates that the interface pointer [▶ 137] is invalid.
16#719	1817	TIMEOUT	Timeout. Loading or saving has exceeded the maximum execution time nTimeout. In general, this occurs mainly with a heavy load on ADS.
16#71E	1822	PENDING	Not really an error, therefore the output bError is not set here. This code signals that the function block is busy. It occurs in parallel to the output bBusy if a new rising edge is registered while the function block is busy.

Examples

- Saving images from the PLC [▶ 2834]

6.1.5.3.1 FB_VN_InitializeFunction



This FB initializes selected functions. Before, it needs to initialize itself, i.e. call the FB until the output bInitialized is true before using it for function initialization.

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 [▶ 205]		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.

Further information

The function block `FB_VN_InitializeFunction` initializes the function specified at `eFunction`. The required data is loaded asynchronously into the PLC in the background.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588]. In contrast to the description, the function to be initialized must be specified for this function block instead of a file path.

Function

The `eFunction` to be initialized.

Options

Selection of the initialization options `nOptions` for the specified function. The possible enum options can be found in the description of the function to be initialized. The options can be linked with a `OR` and thus loaded with a call to the function block.

Application

The initialization of the OCR function with the option to classify capital letters looks like this, for example:

```
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
```

Related functions

- F_VN_CheckFunctionInitialization [▶ 1529] can be used to check whether a function is initialized with a corresponding option.
- F_VN_DeinitializeFunction [▶ 1536] offers the possibility to de-initialize an option of a function and thus release the memory.

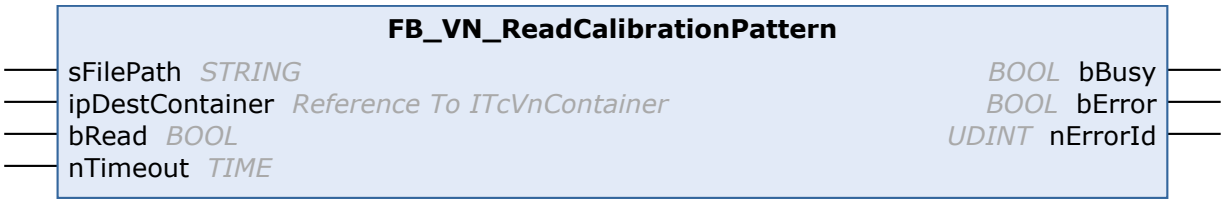
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]		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.

Further information

The function block `FB_VN_ReadCalibrationPattern` loads the description of an individual calibration pattern [▶ 119] asynchronously from a file.

Loading of modified containers

When loading containers, the vision service caches the data internally. Loading changed containers is therefore not possible in the standard system. To do this, adjust the cache settings [▶ 66] or see the sample Serial loading of images to the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Loaded container (return value)

The loaded container with the reference points of the calibration pattern can be reached via the interface pointer `ipPatternReferencePoints` after a successful read operation.

Application

Loading the reference points of a calibration pattern from the path `C:\TcVision\CalibrationPatternReferencePoints.xml` looks like this, for example:

```
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
```

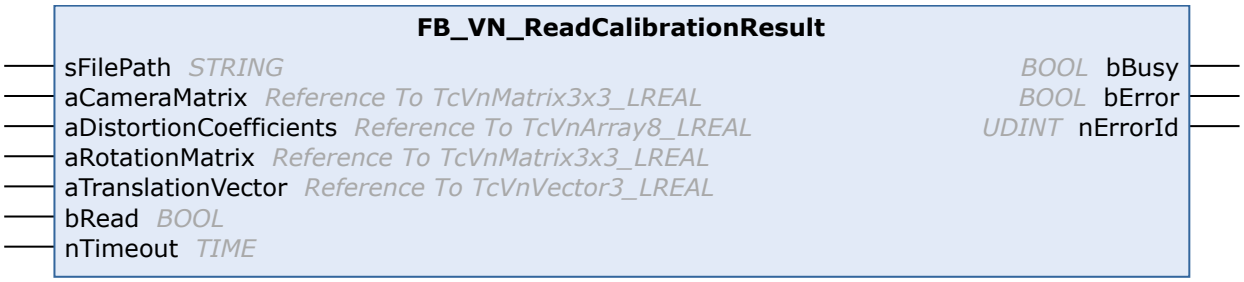
Required License

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 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 [▶ 162]		Returns the camera matrix
aDistortionCoefficients	Reference To TcVnArray8_LREAL [▶ 162]		Returns the distortion coefficients
aRotationMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 162]		Returns the rotation matrix
aTranslationVector	Reference To TcVnVector3_LREAL [▶ 162]		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.

Further information

The function block `FB_VN_ReadCalibrationResult` loads the calibration results from a file asynchronously into the PLC.

Loading of changed calibration results.

When loading the calibration results, the vision service caches the data internally. Loading modified results is therefore not possible in the standard system. To do this, adjust the cache settings [▶ 66] or see the sample Serial loading of images to the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Calibration results

The loaded calibration results can be accessed via the following variables after a successful read operation:

```
// Calibration results
aCameraMatrix            : TcVnMatrix3x3_LREAL;
aDistortionCoefficients  : TcVnArray8_LREAL;
aRotationMatrix          : TcVnMatrix3x3_LREAL;
aTranslationVector       : TcVnVector3_LREAL;
```

Application

Loading the calibration results from the path `C:\TcVision\ CalibrationResult.json` looks like this, for example:

```
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
```

Related function block: FB_VN_WriteCalibrationResult [▶ 1601].

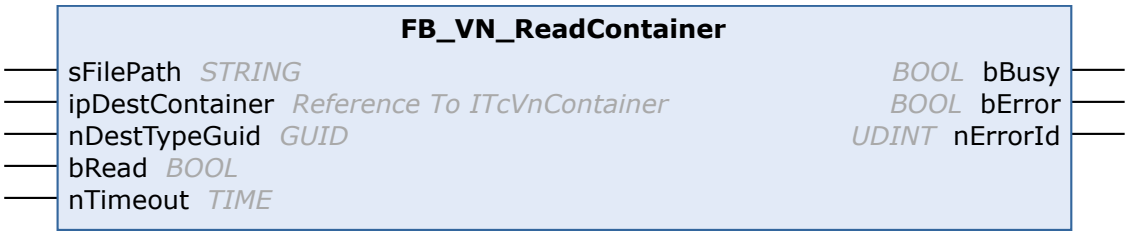
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]		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.

Further information

The function block `FB_VN_ReadContainer` loads a container [▶ 152] from a file asynchronously into the PLC.

Loading of modified containers.

When loading containers, the vision service caches the data internally. Loading changed containers is therefore not possible in the standard system. To do this, adjust the cache settings [▶ 66] or see the example Serial loading of images into the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Container type

The container type [▶ 160] of the container to be loaded can optionally be specified.

Loaded container (return value)

The loaded container is reachable via the interface pointer `ipDestContainer` following successful reading.

Application

Loading a container with explicit type specification `ContainerType_Vector_TcVnPoint2_DINT` from the path `C:\TcVision\Container.xml` looks like this, for example:

```
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
```

Related function block: FB_VN_WriteContainer [▶ 1603].

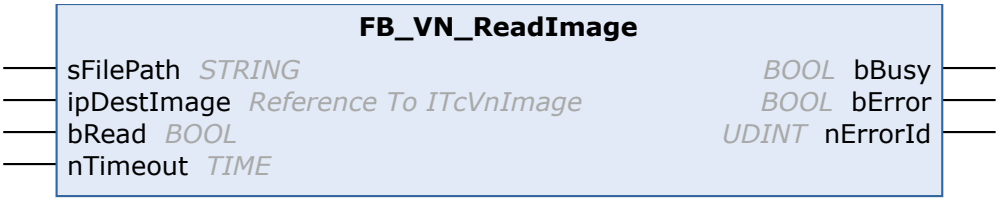
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]		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.

Further information

The function block `FB_VN_ReadImage` loads an image [▶ 149] from a file asynchronously into the PLC.

Loading of modified images

When loading images, the Vision service caches the data internally. Loading changed images is therefore not possible in the standard system. To do this, adjust the cache settings [▶ 66] or see the example Serial loading of images into the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Destination image (return value)

The loaded image is reachable via the interface pointer `ipDestImage` following successful reading.

Application

Loading an image from the path `C:\TcVision\Image.bmp` looks like this, for example:

```
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
```

Related function block: FB_VN_WriteImage [▶ 1605] for saving images.

Samples

- Serial loading of images into the PLC [▶ 2835]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]		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.

Further information

The function block `FB_VN_ReadMlModel` loads an ML model [▶ 420] from a file asynchronously into the PLC.

Loading of modified ML models

When loading the ML model, the vision service caches the data internally. Loading modified ML models is therefore not possible in the standard system. To do this, adjust the cache settings [▶ 66] or see the example Serial loading of images into the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Loaded ML model (return value)

The loaded `ITcVnMlModel` is reachable via the interface pointer `ipDestMlModel` following successful reading.

A special case is the use as `ITcVnColorModel`, because after loading the model a conversion with the following row is still necessary. Alternatively, the `ITcVnMlModel` can also be used directly with the corresponding functions.

```
hr := ipReadMlModel.TcQueryInterface(IID_ITcVnColorModel, ADR(ipColorModel));
```

Application

Loading an ML model from the path `C:\TcVision\MlModel.vmlm` looks like this, for example:

```
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
```

Related function block: FB_VN_WriteMlModel [▶ 1607].

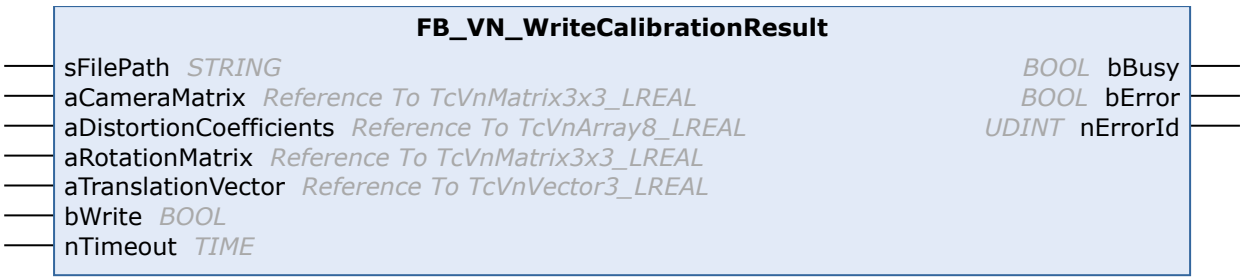
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 162]		Returns the camera matrix
aDistortionCoefficients	Reference To TcVnArray8_LREAL [▶ 162]		Returns the distortion coefficients
aRotationMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 162]		Returns the rotation matrix
aTranslationVector	Reference To TcVnVector3_LREAL [▶ 162]		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.

Further information

The function block `FB_VN_WriteCalibrationResult` saves the calibration results from the PLC asynchronously in a file.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Calibration results

The following results of the calibration functions [▶ 1072] are saved under the specified file path in `JSON` format.

```
// Calibration results
aCameraMatrix            : TcVnMatrix3x3_LREAL;
aDistortionCoefficients  : TcVnArray8_LREAL;
aRotationMatrix          : TcVnMatrix3x3_LREAL;
aTranslationVector       : TcVnVector3_LREAL;
```

File path and format

If `sFilePath` is empty, a file name with the current time is generated and the extension `json` is used.

Application

Saving the calibration results to the `C:\TcVision\CalibrationResult.json` file looks like this, for example:

```
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
```

Related function block: FB_VN_ReadCalibrationResult [▶ 1594].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 370]		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 [▶ 188]	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.

Further information

The function block `FB_VN_WriteContainer` saves a container [▶ 152] from a file asynchronously into the PLC.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Container

The container `ipContainer` is saved as an XML file with the specified file path.

Export format

The export format `eExportFormat` specifies via the enum EtcVnContainerExportFormat [▶ 188] whether the container is to be saved in a human-readable form or serialized:

- `TCVN_CEF_XML`
- `TCVN_CEF_XML_SERIALIZED`
- `TCVN_CEF_CSV`
- `TCVN_CEF_BINARY`

The human-readable form (XML, CSV) has the advantage that entries in the container can be simply checked and edited manually. However, this can lead to rounding inaccuracies. The serialized format has the advantage that no rounding inaccuracies occur. However, the format is not easy to read or edit. The CSV format has the advantage that the data can be easily processed externally in other programs. The binary format is recommended for large amounts of data.

Application

The saving of a container in the serialized XML format in the file `C:\TcVision\Container.xml` looks like this, for example:

```
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
```

Related function block: FB_VN_ReadContainer [▶ 1596].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 411]		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.

Further information

The function block `FB_VN_WriteImage` saves an image [▶ 149] from a file asynchronously into the PLC.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Input image

The format of the input image `ipImage` must match the file format defined in the file path `sFilePath`.

Saving on a rising edge of bWrite

The image is copied internally on a rising edge of the write command. Changes to the image in the following cycle thus have no effects. The image transferred at the time of the rising edge of bWrite is always the one that is saved.

File path and format

The file path `sFilePath` specifies where the input image `ipImage` is saved and which image format should be used for saving. The image format is automatically recognized on the basis of the file extension. If no file name is specified it is automatically generated and contains the current time. If no file extension (e.g. `.jpg`) is specified, the extension `.bmp` and thus the BMP format are automatically used.

The following image formats are supported for saving.

- BMP (`.bmp`)
- PNG (`.png`)
- JPEG (`.jpeg` or `.jpg`)
- TIF (`.tif`)
- TwinCAT serialization (`.tcimg`)

NOTICE

Possibility of errors

Make sure that the selected image format corresponds to the format of the actual input image. Otherwise, errors may occur or the saved images may be faulty.

NOTICE

16-bit images

Only the TIF format and the TwinCAT serialization may be used if 16-bit images are to be saved without losses.

Application

Saving an image to the file `C:\TcVision\Image.bmp` looks like this, for example:

```
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
```

Related function block: FB_VN_ReadImage [▶ 1598] for loading images.

Samples

- Saving images from the PLC [▶ 2834]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 420]		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.

Further information

The function block `FB_VN_WriteMlModel` saves an ML model [▶ 420] from a file asynchronously into the PLC.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

ML model

The ML model `ipMlModel` is saved under the specified file path in binary format.

The saving of a `ITcVnColorModel` represents a special case, since this must still be converted first with the following row into a ML model. Alternatively, Color Model can be replaced by `ITcVnMlModel` with the corresponding functions.

```
hr := ipColorModel.TcQueryInterface(IID_ITcVnMlModel, ADR(ipWriteMlModel));
```

File path and format

If `sFilePath` is empty, a file name with the current time is generated and the extension vmlm is used.

Application

Saving an ML model to the file `C:\TcVision\MlModel.vmlm` looks like this, for example:

```
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
```

Related function block: FB_VN_ReadMlModel [▶ 1600].

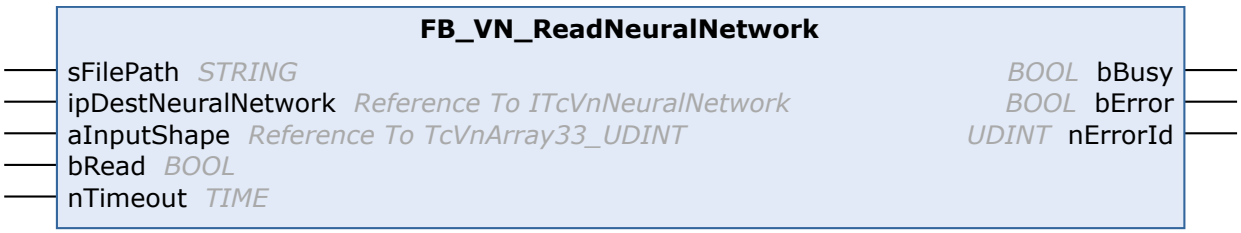
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 430]		Returns the neural network
aInputShape	Reference To TcVnArray33_UDINT [▶ 162]		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.

Further information

The function block `FB_VN_ReadNeuralNetwork` loads a neural network [▶ 430] from a file asynchronously into the PLC. The file must be in the standardized exchange format Open Neural Network Exchange (ONNX), as only this format is supported.

Loading modified neural networks

When loading a neural network, the Vision Service caches the data internally. Therefore, it is not possible to load modified neural networks in the standard system. To do this, adjust the Cache settings [▶ 66] or see the sample Serial loading of images into the PLC [▶ 2835] for an alternative.

Parameter

Some parameters are described centrally in the section File Access Parameters [▶ 1588].

Loaded neural network (return value)

The loaded `ITcVnNeuralNetwork` can be reached via the interface pointer `ipDestNeuralNetwork` after a successful read process and optional test of the execution.

Testing the execution

The optional parameter `aInputShape` can be used to transfer an input format that is used to test the execution of the neural network before loading it into the PLC. This test is helpful, especially for models with dynamic input formats, to ensure that the model can be successfully executed with the specified format. If the execution is not successful, a `HRESULT = ADSERR_DEVICE_INCOMPATIBLE` is returned.

The following information can be used to specify an input format for the test, for example. The execution test can also be disabled with `aInputShape[0] := 0;`.

```
aInputShape[0] := 4;
aInputShape[1] := 1;
aInputShape[2] := 224;
aInputShape[3] := 224;
aInputShape[4] := 3;
```

Application

Loading a neural network, without testing the execution (`aInputShape := 0;`), from the path `C:\TcVision\NeuralNetwork.onnx` looks like this, for example:

```
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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.4 Image Acquisition

This group contains function blocks for acquiring images with Vision devices (GigE Vision cameras [▶ 69] and File Source [▶ 125] objects).

Function blocks

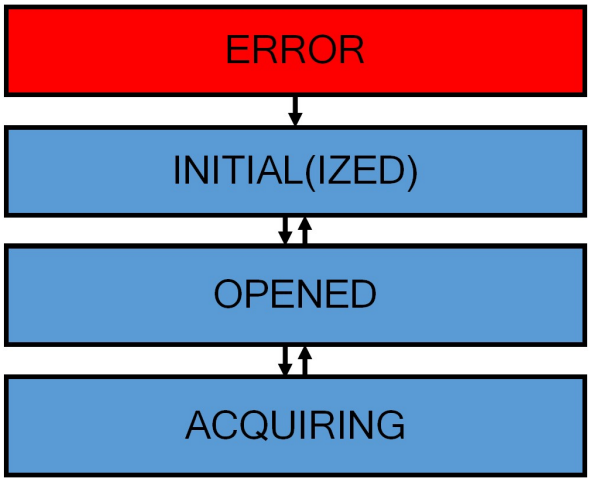
- FB_VN_SimpleCameraControl [▶ 1646] for cameras and file sources
- FB_VN_GevCameraControl [▶ 1621] for GigE Vision cameras
- FB_VN_FileSourceControl [▶ 1613] for file sources

General

The function blocks in this group represent a proxy for the Image Provider TcCOM objects. Accordingly, they need not be called cyclically. Instead, the TcCOM object is controlled via the methods of the function block.

State machine

All function blocks for image acquisition include a state machine to manage the state of the represented device. Essentially, a function block must be placed in the ACQUIRING state in order to receive images continuously. The INITIAL state represents the starting point, after a restart or reset, from which the INITIALIZING can be triggered. From the OPENED state, the communication channel to the device is open in order, for example, to read or write register values.



All states are represented by the enum ETcVnCameraState [▶ 168] and are explained below:

Main states

The main states are stationary states. They essentially indicate whether there is a connection to a Vision device (`OPENED`), whether the device is capturing images (`ACQUIRING`) and whether there is an error with the connection to the device (`ERROR`). You can switch between these states with the methods of the FBs. As a rule, the transition between two main states does not take place immediately, but over at least one cycle. In this intermediate time, the Vision device is in an intermediate state [▶ 1612].

State	Description
ERROR	Error state from which every device can only be reactivated by calling the method `Reset`.
INITIAL	Initial state of each device.
INITIALIZED	Similar to INITIAL, where the camera was manually initialized (e.g. with Force-IP and the initialization commands)
OPENED	There is a connection to the camera, individual images can be triggered and camera registers can be read and written. **This state can be skipped with the method StartAcquisition.**
ACQUIRING	Device is in the acquisition state and is sending images depending on the settings.

Intermediate states

The intermediate states are necessary as most actions cannot be completed immediately (e.g. image acquisition). If the function blocks are in one of these intermediate states, the respective method must be called again in order to complete the transition to the respective main state.

State	Description
INITIALIZING	INITIAL > INITIALIZED
OPENING	INITIAL > OPENED and INITIALIZED > OPENED
STARTACQUISITION	OPENED > ACQUIRING.
STOPACQUISITION	ACQUIRING > OPENED
RESETTINGFEATURES	OPENED > OPENED
TRIGGERING	OPENED > OPENED and ACQUIRING > ACQUIRING
CLOSING	OPENED > INITIALIZED

Methods must be called in intermediate states!

A respective method for state transitions must also be called in the intermediate states. The state transition is only complete and the method no longer needs to be called when a stationary state is detected using the `GetState` method after a corresponding method call. Triggering a method once and subsequently waiting does not result in the desired state transition.

A state machine in the PLC can then be implemented either with a `CASE … OF` structure or with a `IF … ELSIF …` structure.

Samples

- Triggering an image by file name [▶ 2836]

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

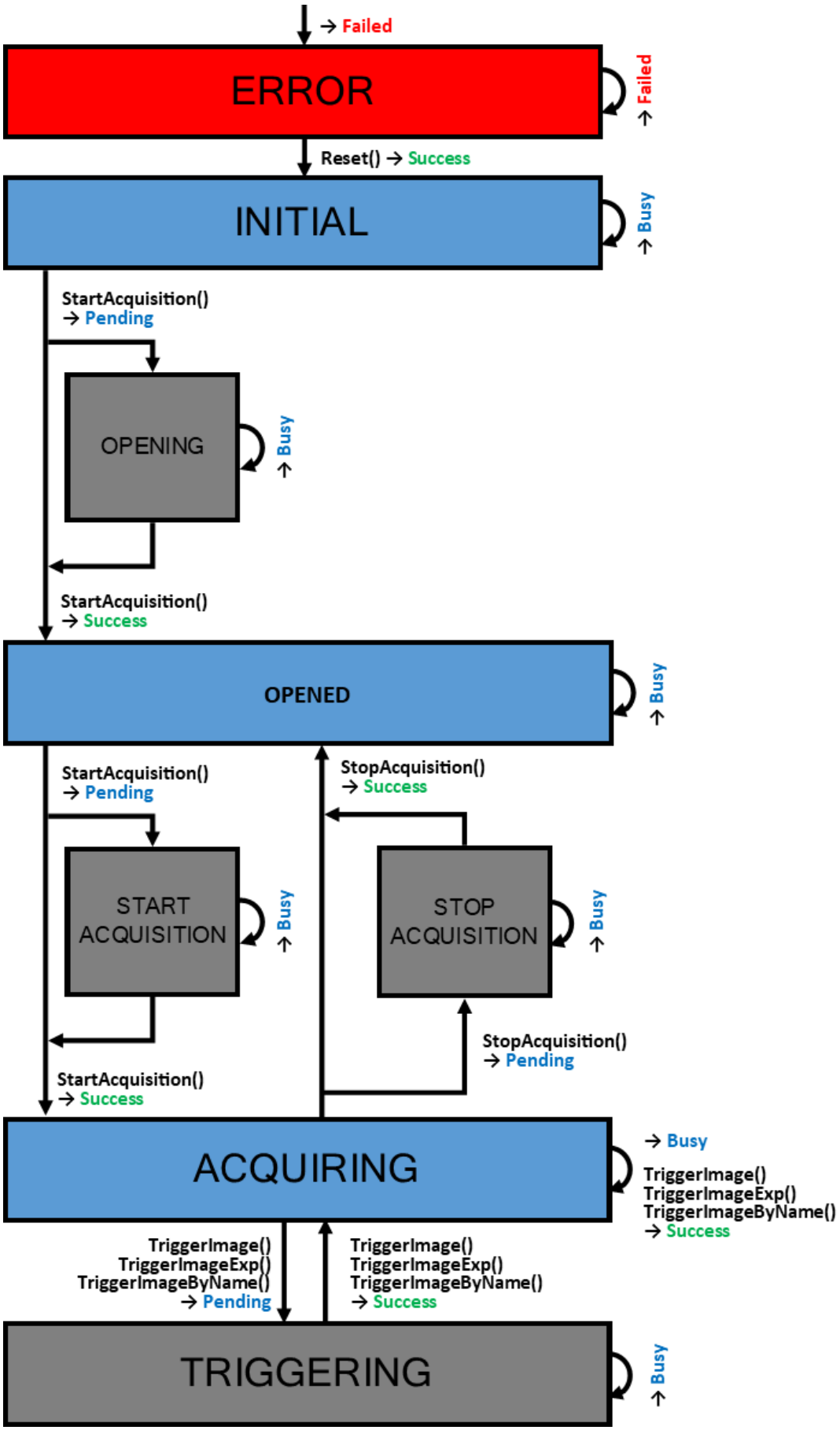
Name	Description
GetCurrentImage [▶ 1616]	Gets the current available image (if any).
GetCurrentImageAndFileName [▶ 1617]	Gets the current available image (if any) and the corresponding file name.
StartAcquisition [▶ 1617]	Activates the File Source. If not in trigger mode, it will start sending images.
StopAcquisition [▶ 1618]	Deactivates the File Source. It will stop sending images.
Reset [▶ 1619]	Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
GetState [▶ 1619]	Gets the current state of the internal camera control state machine.
TriggerImage [▶ 1620]	If in trigger mode, this function triggers the next image in the list.
TriggerImageExp [▶ 1620]	If in trigger mode, this function triggers a specific image in the list.
TriggerImageByName [▶ 1621]	If in trigger mode, this function triggers a specific image in the list by name.

Further information

The function block `FB_VN_FileSourceControl` can control File Source [▶ 125] instances with the full range of functions in the PLC. Alternatively, this is also possible with limited functionality with the function block FB_VN_SimpleCameraControl [▶ 1646].

State machine

The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red.



The following methods can be used to control the state machine:

- GetState [▶ 1619]

- StartAcquisition [▶ 1617]
- StopAcquisition [▶ 1618]
- TriggerImage [▶ 1620]
- TriggerImageByName [▶ 1621]
- Reset [▶ 1619]

If a method is called in a state for which it is not intended, the error code `INVALIDSTATE` (`16#712`) is returned.

Application with continuous image acquisition

A state machine that continuously retrieves received images looks like this, for example:

```
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
```

Application with trigger

A state machine that continuously triggers an image with a special file name looks like this, for example:

```
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
```

As an alternative to the `TriggerImageByName(sFileName)` method, the `TriggerImage()` can be used to always get the next image from the File Source Control list.

Samples

- Triggering an image by file name [▶ 2836]

Related function blocks

- FB_VN_SimpleCameraControl [▶ 1646] for cameras and file sources
- FB_VN_GevCameraControl [▶ 1621] for GigE Vision cameras
- FB_VN_FileSourceControl [▶ 1613] for file sources

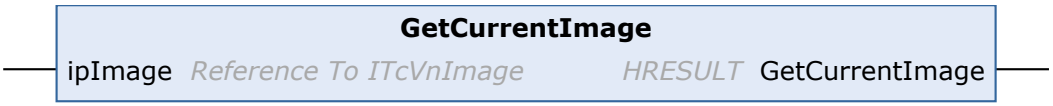
Required License

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 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
```

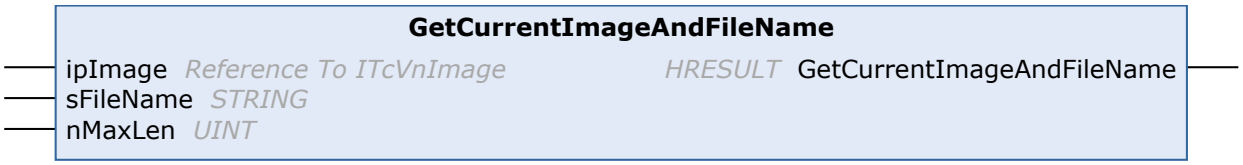
Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 411]	Returns the image

Return value

HRESULT [▶ 141]

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 [▶ 411]	Returns the image
sFileName	STRING	Returns the file name
nMaxLen	UINT	Define the maximum allowed length for the file name

Return value

HRESULT [▶ 141]

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 [▶ 141]

Further information

The method `StartAcquisition` starts the loading of the images. Depending on the trigger mode setting, this may mean providing images directly or waiting for a trigger signal. The image loading can be started from the `INITIAL` and `OPENED` states.

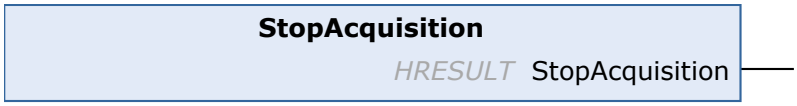
State transitions

With `StartAcquisition` the file source is to be placed in the `ACQUIRING` state. This can take place from the `INITIAL` and `OPENED` states. If the File Source is in `INITIAL`, the state transition to `OPENED` is performed first. As a rule, this takes place via the intermediate state `OPENING`. The transition from `OPENED` to `ACQUIRING` usually takes place via the intermediate state `STARTACQUISITION`.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (`HRESULT`)		**Result state**					
		INITIAL	**OPENING**	**OPENED**	**STARTACQUISITION**	**ACQUIRING**	**ERROR**
Initial state	**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 [▶ 141]

Further information

The method `StopAcquisition` stops the image loading of the File Source.

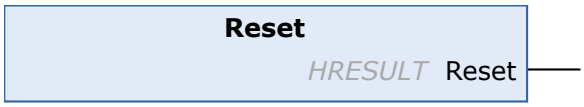
State transitions

With `StopAcquisition`, the File Source is to be brought from the `ACQUIRING` state to the `OPENED` state. As a rule, this doesn't happen immediately, but via the intermediate state `STOPACQUISITION`.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state			
		ACQUIRING	STOPACQUISITION	OPENED	ERROR
Initial state	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 [▶ 141]

Further information

The method `Reset` resets the File Source if it is in an error state.

State transitions

With `Reset` the File Source is to be brought from the `ERROR` state to the `INITIAL` state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the `ERROR` state.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state	
		ERROR	INITIAL
Initial state	ERROR	FAILED	SUCCESS

In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

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

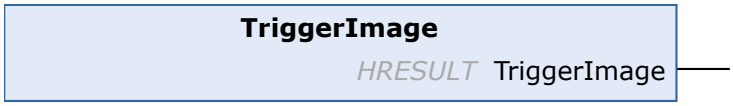
ETcVnCameraState [▶ 168]

Further information

The method `GetState` returns the state of the camera object or File Source object as a value of the enum ETcVnCameraState [▶ 168]. The following states are possible:

- 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 [▶ 141]

Further information

The method TriggerImage triggers an image from the file source.

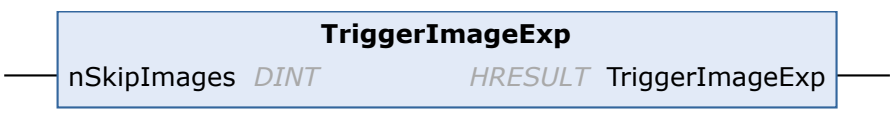
State transitions

With TriggerImage, the File Source is to load only one image while remaining in its current ACQUIRING state. As the image acquisition normally takes several cycles, however, the file source usually briefly enters the intermediate state `TRIGGERING`.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (`HRESULT`)		**Result state**		
		TRIGGERING	**ACQUIRING**	**ERROR**
Initial state	**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

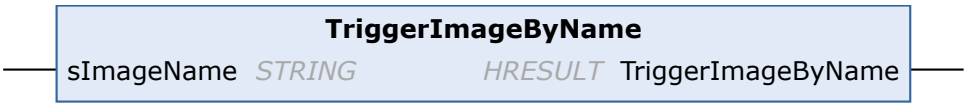
Return value

HRESULT [▶ 141]

Further information

The state transitions of the method `TriggerImageExp` are the same as for TriggerImage [▶ 1620].

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 [▶ 141]

Further information

The state transitions of the method `TriggerImageByName` are the same as for TriggerImage [▶ 1620].

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 [▶ 1628]	Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.
CloseCamera [▶ 1628]	Close the control channel to the camera.
GetCalibPatternRef [▶ 1629]	Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)
GetCameraMatrix [▶ 1630]	Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetCurrentImage [▶ 1630]	Gets the current available image (first in receive queue).
GetCurrentImageUndistorted [▶ 1630]	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 [▶ 1631]	Gets the current available image (first in receive queue) and its corresponding GVSP info.
GetCurrentImageWithGvspInfo Undistorted [▶ 1632]	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 [▶ 1632]	Gets the current available image (first in receive queue) and corresponding timestamps.
GetDistortionCoefficients [▶ 1633]	Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetLastImageFromQueue [▶ 1633]	Gets the last received image from the queue.
GetOmittedImagesNum [▶ 1634]	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 [▶ 1634]	Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
GetState [▶ 1635]	Gets the current state of the internal camera control state machine.
GetTranslationVector [▶ 1635]	Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
InitializeCamera [▶ 1636]	Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
OpenCamera [▶ 1637]	Open a control channel to the camera.
Reset [▶ 1638]	Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
ResetCameraFeatures [▶ 1639]	Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
SetCameraMatrix [▶ 1640]	Sets the camera matrix to the TcVnGevImageProvider TcCOM module
SetDistortionCoefficients [▶ 1641]	Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Name	Description
SetRotationMatrix [▶ 1641]	Sets the rotation matrix to the TcVnGevImageProvider TcCOM module
SetTranslationVector [▶ 1641]	Sets the translation vector to the TcVnGevImageProvider TcCOM module
StartAcquisition [▶ 1642]	Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
StopAcquisition [▶ 1644]	Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
TriggerImage [▶ 1645]	Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

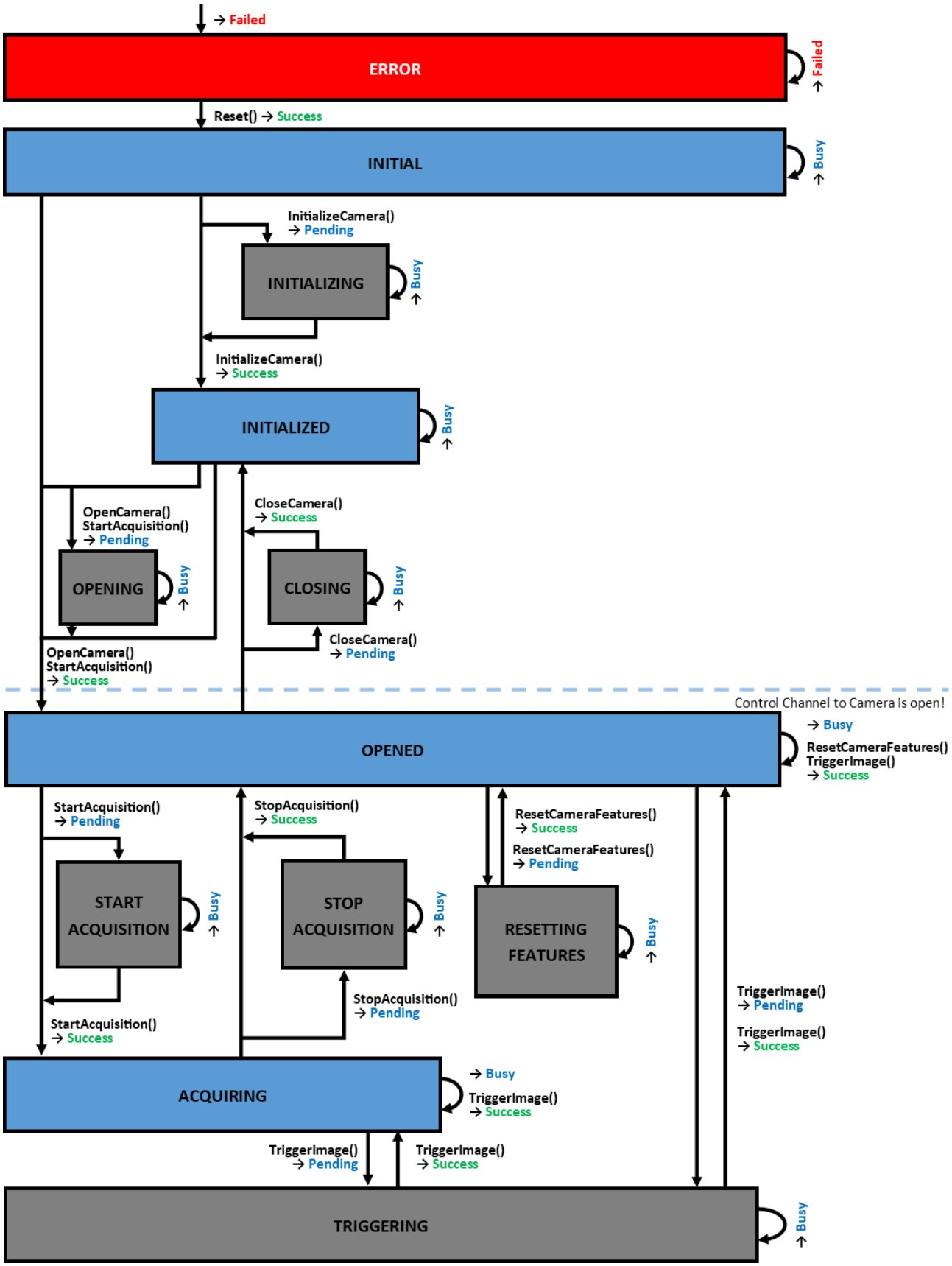
Further information

The function block `FB_VN_GevCameraControl` represents GigE Vision Camera [▶ 69] objects to the full extent in the PLC. Alternatively, the function block FB_VN_SimpleCameraControl [▶ 1646] can also be used for limited access to camera objects.

This function block is not intended to be called directly. In fact, the methods contain all functionalities.

State machine

The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red. The GetState [▶ 1635] method can be used to query the state of the camera. In addition, the methods with which a state transition can be achieved and the return value to which each leads are shown. If Init Commands are present and the Initialization Auto Mode AUTOINIT_AFTER_SO has been selected, the INITIALIZING state in which the automatic initialization takes place is also briefly assumed.



You can query the current state using the GetState method.

The following methods can be used to control the state machine:

- GetState [▶ 1635]
- InitializeCamera [▶ 1636]

- OpenCamera [▶ 1637]
- CloseCamera [▶ 1628]
- StartAcquisition [▶ 1642]
- StopAcquisition [▶ 1644]
- TriggerImage [▶ 1645]
- ResetCameraFeatures [▶ 1639]
- Reset [▶ 1638]

If a method is called in a state for which it is not intended, the error code `INVALIDSTATE` (`16#712`) is returned.

HRESULT

Hex	Dec	Code	Description
16#718	1816	NOTINIT	Camera has not yet been initialized. Initialize the camera either with the method InitializeCamera [▶ 1636] or by setting the TcCOM parameter InitializationAutoMode [▶ 99] to AUTOINIT_SO or AUTOINIT_AFTER_SO.
16#71A	1818	NOINTERFACE	The camera function block is not linked to the camera TcCOM object. Establish this link in the TwinCAT project instance under *Symbol Initialization*.

Application with continuous image acquisition

A state machine for placing a camera in image acquisition mode, handling error states and continuously displaying successfully captured images looks like this, for example:

```
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 this case, the camera can be configured to continuous image acquisition [▶ 2851] or to a hardware trigger [▶ 2851].

Application with software trigger

A state machine for putting a camera into image acquisition mode, handling error states and triggering image acquisition via software trigger looks like this, for example:

```
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 this case, the camera must be configured to a software trigger [▶ 2851] (TriggerSource = software).

Related function blocks

- FB_VN_SimpleCameraControl [▶ 1646] for cameras and file sources
- FB_VN_GevCameraControl [▶ 1621] for GigE Vision cameras
- FB_VN_FileSourceControl [▶ 1613] for file sources

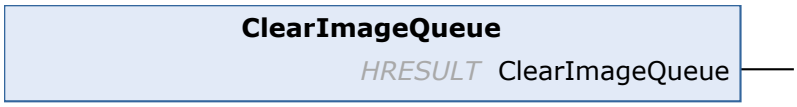
Required License

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 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

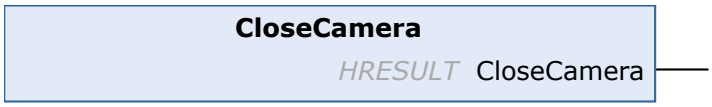
Definition:

```
METHOD ClearImageQueue : HRESULT
```

Return value

HRESULT [▶ 141]

6.1.5.4.2.2 CloseCamera



Close the control channel to the camera.

Syntax

Definition:

```
METHOD CloseCamera : HRESULT
```

Return value

HRESULT [▶ 141]

Further information

The method `CloseCamera` closes an open communication channel to a camera.

State transitions

With `CloseCamera`, the camera is to be brought from the `OPENED` state to the `INITIALIZED` state. As a rule, this transition takes place via the intermediate state `CLOSING`.

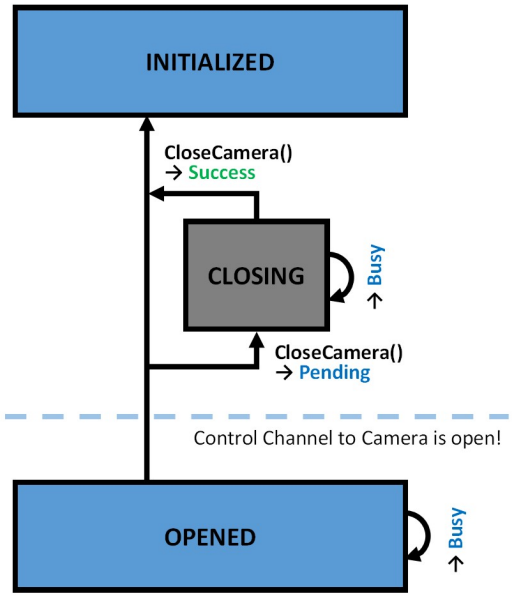


Fig. 19: State transitions with FB_VN_GevCameraControl.CloseCamera

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state			
		OPENED	CLOSING	INITIALIZED	ERROR
Initial state	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
```

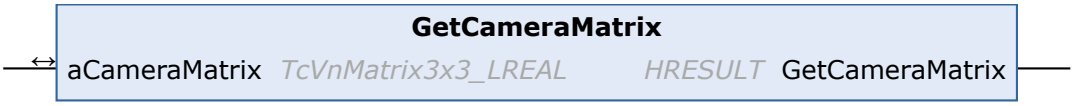
Inputs

Name	Type	Description
ipCalibPatternRef	Reference To ITcVnContainer [▶ 370]	Returns the reference calibration pattern points (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 141]

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
```

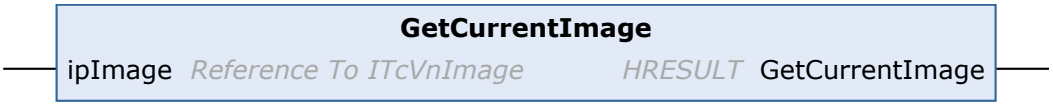
In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 162]	Returns the camera matrix

Return value

HRESULT [▶ 141]

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 [▶ 411]	Returns the image

Return value

HRESULT [▶ 141]

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 [▶ 411]	Returns the undistorted image

Return value

HRESULT [▶ 141]

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 [▶ 411]	Returns the image

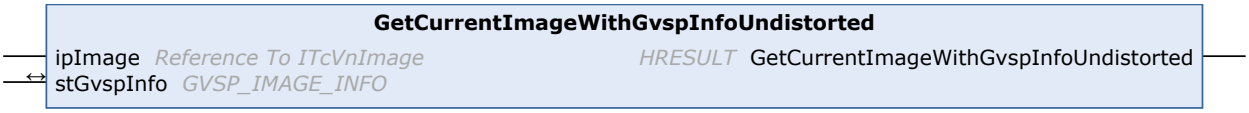
In/Outputs

Name	Type	Description
stGvspInfo	GVSP_IMAGE_INFO [▶ 225]	Returns the GVSP info

Return value

HRESULT [▶ 141]

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 [▶ 411]	Returns the undistorted image

In/Outputs

Name	Type	Description
stGvspInfo	GVSP_IMAGE_INFO [▶ 225]	Returns the GVSP info

Return value

HRESULT [▶ 141]

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 [▶ 411]	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 [▶ 225]	Returns the GVSP info

Return value

HRESULT [▶ 141]

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 [▶ 162]	Returns the distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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
```

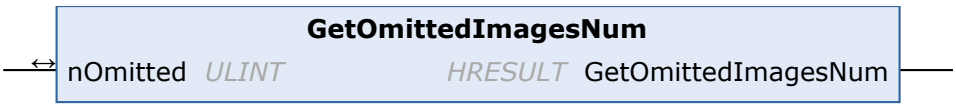
Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 411]	Returns the image

Return value

HRESULT [▶ 141]

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
```

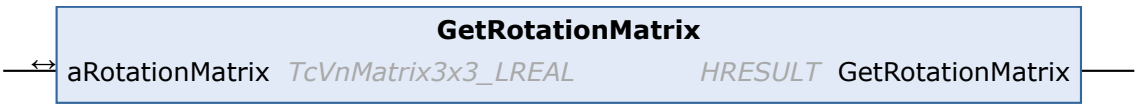
In/Outputs

Name	Type	Description
nOmitted	ULINT	Returns the number of omitted images

Return value

HRESULT [▶ 141]

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 [▶ 162]	Returns the rotation matrix

Return value

HRESULT [▶ 141]

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

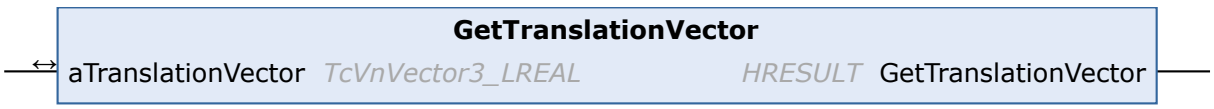
ETcVnCameraState [▶ 168]

Further information

The method `GetState` returns the state of the camera object as a value of the enum ETcVnCameraState [▶ 168]. The following states are possible:

- 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
```

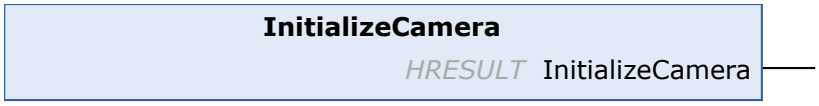
In/Outputs

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 162]	Returns the translation vector

Return value

HRESULT [▶ 141]

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 [▶ 141]

Further information

The method `InitializeCamera` initializes a connected GigE Vision camera just as the parameters in the stored initialization commands have been configured. This method should be used to explicitly control when the camera is initialized. To do this, the `InitializationAutoMode` in the TcCOM parameters of the Image Acquisition object [▶ 99] must be set to `NO_AUTOINIT`. As a result of this, no automatic camera initialization is performed; instead, it must be done manually by means of `InitializeCamera`.

If a different selection was made with `InitializationAutoMode`, an automatic initialization takes place and the `InitializeCamera` method is not required

The initialization of a GigE Vision camera consists of the following steps:

1. Opening the Control Channel to the camera
2. Executing the Force-IP commands (optional)
3. Setting a UserSet or writing Initialization Commands [▶ 87].

The function block assumes the INITIALIZING state during initialization. The function block enters the INITIALIZED state once the initialization is complete. It also assumes this state if the camera connection in the OPENED state is closed by calling the method CloseCamera [▶ 1628].

State transitions

With `InitializeCamera`, the camera is to be brought from the `INITIAL` state to the `INITIALIZED` state. As a rule, this transition takes place via the intermediate state `INITIALIZING`.

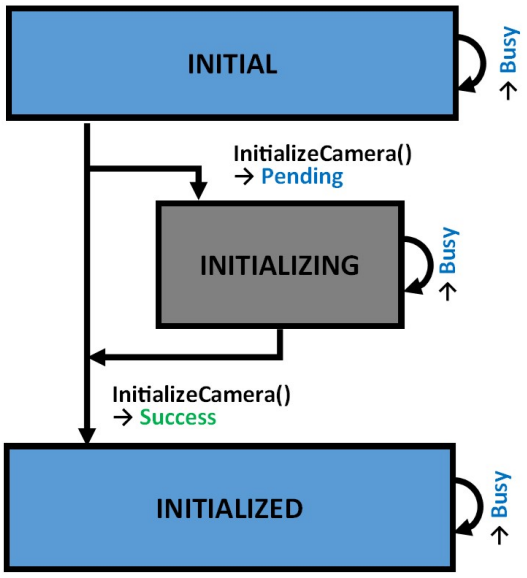
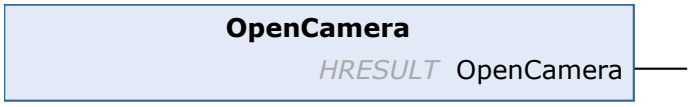


Fig. 20: State transitions with FB_VN_GevCameraControl.InitializeCamera

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state			
		INITIAL	INITIALIZING	INITIALIZED	ERROR
Initial state	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 [▶ 141]

Further information

The method `OpenCamera` opens a communication channel to the camera. In the corresponding `OPENED` state, camera registers can be read and written, images can be acquired and continuous image recording can be started.

If the `InitializationAutoMode` is set to `AUTOINIT_AFTER_SO` in the TcCOM parameters of the Image Acquisition object [▶ 99], the GigE Vision camera is initialized the first time the method OpenCamera is called. Among others, this includes the writing of the Initialization Commands [▶ 87] or the setting of the UserSets. Alternatively, the InitializeCamera [▶ 1636] method can be called in order to perform the initialization manually.

State transitions

With `OpenCamera`, the camera is to be placed in the `OPENED` state. This can take place from the `INITIAL` and `INITIALIZED` states. As a rule, this transition takes place via the intermediate state `OPENING`.

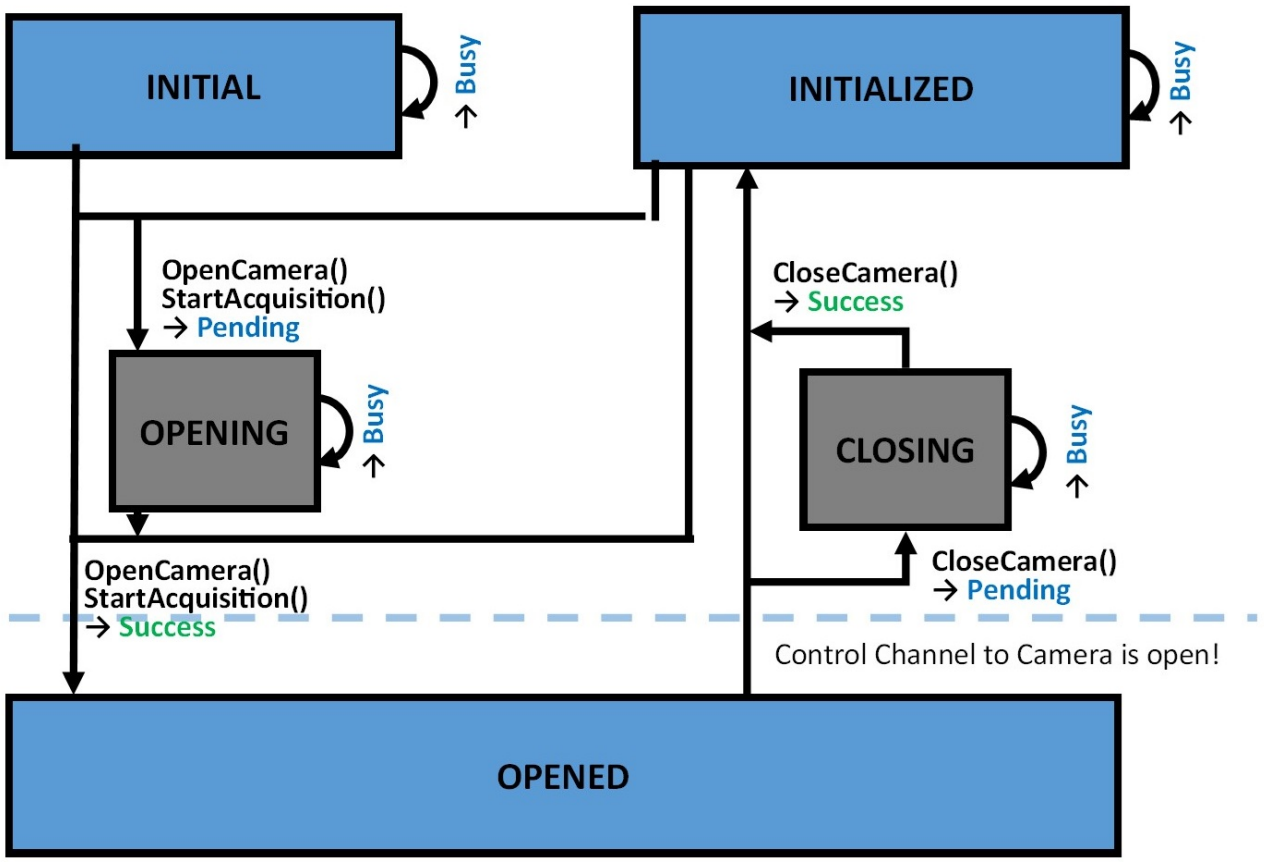
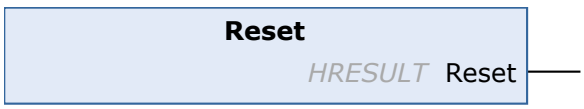


Fig. 21: State transitions with FB_VN_GevCameraControl.OpenCamera

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (`HRESULT`)		**Result state**				
		INITIAL	**INITIALIZED**	**OPENING**	**OPENED**	**ERROR**
Initial state	**INITIAL**	`BUSY`	-	`PENDING`	`SUCCESS`	`FAILED`
	INITIALIZED	-	`BUSY`			
	OPENING		-	`BUSY`		

6.1.5.4.2.18 Reset



Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
METHOD Reset : HRESULT
```

Return value

HRESULT [▶ 141]

Further information

The method `Reset` resets the camera if it is in an error state.

State transitions

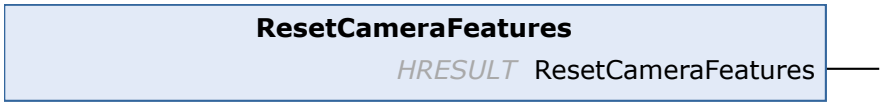
With `Reset` the camera function block should be brought from the `ERROR` state to the `INITIAL` state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the `ERROR` state.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (`HRESULT`)		**Result state**	
		ERROR	**INITIAL**
Initial state	**ERROR**	`FAILED`	`SUCCESS`

In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

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 [▶ 141]

Further information

The method `ResetCameraFeatures` resets the camera parameters to the original state. This means that the Initialization Commands [▶ 87], without the optional Force-IP command, will be rewritten to the camera.

State transitions

With `ResetCameraFeatures` the camera should only reset its parameters while remaining in the `OPENED` state. As a rule, however, the camera briefly enters the intermediate state `RESETTINGFEATURES`, as the parameter reset takes a little time.

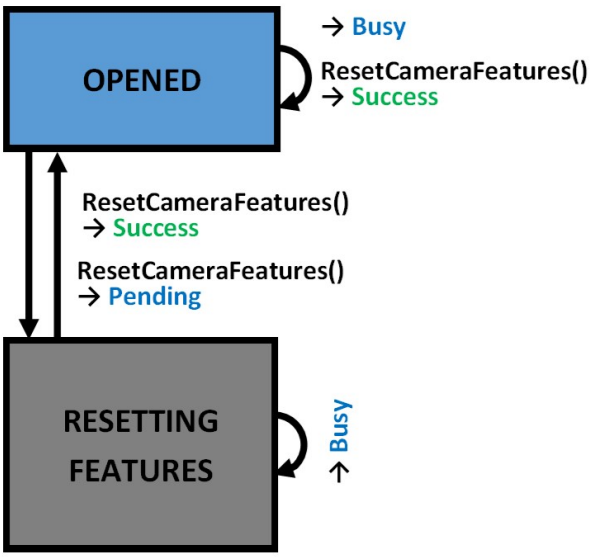
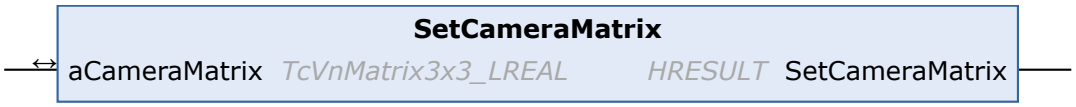


Fig. 22: State transitions with FB_VN_GevCameraControl.ResetCameraFeatures

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state		
		OPENED	RESETTINGFEATURES	ERROR
Initial state	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 [▶ 162]	The camera matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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
```

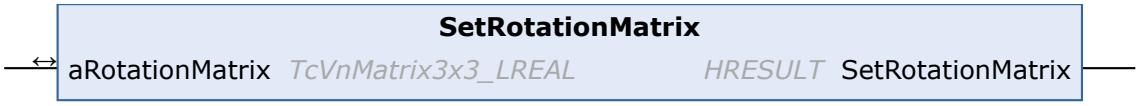
In/Outputs

Name	Type	Description
aDistortionCoefficients	TcVnArray8_LREAL [▶ 162]	The distortion coefficients to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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
```

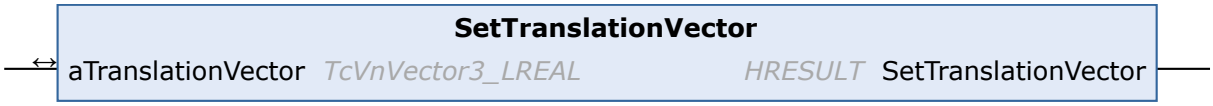
In/Outputs

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 162]	The rotation matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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
```

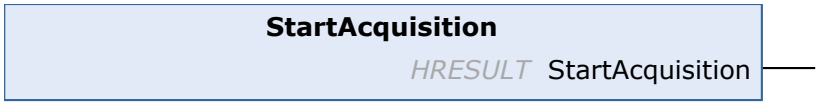
In/Outputs

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 162]	The translation vector to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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 [▶ 141]

Further information

The method `StartAcquisition` starts the image acquisition by a camera. What this means in the individual case depends on the parameterization of the camera. Further information about this can be found in the sample Image acquisition and trigger [▶ 2849]. The image acquisition can be started from the `INITIAL`, `INITIALIZED` and `OPENED` states.

If the camera is still in `INITIAL` or `INITIALIZED`, the method `StartAcquisition` also performs the tasks of the method OpenCamera [▶ 1637]. Usually, the `OPENED` state is not assumed.

State transitions

With `StartAcquisition` the camera is to be placed in the `ACQUIRING` state. This can take place from the `INITIAL`, `INITIALIZED` and `OPENED` states. If the camera is in INITIAL or INITIALIZED, the state transition to OPENED is performed first. As a rule, this takes place via the intermediate state OPENING. The transition from OPENED to ACQUIRING usually takes place via the intermediate state STARTACQUISITION.

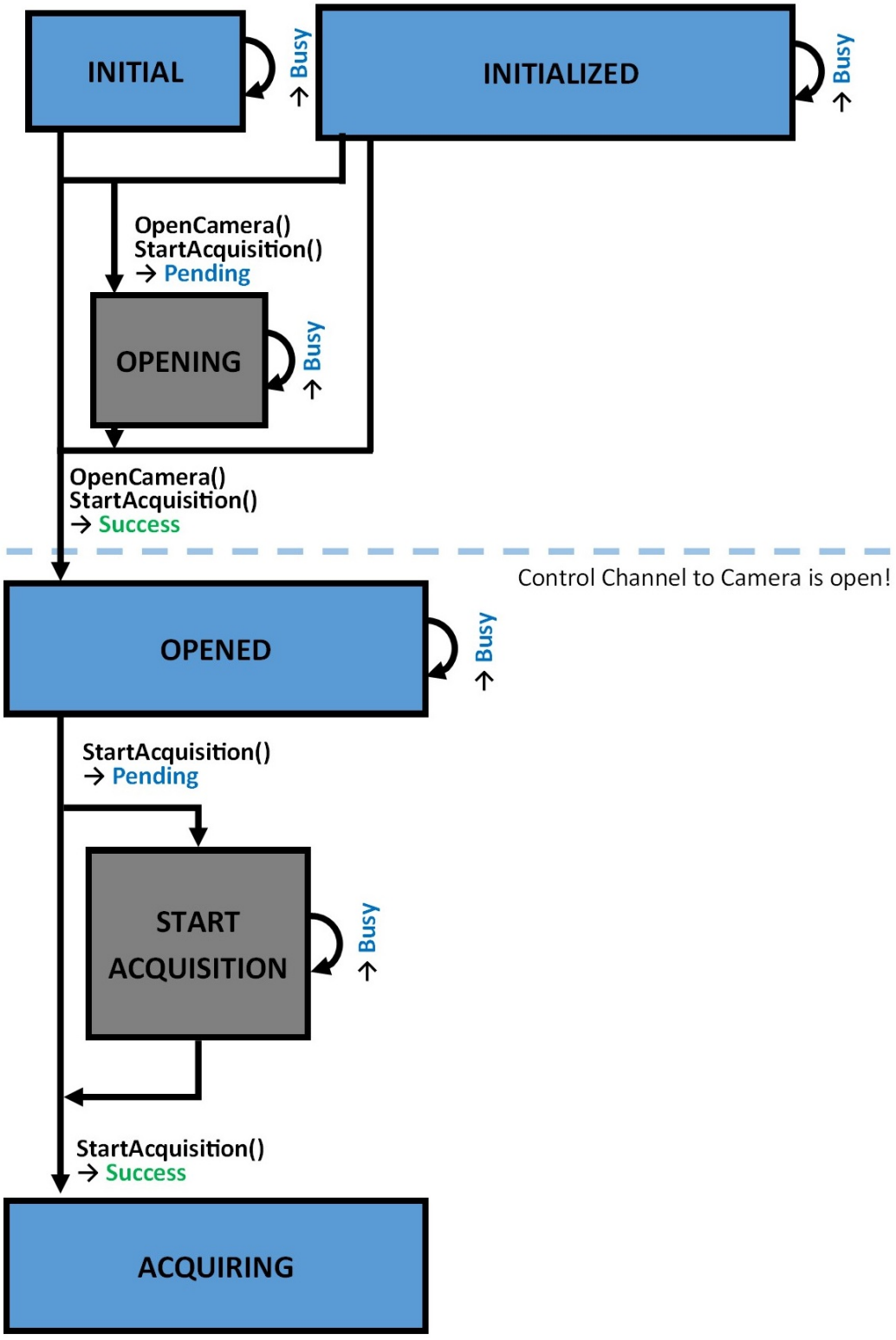
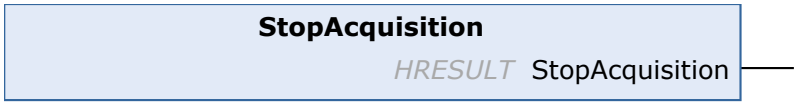


Fig. 23: State transitions with FB_VN_GevCameraControl.StartAcquisition

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state						
		INITIAL	INITIALIZED	OPENING	OPENED	STARTACQUISITION	ACQUIRING	ERROR
Initial state	INITIAL	BUSY	-	PENDING	SUCCESS	-	-	FAILED
	INITIALIZED	-	BUSY	PENDING				
	OPENING		-	BUSY				
	OPENED			-	BUSY	PENDING	SUCCESS	
	STARTACQUISITION				-	BUSY		

6.1.5.4.2.25 StopAcquisition



Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 141]

Further information

The method StopAcquisition stops the image acquisition by the camera.

State transitions

With StopAcquisition, the camera is to be brought from the ACQUIRING state to the OPENED state. As a rule, this doesn't happen immediately, but via the intermediate state STOPACQUISITION.

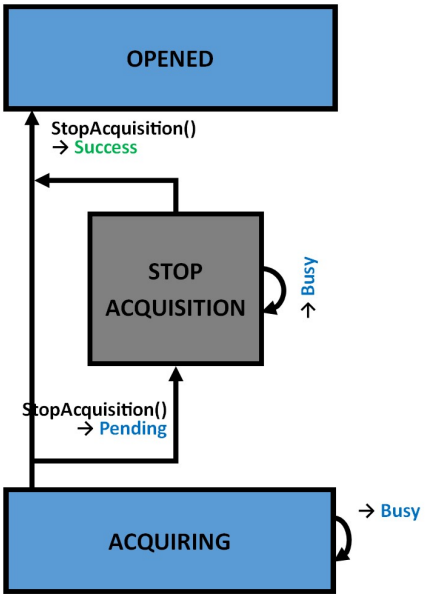
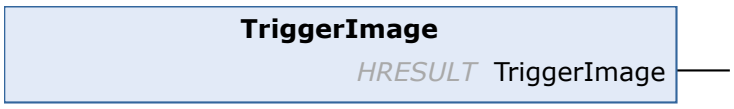


Fig. 24: State transitions with FB_VN_GevCameraControl.StopAcquisition

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state			
		ACQUIRING	STOPACQUISITION	OPENED	ERROR
Initial state	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 [▶ 141]

Further information

Configuring the camera: Image acquisition and trigger [▶ 2849]

State transitions

With `TriggerImage` the camera is to acquire only one image while remaining in the current `ACQUIRING` or `OPENED` state. As the image acquisition normally takes several cycles, the camera usually briefly enters the intermediate state `TRIGGERING`.

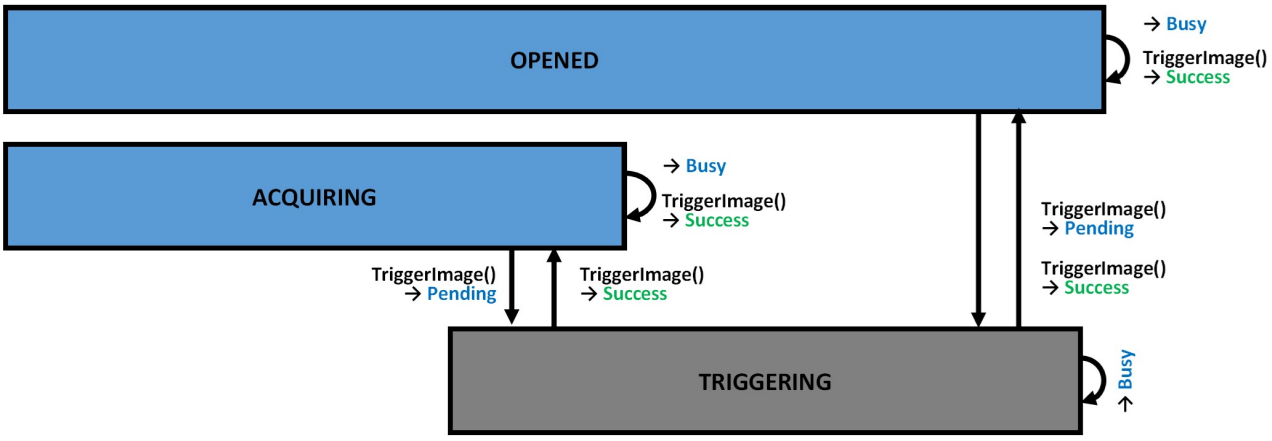


Fig. 25: State transitions with FB_VN_GevCameraControl.TriggerImage

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (HRESULT)		Result state			
		OPENED	ACQUIRING	TRIGGERING	ERROR
Initial state	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 [▶ 1649]	Get the current available image (if any).
GetState [▶ 1650]	Get the current state of the internal camera control state machine.
Reset [▶ 1650]	Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)
StartAcquisition [▶ 1651]	Start the image acquisition.
StopAcquisition [▶ 1651]	Stop the image acquisition.
TriggerImage [▶ 1652]	Trigger the next image.

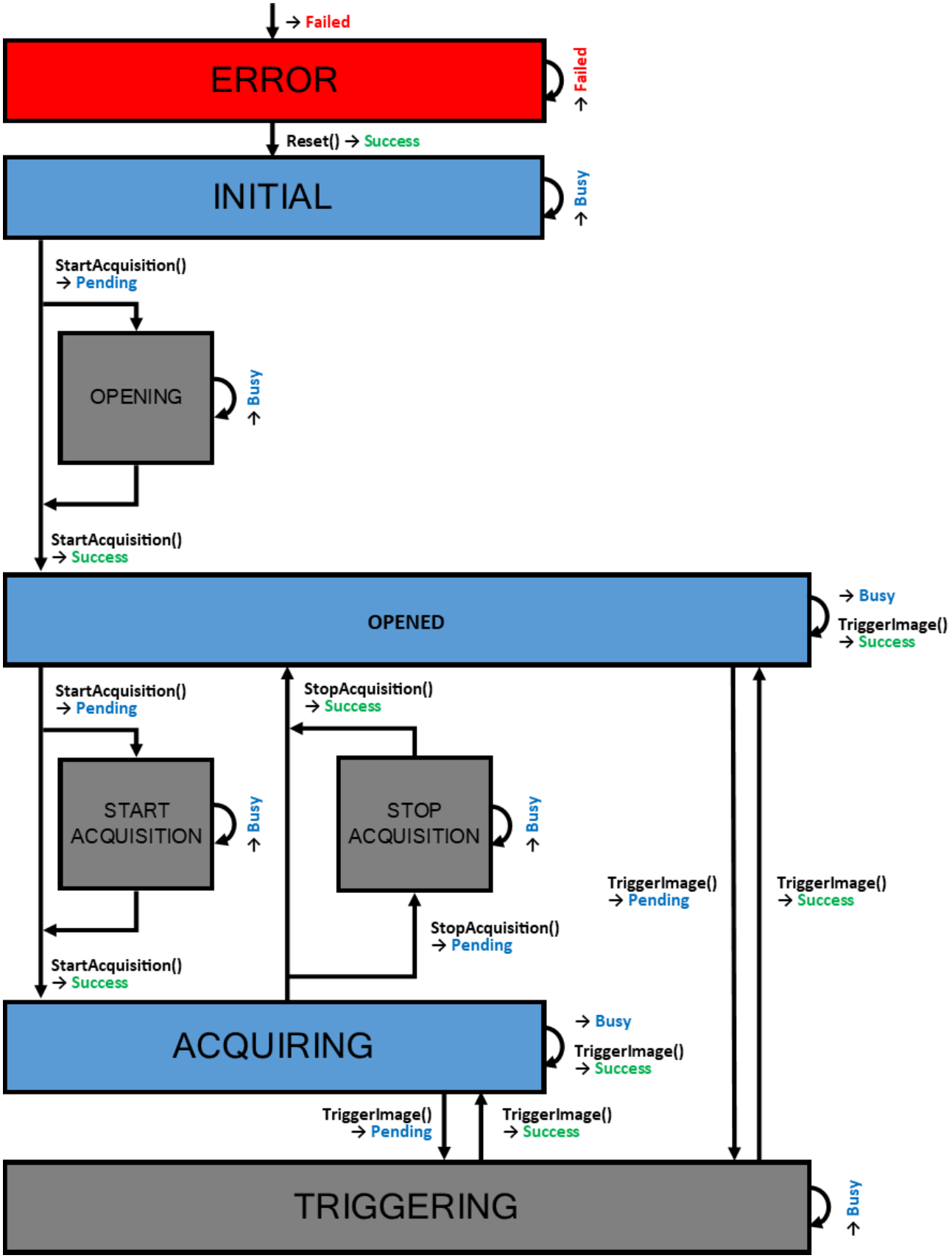
Further information

The function block FB_VN_SimpleCameraControl combines the common features of FB_VN_GevCameraControl [▶ 1621] and FB_VN_FileSourceControl [▶ 1613]. This means that instances of the function block can be linked to both the Image Provider of a camera and the Image Provider of a File Source Control, as in First Steps [▶ 49].

State machine

The following diagram shows the state machine. The main states are marked in blue, the transition states are gray and the error state is red. In contrast to a File Source Control, a camera always also assumes the transition state TCVN_CS_OPENING during the transition from TCVN_CS_INITIAL to TCVN_CS_ACQUIRING and, if necessary, also the main state TCVN_CS_OPENED. If Init Commands are present and the Initialization Auto Mode AUTOINIT_AFTER_SO has been selected, the INITIALIZING state in which the automatic initialization takes place is also briefly assumed. If a transition occurs very quickly, the corresponding state is

not reported to the outside, because the subsequent state has already been reached within a cycle. When linked to a File Source Control, these three states are not assumed. The use of the Trigger Image method from the `TCVN_CS_OPENED` state is only possible with one camera.



The following methods can be used to control the state machine:

- GetState [▶ 1650]

- StartAcquisition [▶ 1651]
- StopAcquisition [▶ 1651]
- TriggerImage [▶ 1652]
- Reset [▶ 1650]

If a method is called in a state for which it is not intended, the error code `INVALIDSTATE` (`16#712`) is returned.

Application with continuous image acquisition

A state machine for placing a camera in image acquisition mode, handling error states and continuously displaying successfully captured images looks like this, for example:

```
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 this case, the camera can be configured to continuous image acquisition [▶ 2851] or to a hardware trigger [▶ 2851].

Application with software trigger

A state machine for putting a camera into image acquisition mode, handling error states and triggering image acquisition via software trigger looks like this, for example:

```
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 this case, the camera must be configured to a software trigger [▶ 2851] (`TriggerSource = Software`).

Samples

In many function samples [▶ 2754], a `FB_VN_SimpleCameraControl` function block is used to load sample images.

Related functions

- FB_VN_SimpleCameraControl [▶ 1646] for cameras and file sources
- FB_VN_GevCameraControl [▶ 1621] for GigE Vision cameras
- FB_VN_FileSourceControl [▶ 1613] for file sources

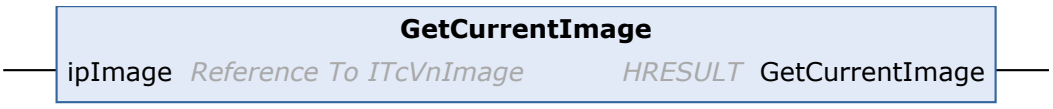
Required License

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 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 [▶ 411]	Returns the image

Return value

HRESULT [▶ 141]

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 [▶ 168]

Further information

The method `GetState` returns the state of the camera object or File Source object as a value of the enum ETcVnCameraState [▶ 168]. The following states are possible:

- 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 [▶ 141]

Further information

The method `Reset` resets the device if it is in an error state.

State transitions

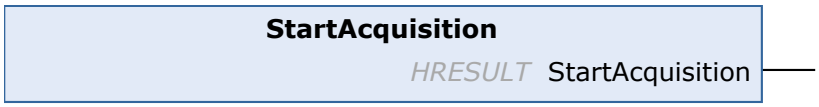
With `Reset` the device is to be brought from the `ERROR` state to the `INITIAL` state. If the reset is successful, this transition takes place immediately, otherwise the function block remains in the `ERROR` state.

All possible state transitions and their return codes are shown in the following transition matrix:

Return Codes (`HRESULT`)		**Result state**	
		ERROR	**INITIAL**
Initial state	**ERROR**	`FAILED`	`SUCCESS`

In case of success, S_OK or S_FALSE (if there was no error to reset) is returned as SUCCESS code.

6.1.5.4.3.4 StartAcquisition



Start the image acquisition.

Syntax

Definition:

```
METHOD StartAcquisition : HRESULT
```

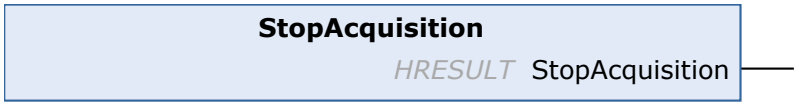
Return value

HRESULT [▶ 141]

Further information

- In connection with a camera, this method corresponds to the StartAcquisition [▶ 1642] method of FB_VN_GevCameraControl [▶ 1621]
- In connection with a File Source Control this method corresponds to the StartAcquisition [▶ 1617] method of FB_VN_FileSourceControl [▶ 1613]

6.1.5.4.3.5 StopAcquisition



Stop the image acquisition.

Syntax

Definition:

```
METHOD StopAcquisition : HRESULT
```

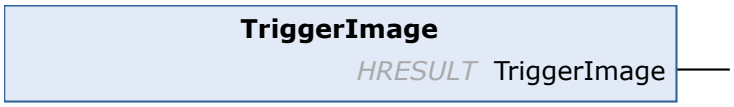
Return value

HRESULT [▶ 141]

Further information

- In connection with a camera, this method corresponds to the StopAcquisition [▶ 1644] method of FB_VN_GevCameraControl [▶ 1621].
- In connection with a File Source Control this method corresponds to the StopAcquisition [▶ 1618] method of FB_VN_FileSourceControl [▶ 1613].

6.1.5.4.3.6 TriggerImage



Trigger the next image.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
```

Return value

HRESULT [▶ 141]

Further information

- In connection with a camera, this method corresponds to the TriggerImage [▶ 1645] method of FB_VN_GevCameraControl [▶ 1621]
- In connection with a File Source Control this method corresponds to the TriggerImage [▶ 1645] method of FB_VN_FileSourceControl [▶ 1613]

6.1.5.5 Image Processing

This group contains function blocks for image processing. In these comprehensive function blocks, the values once set are retained after an execution. Furthermore, the function blocks can also contain several methods.

The functions contained in the chapter Functions [▶ 433], on the other hand, have no internal status information and the variable values are not retained.

Function blocks

- FB_VN_GeneralizedHoughBallard [▶ 1652]
- FB_VN_SSIM [▶ 1657]

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 [▶ 1654]	Detect the template in an image (SetTemplate must have been called before).
SetCannyThreshold [▶ 1654]	Sets the canny edge detection thresholds.
SetInvAccuRatio [▶ 1655]	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 [▶ 1655]	Sets the number of R-table levels.
SetMinDist [▶ 1656]	Sets the minimum distance between the centers of different objects.
SetTemplate [▶ 1656]	Sets the template image to search for in the Detect method.
SetVotesThreshold [▶ 1657]	Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

Further information

The function block `FB_VN_GeneralizedHoughBallard` finds an arbitrary shape in a grayscale image using the generalized Hough transformation. The center positions of the recognized template shapes are determined, whereby different scaling and rotation of the template are not taken into account.

Parameter

Input images

The template and input image must be a 1-channel image with element type USINT (8-bit). First, the template image must be passed using the `SetTemplate` method. After that, the execution can be done by calling the `Detect` method, where the input image is then passed.

Parameters for detection

The following parameters can be set by method call. Otherwise, these default values are used. Since the parameters are already used for internal calculations when the template image is passed, the values must always be set first.

CannyThresholdLow = 50

CannyThresholdHigh = 100

InvAccuRatio = 1.0

Levels = 360

MinDist = 1.0

VotesThreshold = 100

List of center points found (return value)

All center point positions found in the image are returned after calling the `Detect` method in the container `ipPositions`.

Application

The execution consists of 3 steps. First the parameters are set and then the template image is passed. After that it is sufficient to call only the `Detect` method with the images on which to search for the template. Only if parameters or the template are to be changed, the corresponding steps have to be executed again.

```
// 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 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 [▶ 411]	Source image (1 channel, USINT)
ipPositions	Reference To ITcVnContainer [▶ 370]	Returns the centers of the detected template positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 141]

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
```

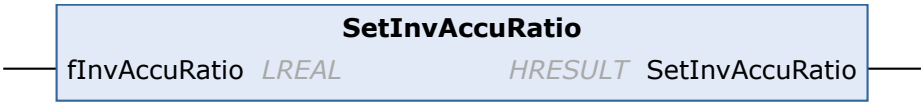
Inputs

Name	Type	Description
nLow	DINT	Low threshold (> 0)
nHigh	DINT	High threshold (> nLow, usually 2 to 3 * nLow)

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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
```

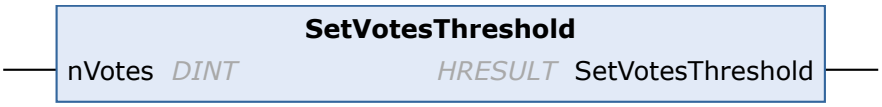
Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 411]	Template image (1 channel, USINT)

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 1658]	Compute the structural similarity (SetReference must have been called before).
SetReference [▶ 1659]	Sets the reference image, used by the SSIM method.

Further information

The function block `FB_VN_SSIM` calculates the structural similarity of two grayscale images. For reference, an interference-free image is assumed.

Parameter

Input images

The reference and input image must be a 1-channel image with element type `USINT` (8-bit). First, the reference image must be passed using the `SetReference` method. After that, the execution can be done by calling the `Compute` method, where the input image is then passed.

Output image (return value)

The output image is of type `Real` and scaled to the value range 0-1. A 1 means agreement and 0 a maximum deviation.

Application

The execution consists of 2 steps. First, the reference image must be set. After that, the `Compute` method can be called, which is passed the current input image to be compared.

For further analysis of the result image e.g. with the F_VN_DetectBlobs [▶ 1178] function it is necessary to convert the image to a `USINT`. The function F_VN_ConvertElementTypeExp [▶ 776] can be used for this purpose. Using the functions F_VN_ImageAverage [▶ 1218] or F_VN_MinPixelValue [▶ 1229] a matching value can be determined directly based on the result image.

```
// 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
```

High memory usage for large images

Since the function performs many calculations with real values in the background, more router memory and computing power is required depending on the image size.

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 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
```

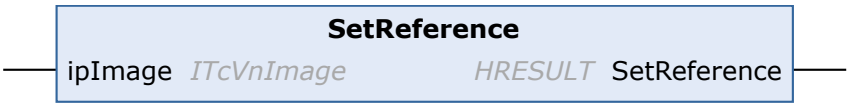
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 411]	Source image (USINT)
ipSSIM	Reference To ITcVnImage [▶ 411]	Returns the SSIM (REAL, values from 0 to 1)

Return value

HRESULT [▶ 141]

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 [▶ 411]	Reference image (USINT)

Return value

HRESULT [▶ 141]

6.2 C++

This chapter contains the TwinCAT Vision API description for the programmatic use of the library in C++.

- Data types [▶ 1659]
  (aliases, constants, arrays, enums, structures)
- Interfaces [▶ 1731]
  (images, containers, etc.)
- Functions [▶ 1890]
  (image processing)
- Function blocks [▶ 2653]
  (communication with cameras and the file system, image processing)

6.2.1 Data Types

There are four data types, each with the prefix "TcVn":

- Alias names [▶ 1660] for certain array types refer to image processing
- Arrays [▶ 1662] with a specific combination of data types and a defined number that are frequently used by the image processing functions.
- Enums [▶ 1662] help with the parameterization of functions and contain the additional prefix "E"
- Structures [▶ 1710] consolidate several related pieces of information (e.g. for geometric descriptions)

In addition, constants [▶ 1660] are available for the description of container types.

6.2.1.1 Aliases

Name	Type
GVCP_REGISTER_ADDRESS	ULONG
TcVnPoint	TcVnPoint2_DINT [▶ 1660]
TcVnPoint2_DINT	TcVnVector2_DINT [▶ 1662]
TcVnPoint2_LREAL	TcVnVector2_LREAL [▶ 1662]
TcVnPoint2_REAL	TcVnVector2_REAL [▶ 1662]
TcVnPoint3_LREAL	TcVnVector3_LREAL [▶ 1662]
TcVnPoint3_REAL	TcVnVector3_REAL [▶ 1662]
TcVnRectangle	TcVnRectangle_DINT [▶ 1729]

6.2.1.2 Constants

The following constants represent types of containers [▶ 152] as `GUID`. These are required, for example, with functions such as CreateContainer [▶ 2149].

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_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* [▶ 1871][10]
TcVnArray3_Point2_REAL	TcVnPoint2_REAL [▶ 1660][3]
TcVnArray33_UDINT	ULONG[33]
TcVnArray4_LREAL	double[4]
TcVnArray4_Point2_REAL	TcVnPoint2_REAL [▶ 1660][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	Unknown
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

6.2.1.5 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 [▶ 1711]	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 [▶ 1711]	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 [▶ 1660]	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 [▶ 1728]	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 [▶ 1660]	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 [▶ 1687]	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 [▶ 1693]	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 [▶ 1689]	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 [▶ 1684]	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;
};
```

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 [▶ 1706]	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 [▶ 1664]	BC_MEDIAN_THRESHOLD	Selects, which of the multi-threshold blob contours should be returned.

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 [▶ 1694]	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 [▶ 1684]	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 [▶ 1690]	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 [▶ 1699]	Pixel encoding
ePixelPackMode	ETcVnPixelPackMode [▶ 1700]	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 [▶ 1660]	Center point
stSize	TcVnSize2_REAL [▶ 1730]	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 do not require a license.

Grouping of interfaces

All interfaces of the TwinCAT Vision library are sorted thematically into the following groups:

- Images [▶ 1841] ()
- Container [▶ 1731] ()
- Machine Learning [▶ 1877] ()
- Miscellaneous [▶ 1880] ()

6.2.2.1 Containers

This group contains interfaces for handling containers [▶ 152].

General information

A container is basically a vector of several elements. The number of elements can be changed dynamically. The corresponding memory in the router memory [▶ 54] is automatically allocated and released for this purpose.

Container types

All elements of a container have the same type. Containers are characterized by a unique GUID based on the type of elements they contain. For ease of use, these GUIDs are defined by constants [▶ 160]. All type designations for containers begin with "ContainerType_Vector_" (with the exception of "ContainerType_String_SINT“). Type requirements are documented for all API elements that make use of containers.

The container type can be determined as follows:

```
ipContainer.GetElementTypeGuid(nTypeGuid);
```

6.2.2.1.1 Function Blocks

This group contains interfaces for the implementation of certain function blocks.

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 [▶ 1732]	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 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* [▶ 1836]	Current element of the 1st container.
ipElement2	ITcVnContainer* [▶ 1836]	Current element of the 2nd container.
ipElement3	ITcVnContainer* [▶ 1836]	Current element of the 3rd container.

Return value

HRESULT [▶ 141]

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 [▶ 1733]	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 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* [▶ 1827]	Current element of the 1st container.
ipElement2	ITcVnForwardIterator* [▶ 1827]	Current element of the 2nd container.
ipElement3	ITcVnForwardIterator* [▶ 1827]	Current element of the 3rd container.

Return value

HRESULT [▶ 141]

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 [▶ 1733]	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 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* [▶ 1836]	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 [▶ 1734]	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 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* [▶ 1827]	Element for which the condition is evaluated.

Return value

bool

6.2.2.1.2 ITcVnAccess

The interface `ITcVnAccess` is for accessing elements of a container [▶ 153] and is provided by an iterator [▶ 1826]. Unlike ITcVnRandomAccess [▶ 1778], only that element to which the associated iterator is pointing in that moment can be addressed.

6.2.2.1.2.1 ITcVnAccess_DINT

Offers an access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1735]	ITcVnAccess_DINT	Gets the value
Set [▶ 1735]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.2 ITcVnAccess_INT

Offers an access interface for INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1736]	ITcVnAccess_INT	Gets the value
Set [▶ 1737]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.3 ITcVnAccess_ITcVnImage

Offers an access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1737]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 1738]	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 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* [▶ 1871]&	Returns the image.

Return value

HRESULT [▶ 141]

6.2.2.1.2.3.2 Set

Sets the image.

Syntax

Definition:

```
HRESULT Set(
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1871]	The image to set.

Return value

HRESULT [▶ 141]

6.2.2.1.2.4 ITcVnAccess_LREAL

Offers an access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1739]	ITcVnAccess_LREAL	Gets the value
Set [▶ 1739]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.5 ITcVnAccess_REAL

Offers an access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1740]	ITcVnAccess_REAL	Gets the value
Set [▶ 1740]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.6 ITcVnAccess_SINT

Offers an access interface for SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1741]	ITcVnAccess_SINT	Gets the value
Set [▶ 1741]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.7 ITcVnAccess_TcVnDMatch

Offers an access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1742]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 1743]	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 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 [▶ 1716]&	Returns the value

Return value

HRESULT [▶ 141]

6.2.2.1.2.7.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnDMatch& stDMatch
)
```

Parameters

Name	Type	Description
stDMatch	TcVnDMatch [▶ 1716]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.8 ITcVnAccess_TcVnKeyPoint

Offers an access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnKeyPoint

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1743]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 1744]	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 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 [▶ 1716]&	Returns the value.

Return value

HRESULT [▶ 141]

6.2.2.1.2.8.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnKeyPoint& stKeyPoint
)
```

Parameters

Name	Type	Description
stKeyPoint	TcVnKeyPoint [▶ 1716]&	The value to set.

Return value

HRESULT [▶ 141]

6.2.2.1.2.9 ITcVnAccess_TcVnPoint2_DINT

Offers an access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1745]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 1745]	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 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 [▶ 1660]&	Returns the value.

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set.

Return value

HRESULT [▶ 141]

6.2.2.1.2.10 ITcVnAccess_TcVnPoint2_LREAL

Offers an access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1746]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 1746]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.11 ITcVnAccess_TcVnPoint2_REAL

Offers an access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1747]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 1747]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.12 ITcVnAccess_TcVnPoint3_LREAL

Offers an access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1748]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 1749]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.13 ITcVnAccess_TcVnPoint3_REAL

Offers an access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1749]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 1750]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.14 ITcVnAccess_TcVnRectangle_DINT

Offers an access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1751]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 1751]	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 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 [▶ 1729]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1729]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.15 ITcVnAccess_TcVnVector2_DINT

Offers an access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1752]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 1752]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.16 ITcVnAccess_TcVnVector2_INT

Offers an access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1753]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 1753]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.17 ITcVnAccess_TcVnVector2_REAL

Offers an access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1754]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 1755]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.18 ITcVnAccess_TcVnVector2_SINT

Offers an access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1755]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 1756]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.19 ITcVnAccess_TcVnVector2_UINT

Offers an access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1757]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 1757]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.20 ITcVnAccess_TcVnVector2_USINT

Offers an access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1758]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 1758]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.21 ITcVnAccess_TcVnVector3_INT

Offers an access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1759]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 1759]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.22 ITcVnAccess_TcVnVector3_REAL

Offers an access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1760]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 1761]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.23 ITcVnAccess_TcVnVector3_SINT

Offers an access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1761]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 1762]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.24 ITcVnAccess_TcVnVector3_UINT

Offers an access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1763]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 1763]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.25 ITcVnAccess_TcVnVector3_USINT

Offers an access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1764]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 1764]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.26 ITcVnAccess_TcVnVector4_DINT

Offers an access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1765]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 1765]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.27 ITcVnAccess_TcVnVector4_INT

Offers an access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1766]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 1767]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.28 ITcVnAccess_TcVnVector4_LREAL

Offers an access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1767]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 1768]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.29 ITcVnAccess_TcVnVector4_SINT

Offers an access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1769]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 1769]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.30 ITcVnAccess_TcVnVector4_UINT

Offers an access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1770]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 1770]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.31 ITcVnAccess_TcVnVector4_USINT

Offers an access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1771]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 1771]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.2.32 ITcVnAccess_UDINT

Offers an access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1772]	ITcVnAccess_UDINT	Gets the value
Set [▶ 1773]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.33 ITcVnAccess_UINT

Offers an access interface for UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1773]	ITcVnAccess_UINT	Gets the value
Set [▶ 1774]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.34 ITcVnAccess_ULINT

Offers an access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1775]	ITcVnAccess_ULINT	Gets the value
Set [▶ 1775]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.35 ITcVnAccess_USINT

Offers an access interface for USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1776]	ITcVnAccess_USINT	Gets the value
Set [▶ 1776]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.2.36 ITcVnAccess_TcVnVector4_REAL

Offers an access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1777]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 1777]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3 ITcVnRandomAccess

The interface `ITcVnRandomAccess` is for accessing any elements of a container [▶ 153] and is provided by an ITcVnRandomAccessIterator [▶ 1833]. Unlike ITcVnAccess [▶ 1735], any element in a container can be addressed.

6.2.2.1.3.1 ITcVnRandomAccess_DINT

Offers a random access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_DINT [▶ 1735]
    ITcVnRandomAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1735]	ITcVnAccess_DINT	Gets the value
Set [▶ 1735]	ITcVnAccess_DINT	Sets the value
GetAt [▶ 1778]	ITcVnRandomAccess_DINT	Gets the value
SetAt [▶ 1779]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.2 ITcVnRandomAccess_INT

Offers a random access interface for INT values

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_INT [▶ 1736]
    ITcVnRandomAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1736]	ITcVnAccess_INT	Gets the value
Set [▶ 1737]	ITcVnAccess_INT	Sets the value
GetAt [▶ 1779]	ITcVnRandomAccess_INT	Gets the value
SetAt [▶ 1780]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.3 ITcVnRandomAccess_ITcVnImage

Offers an random access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_ITcVnImage [▶ 1737]
    ITcVnRandomAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1737]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 1738]	ITcVnAccess_ITcVnImage	Sets the image.
GetAt [▶ 1781]	ITcVnRandomAccess_ITcVnImage	Gets the image.
SetAt [▶ 1781]	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 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* [▶ 1871]&	Returns the image.

Return value

HRESULT [▶ 141]

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* [▶ 1871]	The image to set.

Return value

HRESULT [▶ 141]

6.2.2.1.3.4 ITcVnRandomAccess_LREAL

Offers a random access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_LREAL [▶ 1738]
    ITcVnRandomAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1739]	ITcVnAccess_LREAL	Gets the value
Set [▶ 1739]	ITcVnAccess_LREAL	Sets the value
GetAt [▶ 1782]	ITcVnRandomAccess_LREAL	Gets the value
SetAt [▶ 1783]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.5 ITcVnRandomAccess_REAL

Offers a random access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_REAL [▶ 1739]
    ITcVnRandomAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1740]	ITcVnAccess_REAL	Gets the value
Set [▶ 1740]	ITcVnAccess_REAL	Sets the value
GetAt [▶ 1784]	ITcVnRandomAccess_REAL	Gets the value
SetAt [▶ 1784]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.6 ITcVnRandomAccess_SINT

Offers a random access interface for SINT values

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_SINT [▶ 1741]
    ITcVnRandomAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1741]	ITcVnAccess_SINT	Gets the value
Set [▶ 1741]	ITcVnAccess_SINT	Sets the value
GetAt [▶ 1785]	ITcVnRandomAccess_SINT	Gets the value
SetAt [▶ 1785]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.7 ITcVnRandomAccess_TcVnDMatch

Offers an random access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnDMatch [▶ 1742]
    ITcVnRandomAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1742]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 1743]	ITcVnAccess_TcVnDMatch	Sets the value
GetAt [▶ 1786]	ITcVnRandomAccess_TcVnDMatch	Gets the value
SetAt [▶ 1787]	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 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 [▶ 1716]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1716]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.8 ITcVnRandomAccess_TcVnKeyPoint

Offers a random access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnKeyPoint [▶ 1743]
    ITcVnRandomAccess_TcVnKeyPoint

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1743]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 1744]	ITcVnAccess_TcVnKeyPoint	Sets the value.
GetAt [▶ 1788]	ITcVnRandomAccess_TcVnKeyPoint	Gets the value
SetAt [▶ 1788]	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 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 [▶ 1716]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1716]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.9 ITcVnRandomAccess_TcVnPoint2_DINT

Offers a random access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_DINT [▶ 1744]
    ITcVnRandomAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1745]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 1745]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.
GetAt [▶ 1789]	ITcVnRandomAccess_TcVnPoint2_DINT	Gets the value
SetAt [▶ 1789]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.10 ITcVnRandomAccess_TcVnPoint2_LREAL

Offers a random access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_LREAL [▶ 1745]
    ITcVnRandomAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1746]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 1746]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value
GetAt [▶ 1790]	ITcVnRandomAccess_TcVnPoint2_LREAL	Gets the value
SetAt [▶ 1791]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.11 ITcVnRandomAccess_TcVnPoint2_REAL

Offers a random access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint2_REAL [▶ 1747]
    ITcVnRandomAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1747]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 1747]	ITcVnAccess_TcVnPoint2_REAL	Sets the value
GetAt [▶ 1792]	ITcVnRandomAccess_TcVnPoint2_REAL	Gets the value
SetAt [▶ 1792]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.12 ITcVnRandomAccess_TcVnPoint3_LREAL

Offers a random access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint3_LREAL [▶ 1748]
    ITcVnRandomAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1748]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 1749]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value
GetAt [▶ 1793]	ITcVnRandomAccess_TcVnPoint3_LREAL	Gets the value
SetAt [▶ 1793]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.13 ITcVnRandomAccess_TcVnPoint3_REAL

Offers a random access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnPoint3_REAL [▶ 1749]
    ITcVnRandomAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1749]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 1750]	ITcVnAccess_TcVnPoint3_REAL	Sets the value
GetAt [▶ 1794]	ITcVnRandomAccess_TcVnPoint3_REAL	Gets the value
SetAt [▶ 1795]	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 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 [▶ 1660]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1660]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.14 ITcVnRandomAccess_TcVnRectangle_DINT

Offers a random access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnRectangle_DINT [▶ 1750]
    ITcVnRandomAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1751]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 1751]	ITcVnAccess_TcVnRectangle_DINT	Sets the value
GetAt [▶ 1796]	ITcVnRandomAccess_TcVnRectangle_DINT	Gets the value
SetAt [▶ 1796]	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 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 [▶ 1729]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1729]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.15 ITcVnRandomAccess_TcVnVector2_DINT

Offers a random access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_DINT [▶ 1751]
    ITcVnRandomAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1752]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 1752]	ITcVnAccess_TcVnVector2_DINT	Sets the value.
GetAt [▶ 1797]	ITcVnRandomAccess_TcVnVector2_DINT	Gets the value
SetAt [▶ 1797]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.16 ITcVnRandomAccess_TcVnVector2_INT

Offers a random access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_INT [▶ 1753]
    ITcVnRandomAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1753]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 1753]	ITcVnAccess_TcVnVector2_INT	Sets the value.
GetAt [▶ 1798]	ITcVnRandomAccess_TcVnVector2_INT	Gets the value
SetAt [▶ 1799]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.17 ITcVnRandomAccess_TcVnVector2_REAL

Offers a random access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_REAL [▶ 1754]
    ITcVnRandomAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1754]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 1755]	ITcVnAccess_TcVnVector2_REAL	Sets the value
GetAt [▶ 1800]	ITcVnRandomAccess_TcVnVector2_REAL	Gets the value
SetAt [▶ 1800]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.18 ITcVnRandomAccess_TcVnVector2_SINT

Offers a random access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_SINT [▶ 1755]
    ITcVnRandomAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1755]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 1756]	ITcVnAccess_TcVnVector2_SINT	Sets the value.
GetAt [▶ 1801]	ITcVnRandomAccess_TcVnVector2_SINT	Gets the value
SetAt [▶ 1801]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.19 ITcVnRandomAccess_TcVnVector2_UINT

Offers a random access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_UINT [▶ 1756]
    ITcVnRandomAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1757]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 1757]	ITcVnAccess_TcVnVector2_UINT	Sets the value.
GetAt [▶ 1802]	ITcVnRandomAccess_TcVnVector2_UINT	Gets the value
SetAt [▶ 1803]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.20 ITcVnRandomAccess_TcVnVector2_USINT

Offers a random access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector2_USINT [▶ 1757]
    ITcVnRandomAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1758]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 1758]	ITcVnAccess_TcVnVector2_USINT	Sets the value.
GetAt [▶ 1804]	ITcVnRandomAccess_TcVnVector2_USINT	Gets the value
SetAt [▶ 1804]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.21 ITcVnRandomAccess_TcVnVector3_INT

Offers a random access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_INT [▶ 1759]
    ITcVnRandomAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1759]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 1759]	ITcVnAccess_TcVnVector3_INT	Sets the value
GetAt [▶ 1805]	ITcVnRandomAccess_TcVnVector3_INT	Gets the value
SetAt [▶ 1805]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.22 ITcVnRandomAccess_TcVnVector3_REAL

Offers a random access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_REAL [▶ 1760]
    ITcVnRandomAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1760]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 1761]	ITcVnAccess_TcVnVector3_REAL	Sets the value
GetAt [▶ 1806]	ITcVnRandomAccess_TcVnVector3_REAL	Gets the value
SetAt [▶ 1807]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.23 ITcVnRandomAccess_TcVnVector3_SINT

Offers a random access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_SINT [▶ 1761]
    ITcVnRandomAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1761]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 1762]	ITcVnAccess_TcVnVector3_SINT	Sets the value
GetAt [▶ 1808]	ITcVnRandomAccess_TcVnVector3_SINT	Gets the value
SetAt [▶ 1808]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.24 ITcVnRandomAccess_TcVnVector3_UINT

Offers a random access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_UINT [▶ 1762]
    ITcVnRandomAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1763]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 1763]	ITcVnAccess_TcVnVector3_UINT	Sets the value
GetAt [▶ 1809]	ITcVnRandomAccess_TcVnVector3_UINT	Gets the value
SetAt [▶ 1809]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.25 ITcVnRandomAccess_TcVnVector3_USINT

Offers a random access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector3_USINT [▶ 1763]
    ITcVnRandomAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1764]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 1764]	ITcVnAccess_TcVnVector3_USINT	Sets the value
GetAt [▶ 1810]	ITcVnRandomAccess_TcVnVector3_USINT	Gets the value
SetAt [▶ 1811]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.26 ITcVnRandomAccess_TcVnVector4_DINT

Offers a random access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_DINT [▶ 1765]
    ITcVnRandomAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1765]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 1765]	ITcVnAccess_TcVnVector4_DINT	Sets the value
GetAt [▶ 1812]	ITcVnRandomAccess_TcVnVector4_DINT	Gets the value
SetAt [▶ 1812]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.27 ITcVnRandomAccess_TcVnVector4_INT

Offers a random access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_INT [▶ 1766]
    ITcVnRandomAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1766]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 1767]	ITcVnAccess_TcVnVector4_INT	Sets the value
GetAt [▶ 1813]	ITcVnRandomAccess_TcVnVector4_INT	Gets the value
SetAt [▶ 1813]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.28 ITcVnRandomAccess_TcVnVector4_LREAL

Offers a random access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_LREAL [▶ 1767]
    ITcVnRandomAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1767]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 1768]	ITcVnAccess_TcVnVector4_LREAL	Sets the value
GetAt [▶ 1814]	ITcVnRandomAccess_TcVnVector4_LREAL	Gets the value
SetAt [▶ 1815]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.29 ITcVnRandomAccess_TcVnVector4_SINT

Offers a random access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_SINT [▶ 1768]
    ITcVnRandomAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1769]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 1769]	ITcVnAccess_TcVnVector4_SINT	Sets the value
GetAt [▶ 1816]	ITcVnRandomAccess_TcVnVector4_SINT	Gets the value
SetAt [▶ 1816]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.30 ITcVnRandomAccess_TcVnVector4_UINT

Offers a random access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_UINT [▶ 1769]
    ITcVnRandomAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1770]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 1770]	ITcVnAccess_TcVnVector4_UINT	Sets the value
GetAt [▶ 1817]	ITcVnRandomAccess_TcVnVector4_UINT	Gets the value
SetAt [▶ 1817]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.31 ITcVnRandomAccess_TcVnVector4_USINT

Offers a random access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_USINT [▶ 1771]
    ITcVnRandomAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1771]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 1771]	ITcVnAccess_TcVnVector4_USINT	Sets the value
GetAt [▶ 1818]	ITcVnRandomAccess_TcVnVector4_USINT	Gets the value
SetAt [▶ 1819]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.3.32 ITcVnRandomAccess_UDINT

Offers a random access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_UDINT [▶ 1772]
    ITcVnRandomAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1772]	ITcVnAccess_UDINT	Gets the value
Set [▶ 1773]	ITcVnAccess_UDINT	Sets the value
GetAt [▶ 1820]	ITcVnRandomAccess_UDINT	Gets the value
SetAt [▶ 1820]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.33 ITcVnRandomAccess_UINT

Offers a random access interface for UINT values

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_UINT [▶ 1773]
    ITcVnRandomAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1773]	ITcVnAccess_UINT	Gets the value
Set [▶ 1774]	ITcVnAccess_UINT	Sets the value
GetAt [▶ 1821]	ITcVnRandomAccess_UINT	Gets the value
SetAt [▶ 1821]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.34 ITcVnRandomAccess_ULINT

Offers a randowm access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_ULINT [▶ 1774]
    ITcVnRandomAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1775]	ITcVnAccess_ULINT	Gets the value
Set [▶ 1775]	ITcVnAccess_ULINT	Sets the value
GetAt [▶ 1822]	ITcVnRandomAccess_ULINT	Gets the value
SetAt [▶ 1823]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.35 ITcVnRandomAccess_USINT

Offers a random access interface for USINT values

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_USINT [▶ 1775]
    ITcVnRandomAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1776]	ITcVnAccess_USINT	Gets the value
Set [▶ 1776]	ITcVnAccess_USINT	Sets the value
GetAt [▶ 1824]	ITcVnRandomAccess_USINT	Gets the value
SetAt [▶ 1824]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.1.3.36 ITcVnRandomAccess_TcVnVector4_REAL

Offers a random access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnAccess_TcVnVector4_REAL [▶ 1776]
    ITcVnRandomAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Get [▶ 1777]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 1777]	ITcVnAccess_TcVnVector4_REAL	Sets the value
GetAt [▶ 1825]	ITcVnRandomAccess_TcVnVector4_REAL	Gets the value
SetAt [▶ 1825]	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 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 [▶ 1662]&	Returns the value

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The value to set

Return value

HRESULT [▶ 141]

6.2.2.1.4 Iterators

This group contains interfaces for the handling of iterators in order to access elements of containers [▶ 152].

6.2.2.1.4.1 ITcVnBidirectionalIterator

Offers an interface for a bidirectional iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnIteratorBase [▶ 1830]
    ITcVnForwardIterator [▶ 1827]
      ITcVnBidirectionalIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1831]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1831]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1831]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1832]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1828]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1829]	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 [▶ 1829]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1829]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 1827]	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 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 [▶ 141]

6.2.2.1.4.2 ITcVnForwardIterator

Offers an interface for a forward iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnIteratorBase [▶ 1830]
    ITcVnForwardIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1831]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1831]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1831]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1832]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1828]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1829]	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 [▶ 1829]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1829]	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 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* [▶ 1827]	Iterator interface to compare with

Return value

HRESULT [▶ 141]

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* [▶ 1836]*	Returns the container interface.

Return value

HRESULT [▶ 141]

6.2.2.1.4.2.3 Increment

Increments the iterator. (Alternatively use F_VN_IncrementIterator.)

Syntax

Definition:

```
HRESULT Increment()
```

Return value

HRESULT [▶ 141]

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* [▶ 1836]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 141]

6.2.2.1.4.3 ITcVnIteratorBase

Offers a base interface for iterators

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnIteratorBase

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1831]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1831]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1831]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1832]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.1.4.3.6 SetToEnd

Sets the iterator to the past-the-end element.

Syntax

Definition:

```
HRESULT SetToEnd()
```

Return value

HRESULT [▶ 141]

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 [▶ 1885]
  ITcVnIteratorCopyCreator

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
Create [▶ 1832]	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 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* [▶ 1827]*	Returns the created iterator.

Return value

HRESULT [▶ 141]

6.2.2.1.4.5 ITcVnRandomAccessIterator

Offers an interface for a random access iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnIteratorBase [▶ 1830]
    ITcVnForwardIterator [▶ 1827]
      ITcVnBidirectionalIterator [▶ 1826]
        ITcVnRandomAccessIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1830]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1831]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1831]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1831]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1832]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1828]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1829]	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 [▶ 1829]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1829]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 1827]	ITcVnBidirectionalIterator	Decrements the iterator.
CheckIfGreaterThan [▶ 1834]	ITcVnRandomAccessIterator	Checks if the iterator is greater than another iterator.
CheckIfLessThan [▶ 1835]	ITcVnRandomAccessIterator	Checks if the iterator is less than another iterator.
GetContainerAt [▶ 1835]	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 [▶ 1836]	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 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* [▶ 1833]	Iterator interface to compare with.

Return value

HRESULT [▶ 141]

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* [▶ 1833]	Iterator interface to compare with.

Return value

HRESULT [▶ 141]

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* [▶ 1836]*	Returns the pointer to the container interface.

Return value

HRESULT [▶ 141]

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* [▶ 1836]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 141]

6.2.2.1.5 ITcVnContainer

Offers an interface for an object container.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnContainer

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBasicContainer [▶ 1837]	ITcVnContainer	Checks if the container contains only basic (non-container) elements.
CheckIfEmpty [▶ 1837]	ITcVnContainer	Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)
GetBidirectionalIterator [▶ 1838]	ITcVnContainer	Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.
GetElementNum [▶ 1838]	ITcVnContainer	Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)
GetElementSize [▶ 1838]	ITcVnContainer	Gets the size (in byte) of each element in the container.
GetElementTypeGuid [▶ 1839]	ITcVnContainer	Gets the GUID of the container elements.
GetExportSize [▶ 1839]	ITcVnContainer	Gets combined size (in byte) of all elements in the container.
GetForwardIterator [▶ 1840]	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 [▶ 1840]	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 [▶ 1840]	ITcVnContainer	Gets GUID of the container.
GetTypeName [▶ 1841]	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 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 [▶ 141]

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 [▶ 141]

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* [▶ 1826]*	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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* [▶ 1827]*	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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* [▶ 1833]*	Returns the iterator interface.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2 Images

This group contains interfaces for recording and handling images [▶ 149].

6.2.2.2.1 Acquisition

This group contains interfaces for handling the image acquisition.

6.2.2.2.1.1 ITcIoFileImageAcquisition

Interface for file image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcIoImageAcquisition [▶ 1845]
    ITcIoFileImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1846]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1846]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1847]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1847]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1847]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1847]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1848]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1848]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 1842]	ITcIoFileImageAcquisition	Checks the connection
TriggerImage [▶ 1842]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 1843]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.1.2 ITcIoFileImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcIoImageRecv [▶ 1848]
    ITcIoFileImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 1849]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 1849]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.
ReceiveImage [▶ 1844]	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 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* [▶ 1873]	Interface pointer to the image.
sFileName	PCHAR	Returns the filename to the image.
hrAcquisitionResult	HRESULT [▶ 141]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 141]

6.2.2.2.1.3 ITcIoGevImageAcquisition

Interface for GigE Vision image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcIoImageAcquisition [▶ 1845]
    ITcIoGevImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1846]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1846]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1847]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1847]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1847]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1847]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1848]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1848]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 1845]	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 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 [▶ 1709]&	The internally assumed state of the camera.
eActualState	GEV_CAMERA_STATE [▶ 1709]&	The actually observable state of the camera.

Return value

HRESULT [▶ 141]

6.2.2.2.1.4 ITcIoImageAcquisition

Interface for image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcIoImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1846]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1846]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1847]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1847]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1847]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1847]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1848]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1848]	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 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* [▶ 1848]	Pointer to an interface containing the callback function.

Return value

HRESULT [▶ 141]

6.2.2.2.1.4.2 UnregisterReceiver

Unregister callback interface.

Syntax

Definition:

```
HRESULT UnregisterReceiver()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.1.4.5 StartAcquisition

Start the image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 141]

6.2.2.2.1.4.6 StopAcquisition

Stop the image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.2.2.2.1.4.8 InitializeCamera

Send initialization commands to the camera.

Syntax

Definition:

```
HRESULT InitializeCamera()
```

Return value

HRESULT [▶ 141]

6.2.2.2.1.5 ITcIoImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcIoImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 1849]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 1849]	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 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* [▶ 1873]	Interface pointer to the image.
hrAcquisitionResult	HRESULT [▶ 141]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 141]

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 [▶ 141]	HRESULT indicating the status of the performed operation.

Return value

HRESULT [▶ 141]

6.2.2.2.1.6 ITcVnFileImageProvider

Interface for an image provider for images from outside realtime environment.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageProvider [▶ 1853]
    ITcVnFileImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1854]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1854]	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 [▶ 1855]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1855]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1855]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1856]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1856]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1856]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1856]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1856]	ITcVnImageProvider	Set camera to initial state.
GetCurrentImageAndFileName [▶ 1851]	ITcVnFileImageProvider	Get the current image by detaching its internal reference.
TriggerImage [▶ 1851]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 1851]	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 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* [▶ 1871]*	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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.1.7 ITcVnGevImageProvider

Interface for a GigE Vision image provider.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageProvider [▶ 1853]
    ITcVnGevImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1854]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1854]	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 [▶ 1855]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1855]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1855]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1856]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1856]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1856]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1856]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1856]	ITcVnImageProvider	Set camera to initial state.
GetCurrentGevImage [▶ 1853]	ITcVnGevImageProvider	Gets the current GigE Vision image by detaching its internal reference.
GetCurrentImageWithGvspInfo [▶ 1853]	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 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* [▶ 1868]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 141]

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* [▶ 1871]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.
stGvspInfo	GVSP_IMAGE_INFO [▶ 1710]&	Contains useful meta information like image id, time stamp etc.

Return value

HRESULT [▶ 141]

6.2.2.2.1.8 ITcVnImageProvider

Interface for an image provider.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1854]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1854]	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 [▶ 1855]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1855]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1855]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1856]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1856]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1856]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1856]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1856]	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 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 [▶ 141]

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 [▶ 141]

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* [▶ 1871]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 141]

6.2.2.2.1.8.4 GetError

Gets result of the last operation performed.

Syntax

Definition:

```
HRESULT GetError()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.1.8.7 StartAcquisition

Start image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 141]

6.2.2.2.1.8.8 StopAcquisition

Stop image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.2.2.2.1.8.10 InitializeCamera

Set camera to initial state.

Syntax

Definition:

```
HRESULT InitializeCamera()
```

Return value

HRESULT [▶ 141]

6.2.2.2.2 Export

This group contains interfaces for handling the data export in connection with images.

6.2.2.2.2.1 ITcVnBitmapExport

Interface for exporting an image as a Windows Bitmap (BMP).

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnBitmapExport

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 1857]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 1858]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.2.2.2 ITcVnBitmapExportNotification

Interface required for sending displayable images as ADS notifications

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnBitmapExportNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 1859]	ITcVnBitmapExportNotification	Gets the timestamp of the latest image change.
GetBitmapImageRpcUnlocked [▶ 1859]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 1885]
  ITcVnBitmapExport [▶ 1857]
    ITcVnBitmapExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 1857]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 1858]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.
GetBitmapImageRpcUnlocked [▶ 1860]	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 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 [▶ 141]

6.2.2.2.2.4 ITcVnHistogramExport

Interface for exporting a histogram for an image as an array.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnHistogramExport

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetHistogramSize [▶ 1862]	ITcVnHistogramExport	Gets the size of the histogram array.
GetHistogramArray [▶ 1862]	ITcVnHistogramExport	Gets the histogram array.
GetCustomHistogramSize [▶ 1862]	ITcVnHistogramExport	Gets the size of a custom histogram array.
GetCustomHistogramArray [▶ 1863]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.2.5 ITcVnTiffExport

Interface for exporting an image as tiff.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnTiffExport

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 1864]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 1864]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.2.2.6 ITcVnTiffExportNotification

Interface for sending displayable images as tiff via ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnTiffExportNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 1865]	ITcVnTiffExportNotification	Gets the timestamp of the latest image change.
GetTiffImageRpcUnlocked [▶ 1866]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 1885]
  ITcVnTiffExport [▶ 1863]
    ITcVnTiffExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 1864]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 1864]	ITcVnTiffExport	Export the image as tiff into a given buffer.
GetTiffImageRpcUnlocked [▶ 1867]	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 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 [▶ 141]

6.2.2.2.3 Import

This group contains interfaces for handling the data import in connection with images.

6.2.2.2.3.1 ITcVnTiffImport

Interface for importing a tiff image.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnTiffImport

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
ImportTiffImage [▶ 1867]	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 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 [▶ 141]

6.2.2.2.4 ITcIoGevImage

Interface for GigE Vision images.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageBase [▶ 1873]
    ITcIoGevImage

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1874]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1874]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1875]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1875]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1875]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1876]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1876]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1877]	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 [▶ 1869]	ITcIoGevImage	Check if image data is a copy.
DecreaseHeight [▶ 1869]	ITcIoGevImage	Sets image height to newHeight if it is smaller than the current value.
GetBlockId [▶ 1869]	ITcIoGevImage	Gets the GVSP block ID.
GetGevStatus [▶ 1870]	ITcIoGevImage	Gets the block status.
GetGvspImageInfo [▶ 1870]	ITcIoGevImage	Gets a pointer to the GVSP leader payload.
SetGevStatus [▶ 1870]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 1710]**	Pointer to the GVSP meta information.

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.2.2.2.5 ITcVnDisplayableImage

Interface for displayable images.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnDisplayableImage

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.6 ITcVnImage

Basic interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageBase [▶ 1873]
    ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1874]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1874]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1875]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1875]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1875]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1876]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1876]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1877]	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 [▶ 1872]	ITcVnImage	Initialize an image with an ITcVnImageBase interface.
GetRowPointer [▶ 1873]	ITcVnImage	Gets a pointer to a specific row of an image.
ReleaseRowPointer [▶ 1873]	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 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* [▶ 1873]	Image from which to obtain the data.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.2.7 ITcVnImageBase

Base interface for all image types.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnImageBase

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1874]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1874]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1875]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1875]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1875]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1876]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1876]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1877]	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 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* [▶ 1881]*	Returns a pointer to the image data interface.

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 1728]&	Returns a struct describing the pixel format.

Return value

HRESULT [▶ 141]

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 [▶ 1716]&	Returns a struct describing the image.

Return value

HRESULT [▶ 141]

6.2.2.3 Machine Learning

This group contains interfaces for machine learning.

6.2.2.3.1 ITcVnColorModel

Interface to train and classify an image color.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnMlModel [▶ 1879]
    ITcVnColorModel

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1879]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1879]	ITcVnMlModel	Gets the model type name as a string.
TrainImageColor [▶ 1878]	ITcVnColorModel	Train the color of the provided image.
ClassifyImageColor [▶ 1878]	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 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* [▶ 1871]	Source image.
nDifferentColors	ULONG	Number of different colors to differentiate.
ipMask	ITcVnImage* [▶ 1871]	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 [▶ 141]

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* [▶ 1871]	Source image.
pipDestImage	ITcVnImage* [▶ 1871]*	Returns the color similarity.
fVariance	float	Allowed color variance.
fLuminanceWeight	float	Weight the impact of the luminance.

Return value

HRESULT [▶ 141]

6.2.2.3.2 ITcVnMlModel

Interface for a machine learning model.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnMlModel

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1879]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1879]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.4 Miscellaneous

This group contains other interfaces.

6.2.2.4.1 ITcSerializableNotification

Interface required for sending serialized objects as ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcSerializableNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetComparisonValue [▶ 1880]	ITcSerializableNotification	Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).
SerializeRpcUnlocked [▶ 1881]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.4.2 ITcVnData

Interface for accessing data.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnData

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
CheckIfCopy [▶ 1882]	ITcVnData	Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).
CheckIfImage [▶ 1882]	ITcVnData	Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).
GetSize [▶ 1882]	ITcVnData	Gets the size of the data.
CheckDataPointer [▶ 1883]	ITcVnData	Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).
GetDataPointer [▶ 1883]	ITcVnData	Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).
ReleaseDataPointer [▶ 1883]	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 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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

6.2.2.4.3 ITcVnTimestamp

Offers an interface for timestamps.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnTimestamp

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
UpdateTimestamp [▶ 1884]	ITcVnTimestamp	Updates the timestamp to the current time
GetTimestamp [▶ 1884]	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 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 [▶ 141]

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 [▶ 141]

6.2.2.4.4 Interface ITcUnknown

ITcUnknown defines the reference count as well as the querying of a reference to a more specific interface.

Syntax

```
TCOM_DECL_INTERFACE("00000001-0000-0000-e000-000000000064", ITcUnknown)
```

Declared in: TcInterfaces.h

Required include: -

Methods

Name	Description
TcAddRef [▶ 1885]	Increments the reference counter.
TcQueryInterface [▶ 1885]	Query of the reference to an implemented interface via the IID.
TcRelease [▶ 1886]	Decrements the reference counter.

Every TcCOM interface is derived directly or indirectly from ITcUnknown. As a result, every TcCOM module class implements ITcUnknown because it is derived from ITComObject.

The standard implementation of ITcUnknown ensures that the object is deleted after the last reference is released. For this reason, an interface pointer must not be dereferenced after calling TcRelease().

6.2.2.4.4.1 Method ITcUnknown:TcAddRef

Increments the reference counter and returns the new value.

Syntax

```
ULONG TcAddRef( )
```

Return value

Resulting reference count value.

6.2.2.4.4.2 Method ITcUnknown:TcQueryInterface

Query of an interface pointer with regard to an interface that is given by interface ID (IID).

Syntax

```
HRESULT TcQueryInterface(RITCID iid, PPVOID pipItf )
```

Return value

If successful, S_OK ("0") or another positive value will be returned, cf. Return values. Extended messages refer in particular to the column HRESULT in ADS Return Codes [▶ 2879].

If the demanded interface is not available, the method returns ADSERR_DEVICE_NOINTERFACE.

Parameter

Name	Type	Description
iid	RITCID	Interface IID.
pipItf	PPVOID	Pointer to interface pointer. Is set when the requested interface type is available from the corresponding instance.

Description

Query of the reference to an implemented interface via the IID. It is recommended to use Smart Pointers to initialize and hold interface pointers.

Variant 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;
}
}
```

The interface ID associated with the Smart Pointer can be used as a parameter in TcQueryInterface. The "&" operator will return the pointer to the internal interface pointer member variable of the Smart Pointer. Variant 1 assumes that the interface pointer is initialized when TcQueryInterface indicates success. If the range remains, then the destructor of the Smart Pointer spObj releases the reference.

Variant 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;
}
```

If the interface pointer ip is assigned to the Smart Pointer spObj, then the TcQueryInterface method is called implicitly with IID_ITComObject on the instance to which ip refers. This results in a shorter code, but the original return code of TcQueryInterface is lost.

6.2.2.4.4.3 Method ITcUnknown:TcRelease

This method decrements the reference counter.

Syntax

```
ULONG TcRelease( )
```

Return value

Resulting reference count value.

Description

Decrements the reference counter and returns the new value.

If the reference counter is 0, the object deletes itself.

6.2.2.5 Neural Network

6.2.2.5.1 ITcVnNeuralNetwork

Interface for neural networks.

Inheritance Hierarchy

ITcUnknown [▶ 1885]
  ITcVnMlModel [▶ 1879]
    ITcVnNeuralNetwork

Methods

Name	Origin	Description
TcAddRef [▶ 1885]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1885]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1886]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1879]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1879]	ITcVnMlModel	Gets the model type name as a string.
Execute [▶ 1888]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from the last output layer. The method doesn't perform any pre-processing or transformation.
Execute_SingleLayerOutput [▶ 1888]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from a specified output layer. The method doesn't perform any pre-processing or transformation.
Execute_MultiLayerOutput [▶ 1889]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output from 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 [▶ 1890]	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 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 from the last 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* [▶ 1871]	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* [▶ 1871]*	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 [▶ 141]

6.2.2.5.1.2 Execute_SingleLayerOutput

Execute a neural network using the provided image as input and provide the output from a 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* [▶ 1871]	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* [▶ 1871]*	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 the last output layer is provided.

Return value

HRESULT [▶ 141]

6.2.2.5.1.3 Execute_MultiLayerOutput

Execute a neural network using the provided image as input and provide the output from 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* [▶ 1871]	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* [▶ 1836]*	Container of images containing the output from the specified layers
ipLayerNames	ITcVnContainer* [▶ 1836]	Container of layer names whose outputs are required (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.

Return value

HRESULT [▶ 141]

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* [▶ 1836]*	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	bool	If true, only the unconnected output layer names are returned

Return value

HRESULT [▶ 141]

6.2.3 Functions

The image processing algorithms are encapsulated in functions. More complex algorithms have been implemented as function blocks and can be found at ImageProcessing [▶ 2699].

Structure of a function call

Functions are structured as follows:

```
hr := ProcessImage (hr, <...>, <...>);

Status variable of type HRESULT
Parameters and return values as references
Optional parameters with default values
```

hr	Status variable [▶ 141] of the type HRESULT. The execution status of every function can be followed with this.
Parameter <...>	Function parameters essentially fulfill the following purpose: • Input parameters (e.g. images or containers) that are to be processed. • Parameters for setting the algorithm • Parameters as reference for returning the results
Optional parameter <...>	These parameters are initialized with default values, so the specification is optional.

Grouping of functions

All functions of the TwinCAT Vision library are sorted thematically into the following groups:

Images

- Basic Image Operations [▶ 2164]
- Algebraic Image Operations [▶ 1950]
- Image Segmentation [▶ 2504]
- Image Color and Contrast Processing [▶ 2465]
- Image Analysis [▶ 2434]
- Image Filtering [▶ 2486]

Container

- Basic Container Operations [▶ 1985]
- Algebraic Container Operations [▶ 1917]
- Container Statistics [▶ 2212]

Contours

- Contour Analysis [▶ 2284]

Measurement

- Measurement [▶ 2587]
- Geometric and coordinate transformations [▶ 2360]

Code quality

- Code Quality [▶ 2196]

Code reading

- Code Reading [▶ 2203]

Optical character recognition (OCR)

- Optical Character Recognition [▶ 2636]

Drawing and text

- Drawing [▶ 2315]

Machine Learning

- MachineLearning [▶ 2546]
- Neural Network [▶ 2642]

Advanced Functions

- Fourier Analysis [▶ 2348]
- Keypoint Features [▶ 2511]

Miscellaneous

- Control Functions [▶ 2312]
- Miscellaneous [▶ 2620]

Expert functions

Expert versions exist for many functions. These contain additional parameters and are identified in the PLC library by the name suffix "`Exp`". In C++, these additional parameters are assigned default values and can be omitted when called if not required.

Runtime behavior

The execution duration of some image processing algorithms depends on their parameterization and also the image content. It is therefore not deterministic. In order to guarantee real-time behavior despite that, watchdogs [▶ 146] can be applied to these functions or function groups that abort the functions after a specified time.

⚠ WARNING

TwinCAT crash when functions are used incorrectly

If some functions are used incorrectly, this can lead to errors (and to resulting TwinCAT crashes). These are not intercepted within the function, as corresponding checks would considerably lengthen the execution time. Therefore, you must take care to use the functions properly! For example, we recommend carrying out a validation of the input parameters.

6.2.3.1 License Overview

The following list shows the assignment of TwinCAT Vision functions to available licenses (see Licensing model [▶ 12]). It shows which license is required for the application of certain functions.

TF7100

The basic license TC3 Vision Base (TF7100) is always required.

TC3 Vision Base

AdaptiveThreshold [▶ 2505]

AddContainers [▶ 1940]

AddImages [▶ 1951]

AddImagesWeighted [▶ 1952]

AddScalarToImage [▶ 1953]

AddToContainerElements (LONG) [▶ 1917]

AddToContainerElements (SHORT) [▶ 1918]

AddToContainerElements (double) [▶ 1918]

AddToContainerElements (float) [▶ 1919]

AddToContainerElements (char) [▶ 1920]

AddToContainerElements (TcVnKeyPoint) [▶ 1920]

AddToContainerElements (TcVnPoint2_DINT) [▶ 1921]

AddToContainerElements (TcVnPoint2_LREAL) [▶ 1922]

AddToContainerElements (TcVnPoint2_REAL) [▶ 1922]

AddToContainerElements (TcVnPoint3_LREAL) [▶ 1923]

AddToContainerElements (TcVnPoint3_REAL) [▶ 1924]

AddToContainerElements (TcVnRectangle_DINT) [▶ 1924]

AddToContainerElements (TcVnVector2_DINT) [▶ 1925]

AddToContainerElements (TcVnVector2_INT) [▶ 1926]

AddToContainerElements (TcVnVector2_REAL) [▶ 1926]

AddToContainerElements (TcVnVector2_SINT) [▶ 1927]

AddToContainerElements (TcVnVector2_UINT) [▶ 1928]

AddToContainerElements (TcVnVector2_USINT) [▶ 1928]

AddToContainerElements (TcVnVector3_INT) [▶ 1929]

AddToContainerElements (TcVnVector3_REAL) [▶ 1930]

AddToContainerElements (TcVnVector3_SINT) [▶ 1930]

AddToContainerElements (TcVnVector3_UINT) [▶ 1931]

AddToContainerElements (TcVnVector3_USINT) [▶ 1932]

AddToContainerElements (TcVnVector4_DINT) [▶ 1932]

AddToContainerElements (TcVnVector4_INT) [▶ 1933]

AddToContainerElements (TcVnVector4_LREAL) [▶ 1934]

AddToContainerElements (TcVnVector4_REAL) [▶ 1939]

AddToContainerElements (TcVnVector4_SINT) [▶ 1934]

AddToContainerElements (TcVnVector4_UINT) [▶ 1935]

AddToContainerElements (TcVnVector4_USINT) [▶ 1936]

AddToContainerElements (ULONG) [▶ 1936]

AddToContainerElements (USHORT) [▶ 1937]

AddToContainerElements (ULONGLONG) [▶ 1938]

AddToContainerElements (unsigned char) [▶ 1938]

AddVectorToImage [▶ 1953]

AdjustActiveContour [▶ 2435]

AdvanceIterator [▶ 2144]

AlignRotatedImageRegion [▶ 2394]

AppendToContainer (LONG) [▶ 1987]

AppendToContainer (SHORT) [▶ 1987]

AppendToContainer (ITcVnContainer*) [▶ 1988]

AppendToContainer (ITcVnForwardIterator*) [▶ 1989]

AppendToContainer (ITcVnImage*) [▶ 1989]

AppendToContainer (double) [▶ 1990]

AppendToContainer (float) [▶ 1991]

AppendToContainer (char) [▶ 1991]

AppendToContainer (TcVnDMatch) [▶ 1992]

AppendToContainer (TcVnKeyPoint) [▶ 1993]

AppendToContainer (TcVnPoint2_DINT) [▶ 1993]

AppendToContainer (TcVnPoint2_LREAL) [▶ 1994]

AppendToContainer (TcVnPoint2_REAL) [▶ 1995]

AppendToContainer (TcVnPoint3_LREAL) [▶ 1995]

AppendToContainer (TcVnPoint3_REAL) [▶ 1996]

AppendToContainer (TcVnRectangle_DINT) [▶ 1997]

AppendToContainer (TcVnVector2_DINT) [▶ 1997]

AppendToContainer (TcVnVector2_INT) [▶ 1998]

AppendToContainer (TcVnVector2_REAL) [▶ 1999]

AppendToContainer (TcVnVector2_SINT) [▶ 1999]

AppendToContainer (TcVnVector2_UINT) [▶ 2000]

AppendToContainer (TcVnVector2_USINT) [▶ 2001]

AppendToContainer (TcVnVector3_INT) [▶ 2001]

AppendToContainer (TcVnVector3_REAL) [▶ 2002]

AppendToContainer (TcVnVector3_SINT) [▶ 2003]

AppendToContainer (TcVnVector3_UINT) [▶ 2003]

AppendToContainer (TcVnVector3_USINT) [▶ 2004]

AppendToContainer (TcVnVector4_DINT) [▶ 2005]

AppendToContainer (TcVnVector4_INT) [▶ 2005]

AppendToContainer (TcVnVector4_LREAL) [▶ 2006]

AppendToContainer (TcVnVector4_REAL) [▶ 2011]

AppendToContainer (TcVnVector4_SINT) [▶ 2007]

AppendToContainer (TcVnVector4_UINT) [▶ 2007]

AppendToContainer (TcVnVector4_USINT) [▶ 2008]

AppendToContainer (ULONG) [▶ 2009]

AppendToContainer (USHORT) [▶ 2009]

AppendToContainer (ULONGLONG) [▶ 2010]

AppendToContainer (unsigned char) [▶ 2011]

ApplyColorMap [▶ 2465]

ApplyLut [▶ 2466]

ApplyRotationToAffineTransformation [▶ 2395]

ApplyScalingToAffineTransformation [▶ 2396]

ApplyTranslationToAffineTransformation [▶ 2397]

ApplyYAxisInversionToAffineTransformation [▶ 2398]

ApproximatePolygon [▶ 2285]

BilateralFilter [▶ 2487]

BitwiseAndContainers [▶ 1940]

BitwiseAndImages [▶ 1954]

BitwiseAndImages (Exp) [▶ 1955]

BitwiseAndScalarWithImage [▶ 1956]

BitwiseAndVectorWithImage [▶ 1956]

BitwiseNotContainer [▶ 1941]

BitwiseNotImage [▶ 1957]

BitwiseNotImage (Exp) [▶ 1958]

BitwiseOrContainers [▶ 1942]

BitwiseOrImages [▶ 1958]

BitwiseOrImages (Exp) [▶ 1959]

BitwiseOrScalarWithImage [▶ 1960]

BitwiseOrVectorWithImage [▶ 1961]

BitwiseXorContainers [▶ 1942]

BitwiseXorImages [▶ 1962]

BitwiseXorImages (Exp) [▶ 1962]

BitwiseXorScalarWithImage [▶ 1963]

BitwiseXorVectorWithImage [▶ 1964]

BlendImages [▶ 1965]

BoxFilter [▶ 2488]

BrightBorderObjects [▶ 2489]

CalibrateCameraPlanar [▶ 2370]

CannyEdgeDetection [▶ 2449]

CheckColorRange [▶ 2507]

CheckFunctionInitialization [▶ 2620]

CheckIfEmpty [▶ 2145]

CheckIfIteratorIsAtEnd [▶ 2146]

CheckIfPointIsInsideContour [▶ 2286]

Clahe [▶ 2467]

ClipLineToBoundary [▶ 2621]

ClipLineToBoundary (ITcVnImage*) [▶ 2622]

ClipLineToBoundary (TcVnRectangle_DINT) [▶ 2623]

CombineImageChannels [▶ 2166]

ConnectedComponents [▶ 2450]

ConnectedComponentsWithStats [▶ 2451]

ContainerAverage [▶ 2276]

ContainerAverageElementwise (2) [▶ 2277]

ContainerAverageElementwise (3) [▶ 2278]

ContainerAverageElementwise (4) [▶ 2278]

ContainerAverageVariance [▶ 2279]

ContainerAverageVarianceElementwise (2) [▶ 2280]

ContainerAverageVarianceElementwise (3) [▶ 2280]

ContainerAverageVarianceElementwise (4) [▶ 2281]

ContourArea [▶ 2287]

ContourCenterOfMass [▶ 2288]

ContourCircularity [▶ 2289]

ContourConvexity [▶ 2290]

ContourEccentricity [▶ 2291]

ContourElongation [▶ 2291]

ContourExtremePoint [▶ 2292]

ContourInertiaRatio [▶ 2293]

ContourMoments [▶ 2294]

ContourOrientation [▶ 2295]

ContourPerimeter [▶ 2296]

ContourRoundness [▶ 2297]

ConvertCartesianToPolarAngleImage [▶ 2399]

ConvertCartesianToPolarAngles [▶ 2400]

ConvertCartesianToPolarImages [▶ 2401]

ConvertCartesianToPolarMagnitudeImage [▶ 2402]

ConvertCartesianToPolarMagnitudes [▶ 2403]

ConvertCartesianToPolarPoints [▶ 2404]

ConvertColorSpace [▶ 2468]

ConvertContainerType [▶ 2146]

ConvertDataLayout [▶ 2642]

ConvertElementType [▶ 2166]

ConvertITcUnknownToITcVnBitmapExport [▶ 2624]

ConvertITcUnknownToITcVnContainer [▶ 2625]

ConvertITcUnknownToITcVnImage [▶ 2625]

ConvertITcUnknownToITcVnMlModel [▶ 2626]

ConvertPolarToCartesianImages [▶ 2405]

ConvertPolarToCartesianPoints [▶ 2406]

ConvexHullPoints [▶ 2298]

ConvexityDefects [▶ 2299]

CopyContainer [▶ 2147]

CopyContainerElementsConditional (ITcVnContainer*) [▶ 2148]

CopyContainerElementsConditional (ITcVnForwardIterator*) [▶ 2148]

CopyImage [▶ 2167]

CopyImageRegion [▶ 2168]

CopyImageRegionToRegion [▶ 2169]

CopyIntoDisplayableImage [▶ 2170]

CountNonZeroPixels [▶ 2453]

CreateAssociatedImage [▶ 2171]

CreateBandpassButterworthFilter [▶ 2348]

CreateBandpassGaussianFilter [▶ 2349]

CreateBandrejectButterworthFilter [▶ 2350]

CreateBandrejectGaussianFilter [▶ 2352]

CreateContainer [▶ 2149]

CreateContainerFromArray [▶ 2150]

CreateEmptyImage [▶ 2171]

CreateHighpassButterworthFilter [▶ 2353]

CreateHighpassGaussianFilter [▶ 2354]

CreateImage [▶ 2172]

CreateImageAndSetPixels [▶ 2173]

CreateImageFromArray [▶ 2174]

CreateLowpassButterworthFilter [▶ 2355]

CreateLowpassGaussianFilter [▶ 2356]

CreateNDimensionalImage [▶ 2193]

CreateNDimensionalImageAndSetPixels [▶ 2194]

CreateNDimensionalImageFromArray [▶ 2195]

CreateStructuringElement [▶ 2489]

CustomElementWiseContainerOperation (ITcVnContainer*) [▶ 2300]

CustomElementWiseContainerOperation (ITcVnForwardIterator*) [▶ 2301]

CustomFilter [▶ 2490]

DarkBorderObjects [▶ 2491]

DecomposeAffineTransformation [▶ 2407]

DecomposeHomography [▶ 2408]

DeinitializeFunction [▶ 2627]

DetectBlobs [▶ 2436]

Dft [▶ 2357]

DistanceTransformation [▶ 2453]

DivideContainers [▶ 1943]

DivideImageByScalar [▶ 1965]

DivideImageByVector [▶ 1966]

DivideImages [▶ 1967]

DivideScalarByImage [▶ 1967]

DivideVectorByImage [▶ 1968]

DoubleThreshold [▶ 2507]

DrawArrow [▶ 2316]

DrawArrow (TcVnVector4_DINT) [▶ 2317]

DrawCircle [▶ 2318]

DrawCircles [▶ 2319]

DrawCircularArc [▶ 2320]

DrawComponents [▶ 2321]

DrawContours [▶ 2322]

DrawEllipse [▶ 2323]

DrawLine [▶ 2325]

DrawLine (TcVnVector4_DINT) [▶ 2326]

DrawLine (TcVnVector4_LREAL) [▶ 2327]

DrawLines [▶ 2328]

DrawOrientation [▶ 2332]

DrawOrientation (Exp) [▶ 2332]

DrawPoint [▶ 2333]

DrawPoints [▶ 2334]

DrawPolygon [▶ 2335]

DrawRectangle [▶ 2336]

DrawRectangle (TcVnRectangle_DINT) [▶ 2337]

DrawRectangle (TcVnRectangle_UDINT) [▶ 2338]

DrawRotatedRectangle [▶ 2339]

ElementwiseExp [▶ 1969]

ElementwiseLog [▶ 1970]

EnclosingCircle [▶ 2302]

EnclosingRectangle [▶ 2303]

EnclosingTriangle [▶ 2303]

EraseFromContainer [▶ 2151]

ExportContainer [▶ 2151]

ExportContainer_String [▶ 2152]

ExportContainerSize [▶ 2153]

ExportImage [▶ 2175]

ExportImageAsBmp [▶ 2176]

ExportImageAsBmpSize [▶ 2177]

ExportImageSize [▶ 2178]

ExportSubContainer [▶ 2154]

ExportSubContainer_String [▶ 2155]

ExportSubContainerSize [▶ 2156]

ExtractContainerRange [▶ 2156]

FillCircle [▶ 2340]

FillContainer (LONG) [▶ 2012]

FillContainer (SHORT) [▶ 2013]

FillContainer (ITcVnImage*) [▶ 2013]

FillContainer (double) [▶ 2014]

FillContainer (float) [▶ 2015]

FillContainer (char) [▶ 2015]

FillContainer (TcVnDMatch) [▶ 2016]

FillContainer (TcVnKeyPoint) [▶ 2017]

FillContainer (TcVnPoint2_DINT) [▶ 2017]

FillContainer (TcVnPoint2_LREAL) [▶ 2018]

FillContainer (TcVnPoint2_REAL) [▶ 2019]

FillContainer (TcVnPoint3_LREAL) [▶ 2019]

FillContainer (TcVnPoint3_REAL) [▶ 2020]

FillContainer (TcVnRectangle_DINT) [▶ 2021]

FillContainer (TcVnVector2_DINT) [▶ 2021]

FillContainer (TcVnVector2_INT) [▶ 2022]

FillContainer (TcVnVector2_REAL) [▶ 2023]

FillContainer (TcVnVector2_SINT) [▶ 2023]

FillContainer (TcVnVector2_UINT) [▶ 2024]

FillContainer (TcVnVector2_USINT) [▶ 2025]

FillContainer (TcVnVector3_INT) [▶ 2025]

FillContainer (TcVnVector3_REAL) [▶ 2026]

FillContainer (TcVnVector3_SINT) [▶ 2027]

FillContainer (TcVnVector3_UINT) [▶ 2027]

FillContainer (TcVnVector3_USINT) [▶ 2028]

FillContainer (TcVnVector4_DINT) [▶ 2029]

FillContainer (TcVnVector4_INT) [▶ 2029]

FillContainer (TcVnVector4_LREAL) [▶ 2030]

FillContainer (TcVnVector4_REAL) [▶ 2035]

FillContainer (TcVnVector4_SINT) [▶ 2031]

FillContainer (TcVnVector4_UINT) [▶ 2031]

FillContainer (TcVnVector4_USINT) [▶ 2032]

FillContainer (ULONG) [▶ 2033]

FillContainer (USHORT) [▶ 2033]

FillContainer (ULONGLONG) [▶ 2034]

FillContainer (unsigned char) [▶ 2035]

FillContainer (LONG) [▶ 2036]

FillContainer (SHORT) [▶ 2037]

FillContainer (ITcVnImage*) [▶ 2037]

FillContainer (double) [▶ 2038]

FillContainer (float) [▶ 2039]

FillContainer (char) [▶ 2039]

FillContainer (TcVnDMatch) [▶ 2040]

FillContainer (TcVnKeyPoint) [▶ 2041]

FillContainer (TcVnPoint2_DINT) [▶ 2042]

FillContainer (TcVnPoint2_LREAL) [▶ 2042]

FillContainer (TcVnPoint2_REAL) [▶ 2043]

FillContainer (TcVnPoint3_LREAL) [▶ 2044]

FillContainer (TcVnPoint3_REAL) [▶ 2044]

FillContainer (TcVnRectangle_DINT) [▶ 2045]

FillContainer (TcVnVector2_DINT) [▶ 2046]

FillContainer (TcVnVector2_INT) [▶ 2047]

FillContainer (TcVnVector2_REAL) [▶ 2047]

FillContainer (TcVnVector2_SINT) [▶ 2048]

FillContainer (TcVnVector2_UINT) [▶ 2049]

FillContainer (TcVnVector2_USINT) [▶ 2049]

FillContainer (TcVnVector3_INT) [▶ 2050]

FillContainer (TcVnVector3_REAL) [▶ 2051]

FillContainer (TcVnVector3_SINT) [▶ 2052]

FillContainer (TcVnVector3_UINT) [▶ 2052]

FillContainer (TcVnVector3_USINT) [▶ 2053]

FillContainer (TcVnVector4_DINT) [▶ 2054]

FillContainer (TcVnVector4_INT) [▶ 2054]

FillContainer (TcVnVector4_LREAL) [▶ 2055]

FillContainer (TcVnVector4_REAL) [▶ 2061]

FillContainer (TcVnVector4_SINT) [▶ 2056]

FillContainer (TcVnVector4_UINT) [▶ 2057]

FillContainer (TcVnVector4_USINT) [▶ 2057]

FillContainer (ULONG) [▶ 2058]

FillContainer (USHORT) [▶ 2059]

FillContainer (ULONGLONG) [▶ 2059]

FillContainer (unsigned char) [▶ 2060]

FillContours [▶ 2340]

FillEllipse [▶ 2341]

FillHoles [▶ 2492]

FillPolygon [▶ 2342]

FillRectangle [▶ 2343]

FillRotatedRectangle [▶ 2344]

FindContourHierarchy (Exp) [▶ 2437]

FindContours [▶ 2438]

FitEllipse [▶ 2304]

FitLine [▶ 2305]

FlipImage [▶ 2412]

FourierDescriptors [▶ 2307]

FuseImages [▶ 2179]

FuseImagesArray [▶ 2180]

GaussianFilter [▶ 2493]

GenerateAffineTransformationUnitMatrix2D [▶ 2413]

GenerateColorMap [▶ 2469]

GenerateCustomColorMap [▶ 2470]

GetAffineTransformation [▶ 2413]

GetAffineTransformation2D [▶ 2414]

GetAt (LONG) [▶ 2062]

GetAt (SHORT) [▶ 2062]

GetAt (ITcVnContainer*) [▶ 2063]

GetAt (ITcVnImage*) [▶ 2064]

GetAt (double) [▶ 2065]

GetAt (float) [▶ 2065]

GetAt (char) [▶ 2066]

GetAt (TcVnDMatch) [▶ 2067]

GetAt (TcVnKeyPoint) [▶ 2067]

GetAt (TcVnPoint2_DINT) [▶ 2068]

GetAt (TcVnPoint2_LREAL) [▶ 2069]

GetAt (TcVnPoint2_REAL) [▶ 2069]

GetAt (TcVnPoint3_LREAL) [▶ 2070]

GetAt (TcVnPoint3_REAL) [▶ 2071]

GetAt (TcVnRectangle_DINT) [▶ 2072]

GetAt (TcVnVector2_DINT) [▶ 2072]

GetAt (TcVnVector2_INT) [▶ 2073]

GetAt (TcVnVector2_REAL) [▶ 2074]

GetAt (TcVnVector2_SINT) [▶ 2074]

GetAt (TcVnVector2_UINT) [▶ 2075]

GetAt (TcVnVector2_USINT) [▶ 2076]

GetAt (TcVnVector3_INT) [▶ 2077]

GetAt (TcVnVector3_REAL) [▶ 2077]

GetAt (TcVnVector3_SINT) [▶ 2078]

GetAt (TcVnVector3_UINT) [▶ 2079]

GetAt (TcVnVector3_USINT) [▶ 2079]

GetAt (TcVnVector4_DINT) [▶ 2080]

GetAt (TcVnVector4_INT) [▶ 2081]

GetAt (TcVnVector4_LREAL) [▶ 2082]

GetAt (TcVnVector4_REAL) [▶ 2087]

GetAt (TcVnVector4_SINT) [▶ 2082]

GetAt (TcVnVector4_UINT) [▶ 2083]

GetAt (TcVnVector4_USINT) [▶ 2084]

GetAt (ULONG) [▶ 2084]

GetAt (USHORT) [▶ 2085]

GetAt (ULONGLONG) [▶ 2086]

GetAt (unsigned char) [▶ 2087]

GetConnectedComponent [▶ 2454]

GetContainer [▶ 2157]

GetContainer (Exp) [▶ 2158]

GetForwardIterator [▶ 2158]

GetImageChannel [▶ 2180]

GetImageHeight [▶ 2181]

GetImageInfo [▶ 2182]

GetImageWidth [▶ 2182]

GetLayoutInfo [▶ 2195]

GetNumberOfElements [▶ 2159]

GetPerspectiveTransformation [▶ 2416]

GetPixel [▶ 2183]

GetPixelFormat [▶ 2184]

GetRandomAccessIterator [▶ 2160]

GetRoi [▶ 2184]

GetTimestamp [▶ 2627]

Histogram [▶ 2471]

HistogramEqualization [▶ 2472]

Homography [▶ 2416]

HoughCircles [▶ 2440]

HoughLines [▶ 2442]

HoughLinesP [▶ 2443]

HuMomentInvariants [▶ 2628]

ImageAbsoluteAverage [▶ 2463]

ImageAverage [▶ 2455]

ImageAverageStdDev [▶ 2456]

ImageCenterOfMass [▶ 2457]

ImageMedian [▶ 2458]

ImageMoments [▶ 2459]

IncrementIterator [▶ 2160]

InitMatrixStruct [▶ 2629]

InsertIntoContainer (LONG) [▶ 2088]

InsertIntoContainer (SHORT) [▶ 2089]

InsertIntoContainer (ITcVnContainer*) [▶ 2090]

InsertIntoContainer (ITcVnForwardIterator*) [▶ 2090]

InsertIntoContainer (ITcVnImage*) [▶ 2091]

InsertIntoContainer (double) [▶ 2092]

InsertIntoContainer (float) [▶ 2093]

InsertIntoContainer (char) [▶ 2093]

InsertIntoContainer (TcVnDMatch) [▶ 2094]

InsertIntoContainer (TcVnKeyPoint) [▶ 2095]

InsertIntoContainer (TcVnPoint2_DINT) [▶ 2096]

InsertIntoContainer (TcVnPoint2_LREAL) [▶ 2096]

InsertIntoContainer (TcVnPoint2_REAL) [▶ 2097]

InsertIntoContainer (TcVnPoint3_LREAL) [▶ 2098]

InsertIntoContainer (TcVnPoint3_REAL) [▶ 2099]

InsertIntoContainer (TcVnRectangle_DINT) [▶ 2099]

InsertIntoContainer (TcVnVector2_DINT) [▶ 2100]

InsertIntoContainer (TcVnVector2_INT) [▶ 2101]

InsertIntoContainer (TcVnVector2_REAL) [▶ 2102]

InsertIntoContainer (TcVnVector2_SINT) [▶ 2102]

InsertIntoContainer (TcVnVector2_UINT) [▶ 2103]

InsertIntoContainer (TcVnVector2_USINT) [▶ 2104]

InsertIntoContainer (TcVnVector3_INT) [▶ 2105]

InsertIntoContainer (TcVnVector3_REAL) [▶ 2105]

InsertIntoContainer (TcVnVector3_SINT) [▶ 2106]

InsertIntoContainer (TcVnVector3_UINT) [▶ 2107]

InsertIntoContainer (TcVnVector3_USINT) [▶ 2108]

InsertIntoContainer (TcVnVector4_DINT) [▶ 2108]

InsertIntoContainer (TcVnVector4_INT) [▶ 2109]

InsertIntoContainer (TcVnVector4_LREAL) [▶ 2110]

InsertIntoContainer (TcVnVector4_REAL) [▶ 2116]

InsertIntoContainer (TcVnVector4_SINT) [▶ 2111]

InsertIntoContainer (TcVnVector4_UINT) [▶ 2111]

InsertIntoContainer (TcVnVector4_USINT) [▶ 2112]

InsertIntoContainer (ULONG) [▶ 2113]

InsertIntoContainer (USHORT) [▶ 2114]

InsertIntoContainer (ULONGLONG) [▶ 2114]

InsertIntoContainer (unsigned char) [▶ 2115]

InverseDft [▶ 2358]

InvertAffineTransformation [▶ 2417]

InvertImageColor [▶ 2473]

InvertMatrix3x3 [▶ 2629]

IteratorDistance [▶ 2161]

LaplacianFilter [▶ 2494]

LineIntersectionPoint [▶ 2630]

LineIntersectionPointAndAngle [▶ 2631]

LocalMaxima [▶ 2495]

LocalMinima [▶ 2496]

MaxContainer [▶ 1944]

MaxElement [▶ 2282]

MaxElementElementwise (LONG) [▶ 2212]

MaxElementElementwise (SHORT) [▶ 2213]

MaxElementElementwise (double) [▶ 2214]

MaxElementElementwise (float) [▶ 2214]

MaxElementElementwise (char) [▶ 2215]

MaxElementElementwise (TcVnPoint2_DINT) [▶ 2216]

MaxElementElementwise (TcVnPoint2_LREAL) [▶ 2216]

MaxElementElementwise (TcVnPoint2_REAL) [▶ 2217]

MaxElementElementwise (TcVnPoint3_LREAL) [▶ 2218]

MaxElementElementwise (TcVnPoint3_REAL) [▶ 2218]

MaxElementElementwise (TcVnVector2_DINT) [▶ 2219]

MaxElementElementwise (TcVnVector2_INT) [▶ 2220]

MaxElementElementwise (TcVnVector2_REAL) [▶ 2220]

MaxElementElementwise (TcVnVector2_SINT) [▶ 2221]

MaxElementElementwise (TcVnVector2_UINT) [▶ 2222]

MaxElementElementwise (TcVnVector2_USINT) [▶ 2222]

MaxElementElementwise (TcVnVector3_INT) [▶ 2223]

MaxElementElementwise (TcVnVector3_REAL) [▶ 2224]

MaxElementElementwise (TcVnVector3_SINT) [▶ 2224]

MaxElementElementwise (TcVnVector3_UINT) [▶ 2225]

MaxElementElementwise (TcVnVector3_USINT) [▶ 2226]

MaxElementElementwise (TcVnVector4_DINT) [▶ 2226]

MaxElementElementwise (TcVnVector4_INT) [▶ 2227]

MaxElementElementwise (TcVnVector4_LREAL) [▶ 2228]

MaxElementElementwise (TcVnVector4_REAL) [▶ 2233]

MaxElementElementwise (TcVnVector4_SINT) [▶ 2228]

MaxElementElementwise (TcVnVector4_UINT) [▶ 2229]

MaxElementElementwise (TcVnVector4_USINT) [▶ 2230]

MaxElementElementwise (ULONG) [▶ 2230]

MaxElementElementwise (USHORT) [▶ 2231]

MaxElementElementwise (ULONGLONG) [▶ 2232]

MaxElementElementwise (unsigned char) [▶ 2232]

MaxImage [▶ 1970]

MaxImage (Exp) [▶ 1971]

MaxImageWithScalar [▶ 1972]

MaxImageWithScalar (Exp) [▶ 1972]

MaxImageWithVector [▶ 1973]

MaxImageWithVector (Exp) [▶ 1974]

MaxPixelValue [▶ 2460]

MedianElement [▶ 2283]

MedianElementElementwise (LONG) [▶ 2234]

MedianElementElementwise (SHORT) [▶ 2234]

MedianElementElementwise (double) [▶ 2235]

MedianElementElementwise (float) [▶ 2236]

MedianElementElementwise (char) [▶ 2236]

MedianElementElementwise (TcVnPoint2_DINT) [▶ 2237]

MedianElementElementwise (TcVnPoint2_LREAL) [▶ 2238]

MedianElementElementwise (TcVnPoint2_REAL) [▶ 2238]

MedianElementElementwise (TcVnPoint3_LREAL) [▶ 2239]

MedianElementElementwise (TcVnPoint3_REAL) [▶ 2240]

MedianElementElementwise (TcVnVector2_DINT) [▶ 2240]

MedianElementElementwise (TcVnVector2_INT) [▶ 2241]

MedianElementElementwise (TcVnVector2_REAL) [▶ 2242]

MedianElementElementwise (TcVnVector2_SINT) [▶ 2242]

MedianElementElementwise (TcVnVector2_UINT) [▶ 2243]

MedianElementElementwise (TcVnVector2_USINT) [▶ 2244]

MedianElementElementwise (TcVnVector3_INT) [▶ 2244]

MedianElementElementwise (TcVnVector3_REAL) [▶ 2245]

MedianElementElementwise (TcVnVector3_SINT) [▶ 2246]

MedianElementElementwise (TcVnVector3_UINT) [▶ 2246]

MedianElementElementwise (TcVnVector3_USINT) [▶ 2247]

MedianElementElementwise (TcVnVector4_DINT) [▶ 2248]

MedianElementElementwise (TcVnVector4_INT) [▶ 2248]

MedianElementElementwise (TcVnVector4_LREAL) [▶ 2249]

MedianElementElementwise (TcVnVector4_REAL) [▶ 2254]

MedianElementElementwise (TcVnVector4_SINT) [▶ 2250]

MedianElementElementwise (TcVnVector4_UINT) [▶ 2250]

MedianElementElementwise (TcVnVector4_USINT) [▶ 2251]

MedianElementElementwise (ULONG) [▶ 2252]

MedianElementElementwise (USHORT) [▶ 2252]

MedianElementElementwise (ULONGLONG) [▶ 2253]

MedianElementElementwise (unsigned char) [▶ 2254]

MedianFilter [▶ 2497]

MinContainer [▶ 1944]

MinElement [▶ 2284]

MinElementElementwise (LONG) [▶ 2255]

MinElementElementwise (SHORT) [▶ 2256]

MinElementElementwise (double) [▶ 2256]

MinElementElementwise (float) [▶ 2257]

MinElementElementwise (char) [▶ 2258]

MinElementElementwise (TcVnPoint2_DINT) [▶ 2258]

MinElementElementwise (TcVnPoint2_LREAL) [▶ 2259]

MinElementElementwise (TcVnPoint2_REAL) [▶ 2260]

MinElementElementwise (TcVnPoint3_LREAL) [▶ 2260]

MinElementElementwise (TcVnPoint3_REAL) [▶ 2261]

MinElementElementwise (TcVnVector2_DINT) [▶ 2262]

MinElementElementwise (TcVnVector2_INT) [▶ 2262]

MinElementElementwise (TcVnVector2_REAL) [▶ 2263]

MinElementElementwise (TcVnVector2_SINT) [▶ 2264]

MinElementElementwise (TcVnVector2_UINT) [▶ 2264]

MinElementElementwise (TcVnVector2_USINT) [▶ 2265]

MinElementElementwise (TcVnVector3_INT) [▶ 2266]

MinElementElementwise (TcVnVector3_REAL) [▶ 2266]

MinElementElementwise (TcVnVector3_SINT) [▶ 2267]

MinElementElementwise (TcVnVector3_UINT) [▶ 2268]

MinElementElementwise (TcVnVector3_USINT) [▶ 2268]

MinElementElementwise (TcVnVector4_DINT) [▶ 2269]

MinElementElementwise (TcVnVector4_INT) [▶ 2270]

MinElementElementwise (TcVnVector4_LREAL) [▶ 2270]

MinElementElementwise (TcVnVector4_REAL) [▶ 2276]

MinElementElementwise (TcVnVector4_SINT) [▶ 2271]

MinElementElementwise (TcVnVector4_UINT) [▶ 2272]

MinElementElementwise (TcVnVector4_USINT) [▶ 2272]

MinElementElementwise (ULONG) [▶ 2273]

MinElementElementwise (USHORT) [▶ 2274]

MinElementElementwise (ULONGLONG) [▶ 2274]

MinElementElementwise (unsigned char) [▶ 2275]

MinImage [▶ 1975]

MinImage (Exp) [▶ 1976]

MinImageWithScalar [▶ 1977]

MinImageWithScalar (Exp) [▶ 1977]

MinImageWithVector [▶ 1978]

MinImageWithVector (Exp) [▶ 1979]

MinPixelValue [▶ 2461]

MixImageChannels [▶ 2185]

MorphologicalOperator [▶ 2497]

MultiplyContainers [▶ 1945]

MultiplyImages [▶ 1980]

MultiplyImageWithScalar [▶ 1980]

MultiplyImageWithVector [▶ 1981]

MultiplyMatrices [▶ 2632]

MultiplyWithContainerElements (1) [▶ 1946]

MultiplyWithContainerElements (2) [▶ 1946]

MultiplyWithContainerElements (3) [▶ 1947]

MultiplyWithContainerElements (4) [▶ 1948]

NegateContainer [▶ 1948]

NonMaxSuppression [▶ 2647]

NormalizeImage [▶ 2473]

NormalizeImageForDisplay [▶ 2475]

OptimalDftSize [▶ 2359]

PadImageBorder [▶ 2359]

PerspectiveTransformation [▶ 2418]

PlotIntensityProfile [▶ 2344]

PutLabel [▶ 2346]

PutText [▶ 2347]

PyramidDown [▶ 2419]

PyramidUp [▶ 2420]

Reduce [▶ 2649]

ReduceArg [▶ 2650]

ReferenceColorSimilarity (ITcVnColorModel*) [▶ 2475]

ReferenceColorSimilarity (ITcVnMlModel*) [▶ 2476]

ReferenceColorSimilarity (TcVnVector3_LREAL) [▶ 2477]

RegionOrientation [▶ 2462]

RemapImageToLogPolarSpace [▶ 2420]

RemapImageToPolarSpace [▶ 2422]

RemoveLocalMaxima [▶ 2498]

RemoveLocalMinima [▶ 2499]

ReserveContainerMemory [▶ 2162]

ResetRoi [▶ 2186]

ReshapeImage [▶ 2650]

ResizeImage [▶ 2423]

ReverseContainer [▶ 2162]

RotatedRectangleCorners [▶ 2633]

RotatedRectangleIntersection [▶ 2634]

RotateImage [▶ 2424]

RotateImage (Exp) [▶ 2424]

SauvolaThreshold [▶ 2510]

ScharrFilter [▶ 2500]

SeparableCustomFilter [▶ 2501]

SetAt (LONG) [▶ 2117]

SetAt (SHORT) [▶ 2118]

SetAt (ITcVnContainer*) [▶ 2118]

SetAt (ITcVnImage*) [▶ 2119]

SetAt (double) [▶ 2120]

SetAt (float) [▶ 2120]

SetAt (char) [▶ 2121]

SetAt (TcVnDMatch) [▶ 2122]

SetAt (TcVnKeyPoint) [▶ 2123]

SetAt (TcVnPoint2_DINT) [▶ 2124]

SetAt (TcVnPoint2_LREAL) [▶ 2124]

SetAt (TcVnPoint2_REAL) [▶ 2125]

SetAt (TcVnPoint3_LREAL) [▶ 2126]

SetAt (TcVnPoint3_REAL) [▶ 2127]

SetAt (TcVnRectangle_DINT) [▶ 2127]

SetAt (TcVnVector2_DINT) [▶ 2128]

SetAt (TcVnVector2_INT) [▶ 2129]

SetAt (TcVnVector2_REAL) [▶ 2130]

SetAt (TcVnVector2_SINT) [▶ 2130]

SetAt (TcVnVector2_UINT) [▶ 2131]

SetAt (TcVnVector2_USINT) [▶ 2132]

SetAt (TcVnVector3_INT) [▶ 2133]

SetAt (TcVnVector3_REAL) [▶ 2133]

SetAt (TcVnVector3_SINT) [▶ 2134]

SetAt (TcVnVector3_UINT) [▶ 2135]

SetAt (TcVnVector3_USINT) [▶ 2136]

SetAt (TcVnVector4_DINT) [▶ 2136]

SetAt (TcVnVector4_INT) [▶ 2137]

SetAt (TcVnVector4_LREAL) [▶ 2138]

SetAt (TcVnVector4_REAL) [▶ 2144]

SetAt (TcVnVector4_SINT) [▶ 2139]

SetAt (TcVnVector4_UINT) [▶ 2139]

SetAt (TcVnVector4_USINT) [▶ 2140]

SetAt (ULONG) [▶ 2141]

SetAt (USHORT) [▶ 2141]

SetAt (ULONGLONG) [▶ 2142]

SetAt (unsigned char) [▶ 2143]

SetContainer [▶ 2163]

SetImageChannel [▶ 2187]

SetIteratorToBegin [▶ 2164]

SetPixel [▶ 2187]

SetPixels [▶ 2188]

SetRngSeed [▶ 2634]

SetRoi [▶ 2189]

SetRoi (TcVnRectangle_DINT) [▶ 2190]

SetRoi (TcVnRectangle_UDINT) [▶ 2190]

Sigmoid [▶ 2651]

SobelFilter [▶ 2502]

SoftMax [▶ 2652]

SplitImageChannels [▶ 2191]

StartAbsWatchdog [▶ 2313]

StartRelWatchdog [▶ 2313]

StopWatchdog [▶ 2314]

SubtractContainers [▶ 1949]

SubtractImageFromScalar [▶ 1982]

SubtractImageFromVector [▶ 1982]

SubtractImages [▶ 1983]

SubtractScalarFromImage [▶ 1984]

SubtractVectorFromImage [▶ 1985]

Threshold [▶ 2508]

TrainImageColor [▶ 2479]

TrainImageColor (ITcVnMlModel*) [▶ 2482]

TransformCoordinatesPlanar (Container) [▶ 2390]

TransformCoordinatesPlanar (Point) [▶ 2391]

TransformIntoDisplayableImage [▶ 2192]

UpdateTimestamp [▶ 2635]

UprightBoundingRectangle [▶ 2312]

UprightInnerRectangle [▶ 2464]

VarianceFilter [▶ 2503]

WarpAffine [▶ 2425]

WarpAffine (Container) [▶ 2426]

WarpAffine (Point) [▶ 2427]

WarpAffine (Rectangle) [▶ 2428]

WarpAffine (Exp) [▶ 2429]

WarpPerspective [▶ 2430]

WarpPerspective (Container) [▶ 2430]

WarpPerspective (Point) [▶ 2431]

WarpPerspective (Rectangle) [▶ 2432]

WarpPerspective (Exp) [▶ 2433]

WatershedSegmentation (Exp) [▶ 2509]

WhiteBalance [▶ 2485]

TC3 Vision Code Reading

ReadBarcode [▶ 2205]

ReadBarcode (Exp) [▶ 2206]

ReadDataMatrixCode [▶ 2208]

ReadPharmaCode [▶ 2209]

ReadQRCode [▶ 2210]

TC3 Vision Code Quality

GradeBarcode [▶ 2197]

GradeDataMatrixCode [▶ 2199]

GradeQRCode [▶ 2201]

TC3 Vision Metrology 2D

AdjustSearchWindowOrientationToLinearEdge [▶ 2593]

CalibrateCamera [▶ 2363]

CalibrateCamera (Exp) [▶ 2364]

CalibrateCameraManually [▶ 2368]

CalibrateLinescanCamera [▶ 2372]

ClosestPointsBF [▶ 2594]

CompensateLensDistortion [▶ 2375]

CompensateLensDistortion (Exp1) [▶ 2376]

CompensateLensDistortion (Exp2) [▶ 2377]

CompensateLensDistortionForPoints [▶ 2378]

CompensateLensDistortionForPoints (Exp1) [▶ 2379]

CompensateLensDistortionForPoints (Exp2) [▶ 2380]

DetectPatternPoints [▶ 2381]

GenerateCalibrationPatternReferencePoints [▶ 2384]

ImagePointsWorldDistance [▶ 2385]

LocateCircularArc [▶ 2595]

LocateEdge [▶ 2598]

LocateEdges [▶ 2602]

LocateEllipse [▶ 2605]

MeasureAngleBetweenEdges [▶ 2608]

MeasureEdgeDistance [▶ 2611]

MeasureMinEdgeDistance [▶ 2615]

SolvePnP [▶ 2386]

SortDetectedPatternPoints [▶ 2388]

TransformCoordinatesImageToWorld (Container) [▶ 2388]

TransformCoordinatesImageToWorld (Point) [▶ 2389]

TransformCoordinatesWorldToImage (Container) [▶ 2392]

TransformCoordinatesWorldToImage (Point) [▶ 2393]

TC3 Vision Matching

DrawKeypoints [▶ 2324]

DrawMatches [▶ 2330]

FilterGoodMatches [▶ 2512]

FindReferenceKeyPointsInImage [▶ 2513]

FindReferenceKeyPointsInImageAKAZE [▶ 2517]

FindReferenceKeyPointsInImageBRISK [▶ 2521]

FindReferenceKeyPointsInImageORB [▶ 2525]

GetMatchCoordinates [▶ 2529]

KeyPointsAGAST [▶ 2530]

KeyPointsAndDescriptorsAKAZE [▶ 2531]

KeyPointsAndDescriptorsBRISK [▶ 2532]

KeyPointsAndDescriptorsKAZE [▶ 2533]

KeyPointsAndDescriptorsORB [▶ 2534]

KeyPointsFAST [▶ 2535]

KeyPointsGFTT [▶ 2536]

KeyPointsMSER [▶ 2537]

KeyPointsSB [▶ 2538]

MatchContours [▶ 2308]

MatchContours1vsN [▶ 2310]

MatchDescriptorsBF [▶ 2539]

MatchDescriptorsFlannLsh [▶ 2540]

MatchDescriptorsKnnBF [▶ 2541]

MatchDescriptorsKnnFlannLsh [▶ 2544]

MatchImageHuMoments [▶ 2445]

MatchTemplate [▶ 2445]

MatchTemplateAndEvaluate [▶ 2447]

RegionsMSER [▶ 2545]

TC3 Vision Machine Learning

CreateBoostClassifier [▶ 2547]

CreateKmppModel [▶ 2548]

CreateKnnModel [▶ 2549]

CreateLbgModel [▶ 2550]

CreateLdaTransform [▶ 2552]

CreateLdaTransformViaComponentNum [▶ 2553]

CreateNbcModel [▶ 2554]

CreatePcaTransform [▶ 2554]

CreatePcaTransformViaComponentNum [▶ 2555]

CreatePcaTransformViaVariance [▶ 2556]

CreateRTreesModel [▶ 2557]

CreateStaModel [▶ 2559]

CreateSvmModel [▶ 2562]

CreateSvmSgdClassifier [▶ 2565]

FeatureScaling [▶ 2567]

FeatureTransform [▶ 2568]

GetBatchClusters [▶ 2569]

GetBatchNovelties [▶ 2569]

GetClusterCenter [▶ 2570]

GetClusterNum [▶ 2571]

GetFeatureScales [▶ 2572]

GetSampleCluster [▶ 2573]

GetSampleNovelty [▶ 2573]

Granulometry [▶ 2640]

HaralickFeatures [▶ 2641]

InverseFeatureScaling [▶ 2574]

InverseFeatureScaling (float) [▶ 2576]

InverseFeatureTransform [▶ 2576]

PredictBatch [▶ 2577]

PredictSampleClass [▶ 2578]

PredictSampleScalar [▶ 2579]

PredictSampleVector [▶ 2580]

TrainBatch [▶ 2581]

TrainBatchClusters [▶ 2582]

TrainSample [▶ 2583]

TrainSampleClass [▶ 2584]

TrainSampleCluster [▶ 2585]

TrainSampleScalar [▶ 2585]

TrainSampleVector [▶ 2586]

TC3 Vision OCR

OCR [▶ 2638]

TC3 Vision Neural Network

ExecuteNeuralNetwork [▶ 2643]

ExecuteNeuralNetwork_MultiLayerOutput [▶ 2644]

ExecuteNeuralNetwork_SingleLayerOutput [▶ 2645]

GetNeuralNetworkLayerNames [▶ 2646]

6.2.3.2 Algebraic Container Operations

This group contains functions for the element-wise application of algebraic operators to containers [▶ 152].

Functions

Algebra

- AddContainers [▶ 1940]
- AddToContainerElements [▶ 1917]
- DivideContainers [▶ 1943]
- MultiplyContainers [▶ 1945]
- MultiplyWithContainerElements [▶ 1946]
- NegateContainer [▶ 1948]
- SubtractContainers [▶ 1949]

Bitwise operators

- BitwiseAndContainers [▶ 1940]
- BitwiseNotContainer [▶ 1941]
- BitwiseOrContainers [▶ 1942]
- BitwiseXorContainers [▶ 1942]

Statistical features

- MaxContainer [▶ 1944]
- MinContainer [▶ 1944]

6.2.3.2.1 AddToContainerElements

This chapter contains functions to add a value to each container element according to the data type.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	LONG	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with DINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	SHORT	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with INT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	double	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with LREAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	float	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	char	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with SINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stValue	TcVnKeyPoint [▶ 1716]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnKeyPoint elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_DINT [▶ 1660]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_DINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_LREAL [▶ 1660]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_LREAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_REAL [▶ 1660]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint3_LREAL [▶ 1660]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_LREAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint3_REAL [▶ 1660]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stValue	TcVnRectangle_DINT [▶ 1729]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnRectangle_DINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_DINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_DINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_INT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_INT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_REAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_SINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_SINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_UINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_UINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_USINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_USINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_INT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_INT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_REAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_SINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_SINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_UINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_UINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_USINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_USINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_DINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_DINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_INT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_INT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_LREAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_LREAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_SINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_SINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_UINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_UINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_USINT [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_USINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	ULONG	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with UDINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	USHORT	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with UINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	ULONGLONG	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with ULINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	unsigned char	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with USINT elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_REAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_REAL elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]		Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]		Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&		Returns the resulting container
nIntDivideByZeroResult	LONGLONG	0	Value set as result in case of integer division by zero

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	double	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with 1-dimensional elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_LREAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with 2-dimensional elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_LREAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with 3-dimensional elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_LREAL [▶ 1662]&	Value
ipContainer	ITcVnContainer* [▶ 1836]	Container with 4-dimensional elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container (same type as ipSrcContainer)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1836]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1836]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the resulting container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3 Algebraic Image Operations

This group contains functions for applying algebraic operations to images [▶ 149] pixel by pixel.

Addition

- AddImages [▶ 1951]
- AddImagesWeighted [▶ 1952]
- AddScalarToImage [▶ 1953]
- AddVectorToImage [▶ 1953]

Subtraction

- SubtractImages [▶ 1983]
- SubtractImageFromScalar [▶ 1982]
- SubtractImageFromVector [▶ 1982]
- SubtractScalarFromImage [▶ 1984]
- SubtractVectorFromImage [▶ 1985]

Multiplication

- MultiplyImages [▶ 1980]
- MultiplyImageWithScalar [▶ 1980]
- MultiplyImageWithVector [▶ 1981]

Division

- DivideImages [▶ 1967]
- DivideImageByScalar [▶ 1965]
- DivideImageByVector [▶ 1966]
- DivideScalarByImage [▶ 1967]
- DivideVectorByImage [▶ 1968]

Exponentiation and logarithmization

- ElementwiseExp [▶ 1969]
- ElementwiseLog [▶ 1970]

Bitwise AND

- BitwiseAndImages(Exp) [▶ 1954]
- BitwiseAndScalarWithImage [▶ 1956]
- BitwiseAndVectorWithImage [▶ 1956]

Bitwise NOT

- BitwiseNotImage(Exp) [▶ 1957]

Bitwise OR

- BitwiseOrImages(Exp) [▶ 1958]
- BitwiseOrScalarWithImage [▶ 1960]

- BitwiseOrVectorWithImage [▶ 1961]

Bitwise XOR

- BitwiseXorImages(Exp) [▶ 1962]
- BitwiseXorScalarWithImage [▶ 1963]
- BitwiseXorVectorWithImage [▶ 1964]

Cross-fading

- BlendImages [▶ 1965]

Maximum

- MaxImage(Exp) [▶ 1970]
- MaxImageWithScalar(Exp) [▶ 1972]
- MaxImageWithVector(Exp) [▶ 1973]

Minimum

- MinImage(Exp) [▶ 1975]
- MinImageWithScalar(Exp) [▶ 1977]
- MinImageWithVector(Exp) [▶ 1978]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Samples

- Copy image areas [▶ 2758]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]		First source image
ipSrcImage2	ITcVnImage* [▶ 1871]		Second source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1687]	TCVN_ET_SAME_AS_SOURCE	Destination image depth

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image (ET_USINT or ET_REAL)
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image (same type and size as ipSrcImage1)
ipWeights1	ITcVnImage* [▶ 1871]	Weights for ipSrcImage1 (1 channel, ET_REAL)
ipWeights2	ITcVnImage* [▶ 1871]	Weights for ipSrcImage2 (1 channel, ET_REAL)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
fScalar	double	Scalar value
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image (dividend)
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image (divisor)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (REAL or LREAL)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (REAL or LREAL)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	Mask of type USINT

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	First source image (minuend)
ipSrcImage2	ITcVnImage* [▶ 1871]	Second source image (subtrahend)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
fScalar	double	Scalar value
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
aVector	TcVnVector4_LREAL [▶ 1662]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4 Basic Container Operations

This group contains functions for handling containers [▶ 152].

Functions

Creation of containers

- CopyContainer [▶ 2147]
- CopyContainerElementsConditional_ITcVnContainer [▶ 2148]
- CopyContainerElementsConditional_ITcVnForwardIterator [▶ 2148]
- CreateContainer [▶ 2149]
- CreateContainerFromArray [▶ 2150]
- ExtractContainerRange [▶ 2156]
- ReserveContainerMemory [▶ 2162]

Handling of container elements

- AppendToContainer [▶ 1986]
- CheckIfEmpty [▶ 2145]
- ConvertContainerType [▶ 2146]
- EraseFromContainer [▶ 2151]
- FillContainer(Exp) [▶ 2036]
- GetAt [▶ 2062]
- GetContainer(Exp) [▶ 2157]
- GetNumberOfElements [▶ 2159]
- InsertIntoContainer [▶ 2088]
- ReverseContainer [▶ 2162]
- SetAt [▶ 2117]
- SetContainer [▶ 2163]

Exporting container data to the memory area

- ExportContainer [▶ 2151]
- ExportSubContainer [▶ 2154]
- ExportContainer_String [▶ 2152]
- ExportSubContainer_String [▶ 2155]
- ExportContainerSize [▶ 2153]
- ExportSubContainerSize [▶ 2156]

Handling of iterators

- AdvanceIterator [▶ 2144]
- CheckIfIteratorIsAtEnd [▶ 2146]
- GetForwardIterator [▶ 2158]
- GetRandomAccessIterator [▶ 2160]
- IncrementIterator [▶ 2160]
- IteratorDistance [▶ 2161]
- SetIteratorToBegin [▶ 2164]

6.2.3.4.1 AppendToContainer

This chapter contains functions to add an element to a container according to the data type.

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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1836]	Single element to append to ipContainer or container with several elements to be concatenated
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element(s) will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnForwardIterator* [▶ 1827]	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element(s) will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1871]	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1716]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1716]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1660]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1660]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1660]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1660]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1660]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1729]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1662]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container to which the element will be appended

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2 FillContainer

This chapter contains functions to fill a container with a special value according to the data type. This function is available for containers with basic elements.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	LONG	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	SHORT	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
ipValue	ITcVnImage* [▶ 1871]	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
fValue	double	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
fValue	float	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	char	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnDMatch [▶ 1716]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnKeyPoint [▶ 1716]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_DINT [▶ 1660]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_LREAL [▶ 1660]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_REAL [▶ 1660]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint3_LREAL [▶ 1660]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint3_REAL [▶ 1660]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnRectangle_DINT [▶ 1729]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_DINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_INT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_REAL [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_SINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_UINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_USINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_INT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_REAL [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_SINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_UINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_USINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_DINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_INT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_LREAL [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_SINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_UINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_USINT [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	ULONG	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	USHORT	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	ULONGLONG	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nValue	unsigned char	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_REAL [▶ 1662]&	Value to set the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3 FillContainerExp

This chapter contains expert functions that can be used to fill a container with a specific value, even only partially, according to the data type. This function is available for containers with basic elements.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
ipValue	ITcVnImage* [▶ 1871]	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnDMatch [▶ 1716]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnKeyPoint [▶ 1716]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_DINT [▶ 1660]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_LREAL [▶ 1660]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint2_REAL [▶ 1660]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint3_LREAL [▶ 1660]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnPoint3_REAL [▶ 1660]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
stValue	TcVnRectangle_DINT [▶ 1729]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_DINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_INT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_REAL [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_SINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_UINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector2_USINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_INT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_REAL [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_SINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_UINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector3_USINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_DINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_INT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_LREAL [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_SINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_UINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_USINT [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
aValue	TcVnVector4_REAL [▶ 1662]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4 GetAt

The `F_VN_GetAt` functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [▶ 2158].

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with DINT elements
nElement	LONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with INT elements
nElement	SHORT&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Container with ITcVnContainer elements
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the container at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with ITcVnImage elements
ipElement	ITcVnImage* [▶ 1871]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with LREAL elements
fElement	double&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with REAL elements
fElement	float&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with SINT elements
nElement	char&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnDMatch elements
stElement	TcVnDMatch [▶ 1716]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnKeyPoint elements
stElement	TcVnKeyPoint [▶ 1716]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_DINT elements
aElement	TcVnPoint2_DINT [▶ 1660]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_LREAL elements
aElement	TcVnPoint2_LREAL [▶ 1660]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_REAL elements
aElement	TcVnPoint2_REAL [▶ 1660]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_LREAL elements
aElement	TcVnPoint3_LREAL [▶ 1660]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_REAL elements
aElement	TcVnPoint3_REAL [▶ 1660]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnRectangle_DINT elements
stElement	TcVnRectangle_DINT [▶ 1729]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_DINT elements
aElement	TcVnVector2_DINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_INT elements
aElement	TcVnVector2_INT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_REAL elements
aElement	TcVnVector2_REAL [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_SINT elements
aElement	TcVnVector2_SINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_UINT elements
aElement	TcVnVector2_UINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_USINT elements
aElement	TcVnVector2_USINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_INT elements
aElement	TcVnVector3_INT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_REAL elements
aElement	TcVnVector3_REAL [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_SINT elements
aElement	TcVnVector3_SINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_UINT elements
aElement	TcVnVector3_UINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_USINT elements
aElement	TcVnVector3_USINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_DINT elements
aElement	TcVnVector4_DINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_INT elements
aElement	TcVnVector4_INT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_LREAL elements
aElement	TcVnVector4_LREAL [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_SINT elements
aElement	TcVnVector4_SINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_UINT elements
aElement	TcVnVector4_UINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_USINT elements
aElement	TcVnVector4_USINT [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with UDINT elements
nElement	ULONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with UINT elements
nElement	USHORT&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with ULINT elements
nElement	ULONGLONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with USINT elements
nElement	unsigned char&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_REAL elements
aElement	TcVnVector4_REAL [▶ 1662]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5 InsertIntoContainer

This chapter contains functions to insert an element at a specific container position according to the data type.

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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1836]	Single element to insert into ipContainer or container with several elements
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnForwardIterator* [▶ 1827]	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1871]	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1716]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1716]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1660]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1660]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1660]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1660]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1660]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1729]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1662]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1836]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6 SetAt

The `F_VN_SetAt` functions should be used if individual elements in a container have to be accessed. If all elements in a container are to be processed in succession, it is much better to use an iterator [▶ 2158].

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 [▶ 141]	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* [▶ 1836]	Container with DINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with INT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1836]	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with ITcVnContainer elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1871]	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with ITcVnImage elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with LREAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with REAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with SINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1716]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnDMatch elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1716]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnKeyPoint elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1660]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1660]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1660]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1660]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1660]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnPoint3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1729]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnRectangle_DINT elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector2_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector3_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with UDINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with UINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with ULINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]	Container with USINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1662]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1836]	Container with TcVnVector4_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator to be advanced
nOffset	LONGLONG	Offset, by which ipIterator is advanced

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns the converted container
nDestTypeGuid	GUID	Specifies the destination container type for the conversion

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Destination container (same type ID as ipSrcContainer; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnContainer* [▶ 1733]	Custom condition

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnForwardIterator* [▶ 1734]	Custom condition

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]&	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULONGLONG	Number of elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nBufferSize	ULONGLONG&	Output parameter containing the required buffer size

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container with basic elements
nIndex	ULONGLONG	Index of the requested element
nBufferSize	ULONGLONG&	Output parameter containing the required buffer size

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container
ipDestContainer	ITcVnContainer* [▶ 1836]&	Destination container
nFirstIdx	ULONGLONG	Index of the first element to copy
nLastIdx	ULONGLONG	Index of the last element to copy

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator
ipContainer	ITcVnContainer* [▶ 1836]&	Returns the container interface

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator
ipContainer	ITcVnContainer* [▶ 1836]&	Returns the container interface
nOffset	LONG	Offset to increment (>0) or decrement (<0) the iterator afterwards

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container
ipIterator	ITcVnForwardIterator* [▶ 1827]&	Returns the iterator interface

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container
nNumberOfElements	ULONGLONG&	Returns the number of elements in ipContainer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container
ipIterator	ITcVnRandomAccessIterator* [▶ 1833]&	Returns the iterator interface

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIteratorFirst	ITcVnForwardIterator* [▶ 1827]	First iterator
ipIteratorLast	ITcVnForwardIterator* [▶ 1827]	Last iterator
nDistance	LONGLONG&	Returns the computed distance

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container for which to reserve the memory
nElements	ULONGLONG	Number of elements for which the container should reserve memory

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Container to be reversed

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator
ipContainer	ITcVnContainer* [▶ 1836]	Container interface of which the content is to be assigned to the current element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1827]	Iterator

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5 Basic Image Operations

The group contains functions for handling images [▶ 149].

Functions

Creation of images

- ConvertElementType [▶ 2166]
- CreateAssociatedImage [▶ 2171]
- CreateEmptyImage [▶ 2171]
- CreateImage [▶ 2172]
- CreateImageAndSetPixels [▶ 2173]
- CreateImageFromArray [▶ 2174]
- CreateNDimensionalImage [▶ 2193]

- CreateNDimensionalImageAndSetPixels [▶ 2194]
- CreateNDimensionalImageFromArray [▶ 2195]

Image information

- GetImageHeight [▶ 2181]
- GetImageInfo [▶ 2182]
- GetImageWidth [▶ 2182]
- GetLayoutInfo [▶ 2195]
- GetPixelFormat [▶ 2184]

Copying images

- CopyImage [▶ 2167]
- CopyImageRegion [▶ 2168]
- CopyImageRegionToRegion [▶ 2169]

Pixel manipulation

- GetPixel [▶ 2183]
- SetPixel [▶ 2187]
- SetPixels [▶ 2188]

Region of Interest

- GetRoi [▶ 2184]
- ResetRoi [▶ 2186]
- SetRoi [▶ 2189]
- SetRoi_TcVnRectangle_DINT [▶ 2190]
- SetRoi_TcVnRectangle_UDINT [▶ 2190]

Exporting image content to memory area

- ExportImage [▶ 2175]
- ExportImageAsBmp [▶ 2176]
- ExportImageSize [▶ 2178]
- ExportImageAsBmpSize [▶ 2177]

Fusing images

- FuseImages [▶ 2179]
- FuseImagesArray [▶ 2180]

Handling image channels

- CombineImageChannels [▶ 2166]
- GetImageChannel [▶ 2180]
- MixImageChannels [▶ 2185]
- SetImageChannel [▶ 2187]
- SplitImageChannels [▶ 2191]

Displaying images

- CopyIntoDisplayableImage [▶ 2170]
- TransformIntoDisplayableImage [▶ 2192]

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 [▶ 141]	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* [▶ 1871]&	Multi-channel destination image (An appropriate destination image will be created if required.)
nSrcArraySize	USHORT	pSrcImages array size

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 1687]		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	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* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	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* [▶ 1871]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnDisplayableImage* [▶ 1871]&	Returns the displayable image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]&	Returns the created image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1687]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 255)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1687]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 4)
aValue	TcVnVector4_LREAL [▶ 1662]&	Pixel value

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1687]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 255)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image
nBufferSize	ULONGLONG&	Returns the required buffer size in bytes

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]	Source image 1
ipSrcImage2	ITcVnImage* [▶ 1871]	Source image 2
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcImageArray	TcVnArray10_ITcVnImage [▶ 1662]&	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* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
nChannelIndex	USHORT	Index of the requested source image channel

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image
nHeight	ULONG&	Return the height of ipImage

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image
stImageInfo	TcVnImageInfo [▶ 1716]&	Returns a struct describing the image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image
nWidth	ULONG&	Return the width of ipImage

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
aValue	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image
stPixelFormat	TcVnPixelFormat [▶ 1728]&	Returns a struct describing the pixel format

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
stRoi	TcVnRectangle_UDINT [▶ 1730]&	Returns the coordinates of the region of interest

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
ipIndicesFromTo	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
aValue	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aValue	TcVnVector4_LREAL [▶ 1662]&		Pixel value
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]	Destination image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRoi	TcVnRectangle_DINT [▶ 1729]&	Region of interest
ipDestImage	ITcVnImage* [▶ 1871]	Destination image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRoi	TcVnRectangle_UDINT [▶ 1730]&	Region of interest
ipDestImage	ITcVnImage* [▶ 1871]	Destination image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]&		Source image
ipDestImage	ITcVnDisplayableImage* [▶ 1871]&		Returns the displayable image
bAllowDeepCopy	bool	false	Allow deep image copies, if required

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1662]&	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 [▶ 1687]	Pixel type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1662]&	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 [▶ 1687]	Pixel type
fValue	double	Pixel value

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1662]&	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 [▶ 1687]	Pixel type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Source image
aLayoutInfo	TcVnArray33_UDINT [▶ 1662]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.6 Code Quality

This group contains functions for quality assessment of various 1D and 2D codes. The assessment is based on international standards. The code quality check is intended to ensure that the corresponding code can be reliably decoded at the time of use.

In order to obtain comparable results, the standards define methods for measuring and evaluating specific attributes and for deriving an overall assessment of code quality. The aim is to predict, as far as possible, the expected readability of the codes, using different types of readers, in different applications, environments and under different conditions. A minimum acceptable quality level is therefore usually defined for each individual application.

On the one hand, the standards describe requirements that must be met by the systems used in order to obtain usable results. On the other hand, the standards provide recommendations such as illumination, lens settings and geometric structure in order to be able to compare the ratings with other systems. In order to set up a system that conforms to the standard, it is advisable to obtain the relevant quality assessment specifications and the codes to be tested.

The functions can also be used with non-standard systems. The only consequence of this is that the quality determined is not comparable with other systems. In general, minor differences in the individual evaluation categories are to be expected for different systems.

In accordance with the ISO/IEC 15416:2016 standard, the print quality of barcodes for the code types Code 39, Code 128, EAN 8, EAN 13, ITF, UPC-A, UPC-E can be determined using the following functions:

- GradeBarcode [▶ 2197]

In accordance with the ISO/IEC 15415:2011 standard, the print quality of two-dimensional matrix symbols of the ECC200 Data Matrix code and QR code types can be determined using the following functions:

- GradeDataMatrixCode [▶ 2199]
- GradeQRCode [▶ 2201]

Interpretation of the HRESULT

For the Code Grading functions the HRESULT [▶ 141] return values are used as follows:

Code	Name	Description
0x000	S_OK	Function was executed successfully and all evaluations were carried out in accordance with the standard.
0x001	S_FALSE	Function was executed successfully, not all scan lines could be decoded for 1D codes and not all checks could be performed according to the standard for 2D codes. The overall score was therefore set to 0, but the individual results can be evaluated.
0x256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. Individual results can be evaluated depending on the time of termination
0x70C	NOTFOUND	Function was not executed successfully, no scan line could be decoded for 1D codes and the code was not found or could not be decoded for 2D codes.
0x7xx	All error codes	Function was not executed successfully. See: ADS Return Codes [▶ 2879]

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.

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGrades1D [▶ 1714]&		Returns the code grades in the range from 0 (very bad) to 4 (very good)
eBarcodeType	ETcVnBarcodeType [▶ 1664]		Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
fAngleDeg	float&		Returns the approximate clockwise rotation angle in degree
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 1663]	BSD_ANY	Barcode search direction (BSD_ANY first tries horizontal, then vertical)
pipResultVisualization	ITcVnImage* [▶ 1871]*	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* [▶ 1836]*	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* [▶ 1836]*	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 [▶ 141]

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 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.

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	ITcVnContainer* [▶ 1836]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGradesDM [▶ 1714]&		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* [▶ 1871]*	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* [▶ 1836]*	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* [▶ 1836]*	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 [▶ 141]

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 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.

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	ITcVnContainer* [▶ 1836]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGradesQR [▶ 1715]&		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* [▶ 1871]*	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* [▶ 1836]*	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* [▶ 1836]*	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 [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7 Code Reading

This group contains functions for detecting and reading 1D and 2D codes.

Functions

1D codes

- ReadBarcode(Exp) [▶ 2206]
  - CodaBar
  - Code39
  - Code93
  - Code128
  - EAN8
  - EAN13
  - ITF
  - UPCA
  - UPCE
  - Code39Extended
- ReadPharmaCode [▶ 2209]

2D codes

- ReadDataMatrixCode [▶ 2208]
- ReadQRCode [▶ 2210]

Code requirements

In general, a module size of at least three pixels is required; more than three pixels are recommended for more stable results. The image should be sharp and have a high contrast. In addition, a code orientation of approx. 0° or 90° is recommended so that the code is parallel to an image axis.

Interpretation of the HRESULT

For the Code Reading functions the HRESULT [▶ 141] return values are used as follows:

Code	Name	Description
0x000	S_OK	Function was successfully executed and all expected codes were found (at least one code for the standard functions and nCodeNumber for the expert functions).
0x001	S_FALSE	Function was executed successfully, not all scan lines could be decoded for 1D codes and not all checks could be performed according to the standard for 2D codes. The overall score was therefore set to 0, but the individual results can be evaluated.
0x256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. Some Code Reading functions can return partial results (see individual function descriptions).
0x7xx	All error codes	Function was not executed successfully. See: ADS Return Codes [▶ 2879]

Therefore, the usual check with `SUCCEEDED()` is not sufficient to deduce a found code and thus existing data in `ipDecodedData`. The following query can be used for this purpose:

```
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
}
```

Search strategies for 2D codes

For 2D codes the search algorithms are applied to different variants of the input image in order to recognize as many codes as possible. Both inversion and mirroring of the image are used to generate various variants. The decision as to which variant is tested is called the search strategy. So that you have greater control over the runtime of the functions, this search strategy can be configured. The enum ETcVn2dCodeSearchStrategy [▶ 1662] is available for this purpose. With this you can specify for each possible image transformation (inversion and mirroring) whether it is to be applied and whether the original image or the transformed image should be checked first for codes.

The function parameter `eSearchStrategy` defines precisely this search strategy for corresponding code reading expert functions. The tables below show the different combinations:

Table 3: Inversion

eSearchStrategy	Original image	Inverted image
CSS_ONLY_NOT_INVERTED	1.	-
CSS_FIRST_NOT_INVERTED	1.	2.
CSS_ONLY_INVERTED	-	1.
CSS_FIRST_INVERTED	2.	1.

Table 4: Mirroring

eSearchStrategy	Original image	Mirrored image
CSS_ONLY_NOT_FLIPPED	1.	-
CSS_FIRST_NOT_FLIPPED	1.	2.
CSS_ONLY_FLIPPED	-	1.
CSS_FIRST_FLIPPED	2.	1.

A search strategy can be defined for each type of transformation. The search strategies of the different search strategies can be linked as follows:

```
eSearchStrategy = CSS_ONLY_FLIPPED | CSS_FIRST_INVERTED;
```

The `CSS_DEFAULT` option can be used to select a default setting that varies depending on the code type.

Linking of search strategies not possible

The linking of several search strategies for the same transformation type is not permissible. Only one strategy for inversion and one for mirroring can be linked to each other.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1836]&	Returns the decoded code (ContainerType_Vector_String_SINT)
eBarcodeType	ULONG	Types of barcode to search for (ETcVnBarcodeType)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1836]&		Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	ITcVnContainer* [▶ 1836]&		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 [▶ 1663]		Barcode search direction.
pipAngles	ITcVnContainer* [▶ 1836]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT elements, 1 or 3 channels)
ipDecodedData	ITcVnContainer* [▶ 1836]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1836]*	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* [▶ 1836]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1836]*	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* [▶ 1836]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1836]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1836]*	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* [▶ 1836]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8 Container Statistics

This group contains functions for statistical analysis of containers [▶ 152].

Functions

With Euclidean norm

- ContainerAverage [▶ 2276]
- ContainerAverageVariance [▶ 2279]
- MaxElement [▶ 2282]
- MedianElement [▶ 2283]
- MinElement [▶ 2284]

Element-wise

- ContainerAverageElementwise [▶ 2277]
- ContainerAverageVarianceElementwise [▶ 2280]
- MaxElementElementwise [▶ 2212]
- MedianElementElementwise [▶ 2234]
- MinElementElementwise [▶ 2255]

6.2.3.8.1 MaxElementElementwise

This chapter contains functions with which the max value is determined according to the data type for each element. This function is available for containers with basic elements.

For example, with a TcVnPoint2_DINT, the maximum X-value and the maximum Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	LONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	SHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMax	double&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMax	float&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnPoint2_DINT [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnPoint2_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnPoint2_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnPoint3_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnPoint3_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector2_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector3_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector3_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector3_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector3_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector3_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_LREAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	ULONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	USHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMax	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMax	TcVnVector4_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2 MedianElementElementwise

This chapter contains functions to determine the median value according to the data type for each element. This function is available for containers with basic elements.

For example, for a TcVnPoint2_DINT, the median X-value and the median Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	LONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	SHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMedian	double&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMedian	float&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnPoint2_DINT [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnPoint2_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnPoint2_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnPoint3_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnPoint3_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector2_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector3_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector3_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector3_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector3_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector3_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_LREAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	ULONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	USHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMedian	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMedian	TcVnVector4_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3 MinElementElementwise

This chapter contains functions to determine the min value according to the data type for each element. This function is available for containers with basic elements.

For example, with a TcVnPoint2_DINT, the minimum X-value and the minimum Y-value are determined independently of each other, so the values do not have to come from the same point.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	LONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	SHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMin	double&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
fMin	float&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnPoint2_DINT [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnPoint2_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnPoint2_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnPoint3_LREAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnPoint3_REAL [▶ 1660]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector2_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector3_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector3_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector3_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector3_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector3_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_DINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_INT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_LREAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_SINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_UINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_USINT [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	ULONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	USHORT&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
nMin	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1836]	Source container
aMin	TcVnVector4_REAL [▶ 1662]&	Returns the requested element

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 2D points or vectors
aAverage	TcVnVector2_LREAL [▶ 1662]&	Returns the calculated average values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 3D points or vectors
aAverage	TcVnVector3_LREAL [▶ 1662]&	Returns the calculated average values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 4D vectors
aAverage	TcVnVector4_LREAL [▶ 1662]&	Returns the calculated average values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 2D points or vectors
aAverage	TcVnVector2_LREAL [▶ 1662]&	Returns the calculated average values
aVariance	TcVnVector2_LREAL [▶ 1662]&	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 3D points or vectors
aAverage	TcVnVector3_LREAL [▶ 1662]&	Returns the calculated average values
aVariance	TcVnVector3_LREAL [▶ 1662]&	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container with 4D vectors
aAverage	TcVnVector4_LREAL [▶ 1662]&	Returns the calculated average values
aVariance	TcVnVector4_LREAL [▶ 1662]&	Returns the calculated variance values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1826]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1826]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1836]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1826]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9 Contour Analysis

The group contains functions for the analysis of contours:

Functions

Shape approximation

- ApproximatePolygon [▶ 2285]
- ConvexHullPoints [▶ 2298]
- EnclosingCircle [▶ 2302]
- EnclosingRectangle [▶ 2303]
- EnclosingTriangle [▶ 2303]

- FitEllipse [▶ 2304]
- FitLine [▶ 2305]
- UprightBoundingRectangle [▶ 2312]

Points

- CheckIfPointIsInsideContour [▶ 2286]
- ContourExtremePoint [▶ 2292]
- ConvexityDefects [▶ 2299]

Geometrical features

- ContourArea [▶ 2287]
- ContourCenterOfMass [▶ 2288]
- ContourCircularity [▶ 2289]
- ContourConvexity [▶ 2290]
- ContourEccentricity [▶ 2291]
- ContourElongation [▶ 2291]
- ContourInertiaRatio [▶ 2293]
- ContourMoments [▶ 2294]
- ContourOrientation [▶ 2295]
- ContourPerimeter [▶ 2296]
- ContourRoundness [▶ 2297]
- FourierDescriptors [▶ 2307]

Matching

- MatchContours [▶ 2308]
- MatchContours1vsN [▶ 2310]

User-defined operations

- CustomElementWiseContainerOperation_ITcVnContainer [▶ 2300]
- CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 2301]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContour	ITcVnContainer* [▶ 1836]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
ipDestContour	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL)
aPoint	TcVnPoint2_REAL [▶ 1660]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 1660]&	Returns the center of mass of the contour

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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]

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	Source container with 2D points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
eDirection	ETcVnExtremePointDirection [▶ 1688]	Selects the search direction for the extreme point
aExtremePoint	TcVnPoint2_LREAL [▶ 1660]&	Returns the extreme point of the contour, according to eDirection

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
stMoments	TcVnMoments [▶ 1717]&	Returns a struct containing the moments

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]		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 [▶ 1730]&		Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.
eMethod	ETcVnOrientationMethod [▶ 1698]	OM_PCA	Method for calculating the orientation.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]		Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
ipConvexHull	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	Source contour (ContainerType_Vector_TcVnPoint2_DINT; The elements of this container are the individual points.)
ipConvexityDefects	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer1	ITcVnContainer* [▶ 1836]	Container 1
ipContainer2	ITcVnContainer* [▶ 1836]	Container 2
ipContainer3	ITcVnContainer* [▶ 1836]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnContainer* [▶ 1731]	Custom operation on the container elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer1	ITcVnContainer* [▶ 1836]	Container 1
ipContainer2	ITcVnContainer* [▶ 1836]	Container 2
ipContainer3	ITcVnContainer* [▶ 1836]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnForwardIterator* [▶ 1732]	Custom operation on the elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
aCenter	TcVnPoint2_REAL [▶ 1660]&	Returns the center of the circle
fRadius	float&	Returns the radius of the circle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
stRectangle	TcVnRotatedRectangle [▶ 1730]&	Returns the determined rectangle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]	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 [▶ 1662]&	Returns the 3 triangle vertices
fArea	double&	Returns the triangle area

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]	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 [▶ 1730]&	Resulting ellipse, described by a rotated rectangle.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]		Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
aFitLine	TcVnVector4_LREAL [▶ 1662]&		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 [▶ 1685]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1836]	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* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour1	ITcVnContainer* [▶ 1836]		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* [▶ 1836]		Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 1683]		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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRefContour	ITcVnContainer* [▶ 1836]		Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer* [▶ 1836]		Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	ITcVnContainer* [▶ 1836]&		Returns the indexes of the best matches (CTcVnContainer_Vector_ULINT; sorted, first element is best match, i.e. has lowest dissimilarity)
ipDissimilarities	ITcVnContainer* [▶ 1836]&		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 [▶ 1683]		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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1836]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
stRectangle	TcVnRectangle_UDINT [▶ 1730]&	Returns the determined rectangle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.10 Control Functions

This group contains functions for sequential control, e.g. for the handling of watchdogs [▶ 146].

Functions

Watchdogs

- StartAbsWatchdog [▶ 2313]
- StartRelWatchdog [▶ 2313]
- StopWatchdog [▶ 2314]

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
tStop	LONG		Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 1708]	WD_ACC_TYPE_PRODUCT	Accumalation method used for combining the results of multiple functions enclosed by this watchdog

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
tStop	LONG		Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 1708]	WD_ACC_TYPE_PRODUCT	Accumalation method used for combining the results of multiple functions enclosed by this watchdog

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11 Drawing

This group contains functions for painting and describing images.

Handling 4-channel colors

4-channel colors are accepted directly into the result image. No alpha blending or similar takes place.

Functions

Arrow

- DrawArrow [▶ 2316]
- DrawArrow_TcVnVector4_DINT [▶ 2317]

Circles and Arcs

- DrawCircle [▶ 2318]
- DrawCircles [▶ 2319]
- DrawCircularArc [▶ 2320]
- FillCircle [▶ 2340]

Components

- DrawComponents [▶ 2321]

Contours

- DrawContours [▶ 2322]
- FillContours [▶ 2340]

Ellipse

- DrawEllipse [▶ 2323]
- FillEllipse [▶ 2341]

Keypoints and Matches

- DrawKeypoints [▶ 2324]
- DrawMatches [▶ 2330]

Lines

- DrawLine [▶ 2325]
- DrawLines [▶ 2328]
- DrawLine_TcVnVector4_DINT [▶ 2326]

- DrawLine_TcVnVector4_LREAL [▶ 2327]

Orientation

- DrawOrientation(Exp) [▶ 2332]

Points

- DrawPoint [▶ 2333]
- DrawPoints [▶ 2334]

Polygon

- DrawPolygon [▶ 2335]
- FillPolygon [▶ 2342]

Rectangle

- DrawRectangle [▶ 2336]
- DrawRectangle_TcVnRectangle_DINT [▶ 2337]
- DrawRectangle_TcVnRectangle_UDINT [▶ 2338]
- DrawRotatedRectangle [▶ 2339]
- FillRectangle [▶ 2343]
- FillRotatedRectangle [▶ 2344]

Intensity Profile

- PlotIntensityProfile [▶ 2344]

Text

- PutLabel [▶ 2346]
- PutText [▶ 2347]

Painting and filling

For many shapes there is an option to paint both the outline of the shape and the filled shape. Two function groups are available for this purpose (Draw... and Fill...). In addition, setting the outline thickness nThickness to -1 leads to the shape being filled.

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 [▶ 141]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
bDoubleHead	bool		To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 1694]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_DINT [▶ 1662]&		The start and end point of a line segment [x1, y1, x2, y2]
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
bDoubleHead	bool		To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 1694]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipCircles	ITcVnContainer* [▶ 1836]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stCircularArc	TcVnCircularArc [▶ 1713]&		Circular arc definition
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the arc is filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLabelImage	ITcVnImage* [▶ 1871]		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* [▶ 1871]&		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 [▶ 1662]&	{-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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContours	ITcVnContainer* [▶ 1836]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the contours are filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
ipHierarchy	ITcVnContainer* [▶ 1836]	nullptr	Contour hierarchy (ContainerType_Vector_TcVnVector4_DINT)
nMaxLevel	LONG	2147483647	Maximum level of contours to be drawn
aOffset	TcVnPoint [▶ 1660]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stEllipse	TcVnRotatedRectangle [▶ 1730]&		Ellipse to be drawn (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the ellipse is filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer* [▶ 1836]		Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate color image will be created if required.)
aColor	TcVnVector4_LREAL [▶ 1662]&	{-1, -1, -1, -1}	Color to draw the keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)
eFlags	ETcVnDrawMatchesFlags [▶ 1685]	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 [▶ 141]

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 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 [▶ 141]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_DINT [▶ 1662]&		The start and end point of a line segment [x1, y1, x2, y2]
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1662]&		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLines	ITcVnContainer* [▶ 1836]		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* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1871]		Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer* [▶ 1836]		Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage* [▶ 1871]		Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer* [▶ 1836]		Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer* [▶ 1836]		Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate color image will be created if required.)
aMatchColor	TcVnVector4_LREAL [▶ 1662]&	{-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 [▶ 1662]&	{-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* [▶ 1836]	nullptr	Mask to select the matches to be drawn (ContainerType_Vector_SINT; Set to 0 if all matches should be drawn.)
eFlags	ETcVnDrawMatchesFlags [▶ 1685]	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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stOrientation	TcVnRotatedRectangle [▶ 1730]&	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColorMainAxis	TcVnVector4_LREAL [▶ 1662]&	Color of the major axis
nThickness	LONG	Line thickness

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stOrientation	TcVnRotatedRectangle [▶ 1730]&	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColorMainAxis	TcVnVector4_LREAL [▶ 1662]&	Color of the major axis
nThickness	LONG	Line thickness
aColorHorizontalAxis	TcVnVector4_LREAL [▶ 1662]&	Color of the horizontal axis
fFontScale	double	Scaling factor for the text

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		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* [▶ 1871]		Destination image
eShape	ETcVnDrawShape [▶ 1686]		Point shape
aColor	TcVnVector4_LREAL [▶ 1662]&		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 [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1836]		Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
eShape	ETcVnDrawShape [▶ 1686]		Point shape
aColor	TcVnVector4_LREAL [▶ 1662]&		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 [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1836]		Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		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 [▶ 1694]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRectangle_DINT [▶ 1729]&	Rectangle to be drawn
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRectangle_UDINT [▶ 1730]&	Rectangle to be drawn
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRotatedRectangle [▶ 1730]&		Rectangle to be drawn (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&		Color
nThickness	LONG		Line thickness (if negative, the rectangle is filled)
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContours	ITcVnContainer* [▶ 1836]	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* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stEllipse	TcVnRotatedRectangle [▶ 1730]&	Ellipse to be painted (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1836]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRotatedRectangle [▶ 1730]&	Rectangle to be painted (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1871]	Destination image
aColor	TcVnVector4_LREAL [▶ 1662]&	Color

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (USINT, 1 channel. x: position, y: intensity, origin: bottom left. An appropriate destination image will be created if required.)
aStartPoint	TcVnPoint2_REAL [▶ 1660]&		Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 1660]&		End point of the line segment
aBackgroundColor	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Background color
aLineColor	TcVnVector4_LREAL [▶ 1662]&	{255, 255, 255, 255}	Line color
aGridColor	TcVnVector4_LREAL [▶ 1662]&	{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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
sText	PCCH		Text
ipDestImage	ITcVnImage* [▶ 1871]		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 [▶ 1691]	FT_HERSHEY_SIMPLEX	Font type
aFontColor	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Font color
aBackgroundColor	TcVnVector4_LREAL [▶ 1662]&	{255, 255, 255, 255}	Background color
eLineType	ETcVnLineType [▶ 1694]	LT_4_CONNECTED	Line type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
sText	PCCH		Text
ipDestImage	ITcVnImage* [▶ 1871]		Destination image
nX	ULONG		x coordinate (bottom left)
nY	ULONG		y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 1691]		Font type
fFontScale	double		Scaling factor
aColor	TcVnVector4_LREAL [▶ 1662]&		Text color
nThickness	LONG	1	Line thickness
eLineType	ETcVnLineType [▶ 1694]	LT_8_CONNECTED	Line type
bBottomLeftOrigin	bool	false	Sets the image origin to the bottom left corner, if true

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12 Fourier Analysis

This group contains functions for frequency analysis and frequency filtering in images.

Functions

Butterworth filter

- CreateBandpassButterworthFilter [▶ 2348]
- CreateBandrejectButterworthFilter [▶ 2350]
- CreateHighpassButterworthFilter [▶ 2353]
- CreateLowpassButterworthFilter [▶ 2355]

Gaussian filter

- CreateBandpassGaussianFilter [▶ 2349]
- CreateBandrejectGaussianFilter [▶ 2352]
- CreateHighpassGaussianFilter [▶ 2354]
- CreateLowpassGaussianFilter [▶ 2356]

Discrete Fourier transform

- Dft [▶ 2357]
- InverseDft [▶ 2358]
- OptimalDftSize [▶ 2359]

Miscellaneous

- PadImageBorder [▶ 2359]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSpatialImage	ITcVnImage* [▶ 1871]	Source image (elements of type REAL or LREAL, 1 (Re) or 2 (Re + Im) channels)
ipFrequencyImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFrequencyImage	ITcVnImage* [▶ 1871]	Source frequency image (elements of type REAL or LREAL, 1 (packed CCS) or 2 (Re + Im) channels)
ipSpatialImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.12 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipPaddedImage	ITcVnImage* [▶ 1871]&		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 [▶ 1665]	BIM_CONSTANT	Specifies how the pixel values of the padding area are determined
aPaddingValue	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Specifies the padding value if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13 Geometric and Coordinate Transformations

This group contains functions for transforming images and pixels.

Functions

Calibration [▶ 2362]

- Intrinsic and extrinsic calibration
- Compensation of optical distortions
- Transformation between image and world coordinates

Affine Transformation

- ApplyRotationToAffineTransformation [▶ 2395]
- ApplyScalingToAffineTransformation [▶ 2396]
- ApplyTranslationToAffineTransformation [▶ 2397]
- ApplyYAxisInversionToAffineTransformation [▶ 2398]
- DecomposeAffineTransformation [▶ 2407]
- GenerateAffineTransformationUnitMatrix2D [▶ 2413]
- GetAffineTransformation [▶ 2413]
- GetAffineTransformation2D [▶ 2414]
- InvertAffineTransformation [▶ 2417]
- WarpAffine(Exp) [▶ 2425]
- WarpAffine_Container [▶ 2426]
- WarpAffine_Point [▶ 2427]
- WarpAffine_Rectangle [▶ 2428]

Transformation between cartesian and polar coordinates

- ConvertCartesianToPolarAngleImage [▶ 2399]
- ConvertCartesianToPolarAngles [▶ 2400]
- ConvertCartesianToPolarImages [▶ 2401]
- ConvertCartesianToPolarMagnitudeImage [▶ 2402]
- ConvertCartesianToPolarMagnitudes [▶ 2403]
- ConvertCartesianToPolarPoints [▶ 2404]
- ConvertPolarToCartesianImages [▶ 2405]
- ConvertPolarToCartesianPoints [▶ 2406]
- RemapImageToLogPolarSpace [▶ 2420]
- RemapImageToPolarSpace [▶ 2422]

Simple image operations

- FlipImage [▶ 2412]
- PyramidDown [▶ 2419]
- PyramidUp [▶ 2420]
- ResizeImage [▶ 2423]

Image alignment and rotation

- AlignRotatedImageRegion [▶ 2394]
- RotateImage(Exp) [▶ 2424]

Perspective transformation

- DecomposeHomography [▶ 2408]
- GetPerspectiveTransformation [▶ 2416]
- Homography [▶ 2416]

- PerspectiveTransformation [▶ 2418]
- WarpPerspective(Exp) [▶ 2430]
- WarpPerspective_Container [▶ 2430]
- WarpPerspective_Point [▶ 2431]
- WarpPerspective_Rectangle [▶ 2432]

6.2.3.13.1 Calibration

This subgroup contains functions for the correlation between pixels and real world pixels. They are usually used together with the measurement functions [▶ 2587].

Functions

Calibration

- CalibrateCamera(Exp) [▶ 2363]
- CalibrateCameraManually [▶ 2368]
- CalibrateCameraPlanar [▶ 2370]
- CalibrateLinescanCamera [▶ 2372]
- DetectPatternPoints [▶ 2381]
- GenerateCalibrationPatternReferencePoints [▶ 2384]
- SortDetectedPatternPoints [▶ 2388]
- SolvePnP [▶ 2386]

Compensation of lens distortion

- CompensateLensDistortion [▶ 2375]
- CompensateLensDistortionForPoints [▶ 2378]

Transformation between image and world coordinates

- TransformCoordinatesImageToWorld [▶ 2389]
- TransformCoordinatesWorldToImage [▶ 2393]
- TransformCoordinatesPlanar [▶ 2391]
- ImagePointsWorldDistance [▶ 2385]

Requirements for the input images

The input images should not be used in a packed or filtered format, such as Bayer format. In the case of the calibration functions, this has the consequence that the recognition of the patterns is disturbed, which can lead to poorer calibration results or no calibration results at all. Since an interpolation takes place during the compensation functions, in which the original arrangement of the pixels is changed, the image can no longer be converted correctly afterwards. Therefore, it is recommended to always perform a conversion first before using a function from this subgroup.

Requirements for the camera matrix

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

The camera matrix is made up of the focal lengths fx and fy and the principal point (cx,cy), which are expressed in pixel units. The principal point, which is the intersection of the optical axis with the image, is normally located near the center of the image. The following conditions must be met:

- 0 < fx and fy
- cx >= 0 and <= image width -1
- cy >= 0 and <= image height
- The other positions in the matrix must be filled with 0 or 1, as shown in the figure. Other values are not permissible.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image containing a calibration pattern (1 channel, USINT elements)
ipReferencePoints	ITcVnContainer* [▶ 1836]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Returns the translation vector
fReprojError	double&	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1719]&	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1712]&	Calibration options

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		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* [▶ 1836]		Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1719]&		Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1712]&		Calibration options
bSubpixelAccuracy	bool	false	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]	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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 1708]	WCS_X_RIGHT_Y_DOWN	World coordinate system orientation

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1836]		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* [▶ 1836]		Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&		Returns the translation vector (for the first inner container)
fReprojError	double&		Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1712]&		Calibration options
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 1708]	WCS_X_RIGHT_Y_DOWN	World coordinate system orientation

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Binary source image (background zero, markers non-zero)
aMarkerPositions	TcVnArray4_Point2_REAL [▶ 1662]&		Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system
aMarkerImagePositions	TcVnArray4_Point2_REAL [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		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* [▶ 1836]		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 [▶ 1662]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1712]&		Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_APPROX_ERF	Selection of the edge detection algorithm

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Allows additional scaling and shifting of the result image

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Allows additional shifting of the result point coordinates

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel)
ipPatternPoints	ITcVnContainer* [▶ 1836]&		Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1719]&		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 [▶ 1687]	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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipReferencePoints	ITcVnContainer* [▶ 1836]&	Returns the generated reference points (ContainerType_Vector_TcVnPoint3_REAL)
ePattern	ETcVnCalibrationPattern [▶ 1666]	Calibration pattern type
eOrigin	ETcVnCalibrationPatternOrigin [▶ 1666]	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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aImagePoint1	TcVnPoint2_REAL [▶ 1660]&	First point in the image coordinate system
aImagePoint2	TcVnPoint2_REAL [▶ 1660]&	Second point in the image coordinate system
fWorldDistance	double&	Returns the distance between aImagePoint1 and aImagePoint2 in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the given points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1836]		Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer* [▶ 1836]		Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&		Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
eMethod	ETcVnSolvePnPMethod [▶ 1702]	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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1836]	Detected pattern point positions in the image coordinate system (will be sorted by this function, ContainerType_Vector_TcVnPoint2_REAL)
ipPatternPoints	ITcVnContainer* [▶ 1836]	The pattern points in the calibration pattern coordinate system, used as a reference to sort ipImagePoints (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints2D	ITcVnContainer* [▶ 1836]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints3D	ITcVnContainer* [▶ 1836]&	Returns the transformed 3D points (TcVnPoint3_LREAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1660]&	Point in the image coordinate system
aDestPoint	TcVnPoint3_LREAL [▶ 1660]&	Returns the point in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns the transformed 2D points (TcVnPoint2_LREAL)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1660]&	Point in the source coordinate system
aDestPoint	TcVnPoint2_LREAL [▶ 1660]&	Returns point in the destination coordinate system
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints3D	ITcVnContainer* [▶ 1836]	Container with 3D source points (TcVnPoint3_LREAL)
ipDestPoints2D	ITcVnContainer* [▶ 1836]&	Returns the transformed 2D points (TcVnPoint2_REAL or TcVnPoint2_DINT, depending on bInteger)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector
bInteger	bool	Selects whether the function returns a container of TcVnPoint2_DINT or TcVnPoint2_REAL

Return value

HRESULT [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint3_LREAL [▶ 1660]&	3D source point (world coordinate system)
aDestPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the transformed 2D point (image coordinate system)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate image will be created if required.)
stRegion	TcVnRotatedRectangle [▶ 1730]&		Image region to be aligned
eInterpolationType	ETcVnInterpolationType [▶ 1693]	IT_BILINEAR	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1665]	BIM_CONSTANT	Border interpolation method (ISOLATED not supported)
aBorderValue	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1871]		Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1871]		Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1836]		Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1871]		Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1871]		Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	ITcVnImage* [▶ 1871]&		Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1871]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1871]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	ITcVnImage* [▶ 1871]&	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1836]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestMags	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1836]		Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageMag	ITcVnImage* [▶ 1871]		Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage* [▶ 1871]		Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	ITcVnImage* [▶ 1871]&		Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageY	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsPolar	ITcVnContainer* [▶ 1836]		Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 1662]&		Affine transformation matrix
fRotationAngle	double&		Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)
aTranslationVector	TcVnVector2_LREAL [▶ 1662]&	{0, 0}	2D translation vector. The values are in pixels
aScaleVector	TcVnVector2_LREAL [▶ 1662]&	{0, 0}	2D scale vector
aSkewVector	TcVnVector2_LREAL [▶ 1662]&	{0, 0}	2D skew vector. The values are the skew angles in x and y directions in radian

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aHomography	TcVnMatrix3x3_LREAL [▶ 1662]&		Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&		Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 1662]&		Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 1662]&		Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 1662]&		Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 1662]&		Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 1662]&		Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 1662]&		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* [▶ 1836]	nullptr	Container with source points (ContainerType_Vector_TcVnPoint2_REAL)
ipPointsDes	ITcVnContainer* [▶ 1836]	nullptr	Container with destination points (same number as ipPoints1, ContainerType_Vector_TcVnPoint2_REAL)

Name	Type	Default	Description
ipInlierMask	ITcVnContainer* [▶ 1836]	nullptr	A mask marking the inliers (optional, set to 0 if not available; ContainerType_Vector_SINT or ContainerType_Vector_USINT)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
eFlipAxis	ETcVnFlipAxis [▶ 1691]	Selects the axis around which to flip the image

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Resulting affine transformation unit matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoints	TcVnArray3_Point2_REAL [▶ 1662]&	Source points
aDestPoints	TcVnArray3_Point2_REAL [▶ 1662]&	Destination points
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Resulting transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]		Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer* [▶ 1836]		Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 1662]&		Returns the affine transformation matrix, which transforms the source points to the destination points
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1688]	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* [▶ 1836]*	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoints	TcVnArray4_Point2_REAL [▶ 1662]&	Source points
aDestPoints	TcVnArray4_Point2_REAL [▶ 1662]&	Destination points
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Resulting transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]		Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer* [▶ 1836]		Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1662]&		Returns the perspective transformation matrix, which transforms the source points to the destination points
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1688]	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* [▶ 1836]*	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Source affine transformation matrix
aInvertedTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Resulting inverted affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Container with 3D source points (TcVnPoint3_REAL or TcVnPoint3_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns the transformed points (same type as ipSrcPoints)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1662]&	Translation vector

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
aCenter	TcVnPoint2_REAL [▶ 1660]&		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 [▶ 1693]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
aCenter	TcVnPoint2_REAL [▶ 1660]&		Center point for the transformation
fMaxRadius	double	0	Maximum radius for the transformation (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 1693]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate image will be created if required.)
nWidth	ULONG		New width
nHeight	ULONG		New height
eInterpolationType	ETcVnInterpolationType [▶ 1693]		Interpolation type
ePaddingMode	ETcVnPaddingMode [▶ 1698]	PM_NONE	Image padding mode
aBorderValue	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Border value, if TCVN_PM_LETTERBOX is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
eAngle	ETcVnRotationAngle [▶ 1701]	Rotation angle

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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.

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1693]	IT_BILINEAR	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1665]	BIM_CONSTANT	Border interpolation method (ISOLATED not supported)
aBorderValue	TcVnVector4_LREAL [▶ 1662]&	{0, 0, 0, 0}	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1660]&	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 1660]&	Destination point
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]&	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Affine transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1662]&	Affine transformation matrix
nDestWidth	ULONG	Width of the destination image
nDestHeight	ULONG	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 1693]	Interpolation method
bWarpInverse	bool	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1665]	Border interpolation method (ISOLATED not supported)
fBorderValue	double	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1836]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1836]&	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1660]&	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 1660]&	Destination point
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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* [▶ 1836]&	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Perspective transformation matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Perspective transformation matrix
nDestWidth	ULONG	Width of the destination image
nDestHeight	ULONG	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 1693]	Interpolation method
bWarpInverse	bool	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1665]	Border interpolation method (ISOLATED not supported)
fBorderValue	double	Border value, if CONSTANT is used

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14 Image Analysis

This group contains functions for extracting information from images.

Functions

Object detection [▶ 2435]

- Contour search
- Object finding by Hough transformation
- Matching by image comparison

Statistical image features

- ImageAverage [▶ 2455]
- ImageAverageStdDev [▶ 2456]
- ImageAbsoluteAverage [▶ 2463]
- ImageMedian [▶ 2458]
- MaxPixelValue [▶ 2460]
- MinPixelValue [▶ 2461]

Edge detection

- CannyEdgeDetection [▶ 2449]

Finding contiguous image regions

- ConnectedComponents [▶ 2450]
- ConnectedComponentsWithStats [▶ 2451]
- GetConnectedComponent [▶ 2454]

Alignment of a region

- RegionOrientation [▶ 2462]
- UprightInnerRectangle [▶ 2464]

Miscellaneous

- CountNonZeroPixels [▶ 2453]
- DistanceTransformation [▶ 2453]
- ImageCenterOfMass [▶ 2457]
- ImageMoments [▶ 2459]

6.2.3.14.1 Object Detection

This subgroup contains functions for object detection.

Functions

Object detection

- DetectBlobs [▶ 2436]
- FindContours [▶ 2438]
- FindContourHierarchyExp [▶ 2437]
- HoughCircles [▶ 2440]
- HoughLines [▶ 2442]
- HoughLinesP [▶ 2443]

Matching

- MatchImageHuMoments [▶ 2445]
- MatchTemplate [▶ 2445]
- MatchTemplateAndEvaluate [▶ 2447]

Miscellaneous

- AdjustActiveContour [▶ 2435]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Source image (1 channel, USINT elements)
ipActiveContour	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		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 [▶ 1719]&		Parameters to filter the detected contours.
aOffset	TcVnPoint [▶ 1660]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (1 channel, binary)
ipContours	ITcVnContainer* [▶ 1836]&	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* [▶ 1836]&	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 [▶ 1683]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 1682]	Contour encoding
aOffset	TcVnPoint [▶ 1660]&	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel, binary)
ipContours	ITcVnContainer* [▶ 1836]&		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 [▶ 1683]	CRM_EXTERNAL	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 1682]	CAM_SIMPLE	Contour encoding
aOffset	TcVnPoint [▶ 1660]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, gray-level)
ipCircles	ITcVnContainer* [▶ 1836]&		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 [▶ 1692]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage1	ITcVnImage* [▶ 1871]	First image (1 channel)
ipImage2	ITcVnImage* [▶ 1871]	Second image (1 channel)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 1683]	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 [▶ 141]

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 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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage* [▶ 1871]		Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	ITcVnImage* [▶ 1871]&		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 [▶ 1705]	TMM_CCORR_NORMED	Specifies the template match method
ipTemplateMask	ITcVnImage* [▶ 1871]	nullptr	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage* [▶ 1871]		Template image (same type as ipSrcImage, smaller width and height)
ipMatches	ITcVnContainer* [▶ 1836]&		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 [▶ 1705]	TMM_CCORR_NORMED	Specifies the template match method
ipTemplateMask	ITcVnImage* [▶ 1871]	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 [▶ 1693]	IT_BILINEAR	Image resize interpolation type (only used if fScaleFactor != 1, TCVN_IT_BILINEAR recommended for most cases)
pipMatchValues	ITcVnContainer* [▶ 1836]*	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (elements of type USINT)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary)
ipLabelImage	ITcVnImage* [▶ 1871]&		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 [▶ 1699]	PC_8	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 1687]	TCVN_ET_DINT	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 1681]	CCA_GRANA	Selects the applied algorithm

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary)
ipLabelImage	ITcVnImage* [▶ 1871]&		Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	ITcVnContainer* [▶ 1836]&		Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	ITcVnContainer* [▶ 1836]&		Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	ITcVnContainer* [▶ 1836]&		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 [▶ 1699]	PC_8	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 1687]	TCVN_ET_DINT	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 1681]	CCA_GRANA	Selects the applied algorithm

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (1 channel)
nNonZero	ULONGLONG&	Returns the number of non-zero pixels in ipSrcImage

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 1685]		Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 1685]		Size of the distance transformation mask (PRECISE not supported for label computation)
pipDestLabels	ITcVnImage* [▶ 1871]*	nullptr	Returns the component labels (Discrete Voronoi diagram; DINT, 1 channel. An appropriate image will be created if required.)
eLabelType	ETcVnDistanceTransformationLabel [▶ 1684]	DTL_CCOMP	Type of the labels

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLabelImage	ITcVnImage* [▶ 1871]		Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	LONG		Value of the requested label
ipMaskImage	ITcVnImage* [▶ 1871]&		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* [▶ 1836]*	nullptr	Returns the outer contour of the component(ContainerType_Vector_TcVnPoint2_DINT. Contour is optional, set to 0 if not required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 - 4 channels)
aAverage	TcVnVector4_LREAL [▶ 1662]&		Returns the (channel-wise) average pixel value of ipSrcImage
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 - 4 channels)
aAverage	TcVnVector4_LREAL [▶ 1662]&		Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 1662]&		Returns the (channel-wise) pixel value standard deviation of ipSrcImage
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
aCenterOfMass	TcVnPoint2_LREAL [▶ 1660]&		Returns the center of mass of the image
ipMask	ITcVnImage* [▶ 1871]	nullptr	Optional mask (1 channel of type USINT, same width and height as ipSrcImage)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 - 4 channels)
aMedian	TcVnVector4_LREAL [▶ 1662]&		Returns the (approximated, channel-wise) median pixel value of ipSrcImage
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Source image (1 channel, elements of type USINT or REAL)
stMoments	TcVnMoments [▶ 1717]&	Returns a struct containing the moments
bBinaryImage	bool	If true, all non-zero pixels are treated as ones

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]		Source image
aMaxValue	TcVnVector4_LREAL [▶ 1662]&		Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 1660]&		Returns the first found position of aMaxValue (not supported for multi-channel images with ELEMENTWISE)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 1707]	VCM_EUCLIDEAN	Select a vector compare method for multi-channel images

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]		Source image
aMinValue	TcVnVector4_LREAL [▶ 1662]&		Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 1660]&		Returns the first found position of aMinValue (not supported for multi-channel images with ELEMENTWISE)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 1707]	VCM_EUCLIDEAN	Select a vector compare method for multi-channel images

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel, binary)
stOrientation	TcVnRotatedRectangle [▶ 1730]&		Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.
eMethod	ETcVnOrientationMethod [▶ 1698]	OM_PCA	Method for calculating the orientation.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
fAverage	double&	Returns the average absolute pixel value

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary, values > 0 mark valid area)
stRectangle	TcVnRectangle_DINT [▶ 1729]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15 Image Color and Contrast Processing

This group contains functions for color and contrast processing.

Functions

Frequency distribution, intensity and contrast Adjustment of an image

- Clahe [▶ 2467]
- Histogram [▶ 2471]
- HistogramEqualization [▶ 2472]
- NormalizeImage [▶ 2473]
- NormalizeImageForDisplay [▶ 2475]

Look-Up Table and Color Maps

- ApplyColorMap [▶ 2465]
- ApplyLut [▶ 2466]
- GenerateColorMap [▶ 2469]
- GenerateCustomColorMap [▶ 2470]

Color Matching

- ReferenceColorSimilarity [▶ 2475]
- TrainImageColor [▶ 2479]
- TrainImageColor_ITcVnMlModel [▶ 2482]

White balance

- WhiteBalance [▶ 2485]

Miscellaneous

- ConvertColorSpace [▶ 2468]
- InvertImageColor [▶ 2473]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Returns the colored image (TCVN_ET_USINT, 3 channel. An appropriate image will be created if required.)
ipColorMap	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (1-4 channels of 8 or 16 bit types SINT, USINT, INT, UINT)
ipDestImage	ITcVnImage* [▶ 1871]&	Returns the result image (same size, type and channels as ipSrcImage. An appropriate image will be created if required.)
ipLut	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
eTransform	ETcVnColorSpaceTransform [▶ 1670]	Transforms to be applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipColorMap	ITcVnContainer* [▶ 1836]&	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
eColorMap	ETcVnColorMap [▶ 1668]	Selects a color map (similar to GNU Octave/MATLAB types)
eColorMapSize	ETcVnColorMapSize [▶ 1669]	Defines how many elements the generated color map should have

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipColorMap	ITcVnContainer* [▶ 1836]&	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
ipInitialColors	ITcVnContainer* [▶ 1836]	User defined colors (ContainerType_Vector_TcVnVector3_REAL, at least 2 elements)
eColorMapSize	ETcVnColorMapSize [▶ 1669]	Defines how many elements the generated color map should have

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestHistogram	ITcVnContainer* [▶ 1836]&		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* [▶ 1871]	nullptr	Source image mask (TCVN_ET_USINT, 1 channel. Mask is optional, set to 0 if not required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1696]	NT_MINMAX	Normalization type (only INF, L1, L2 or MINMAX)
eDestType	ETcVnElementType [▶ 1687]	TCVN_ET_SAME_AS_SOURCE	Destination image depth (usually SAME_AS_SOURCE)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to restrict the normalization to specific pixel positions (set 0 to normalize the whole image)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
eSignedNormalization	ETcVnSignedNormalization [▶ 1702]	Option for normalizing signed values

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1871]&		Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel* [▶ 1877]		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1871]&		Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel* [▶ 1879]		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1662]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (3 channels (RGB) of type USINT)
ipColorModel	ITcVnColorModel* [▶ 1877]&		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 [▶ 1681]	CTM_LAB	Color training method
ipMask	ITcVnImage* [▶ 1871]	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 [▶ 1668]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (3 channels (RGB) of type USINT)
ipColorModel	ITcVnMlModel* [▶ 1879]&		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 [▶ 1681]	CTM_LAB	Color training method
ipMask	ITcVnImage* [▶ 1871]	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 [▶ 1668]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (3 channels (RGB), USINT or UINT)
pipDestImage	ITcVnImage* [▶ 1871]*	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16 Image Filtering

This group contains functions for filtering images.

Functions

Blurring

- BilateralFilter [▶ 2487]
- BoxFilter [▶ 2488]
- GaussianFilter [▶ 2493]
- MedianFilter [▶ 2497]

Edge detection

- LaplacianFilter [▶ 2494]
- ScharrFilter [▶ 2500]
- SobelFilter [▶ 2502]

User-defined filters

- CustomFilter [▶ 2490]
- SeparableCustomFilter [▶ 2501]

Fill areas

- BrightBorderObjects [▶ 2489]
- DarkBorderObjects [▶ 2491]
- FillHoles [▶ 2492]

Morphological operator

- CreateStructuringElement [▶ 2489]
- MorphologicalOperator [▶ 2497]

Local extremes

- LocalMaxima [▶ 2495]
- LocalMinima [▶ 2496]
- RemoveLocalMaxima [▶ 2498]
- RemoveLocalMinima [▶ 2499]

Local variance

- VarianceFilter [▶ 2503]

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1687]	TCVN_ET_SAME_AS_SOURCE	Destination image depth
aAnchor	TcVnPoint [▶ 1660]&	{-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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipStructuringElement	ITcVnImage* [▶ 1871]&	Returns the created structuring element (Non-zero interface pointers are reused.)
eShape	ETcVnStructuringElementShape [▶ 1703]	Shape of the structuring element (rectangle, cross, or ellipse)
nWidth	ULONG	Width
nHeight	ULONG	Height

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1687]		Destination image depth
stKernel	TcVnMatrix [▶ 1717]&		Custom filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 1660]&	{-1, -1}	Anchor point of the kernel
fDelta	double	0	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1687]		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT or UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT or UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (for nFilterSize 3 or 5: USINT, UINT, REAL. For bigger filters, only USINT is supported.
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (only 1 channel supported for reconstruction operators)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
eOperator	ETcVnMorphologicalOperator [▶ 1695]	Operator type
ipStructuringElement	ITcVnImage* [▶ 1871]	Structuring element to be applied (Typically created via F_VN_CreateStructuringElement.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1687]		Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 1690]		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1687]		Destination image depth
stKernelX	TcVnMatrix [▶ 1717]&		1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 1717]&		1D custom column-filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 1660]&	{-1, -1}	Anchor point of the kernel
fDelta	double	0	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipDestImage	ITcVnImage* [▶ 1871]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1687]		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1665]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17 Image Segmentation

This group contains functions for segmenting images.

Functions

Segmentation

- AdaptiveThreshold [▶ 2505]
- CheckColorRange [▶ 2507]
- DoubleThreshold [▶ 2507]
- Threshold [▶ 2508]

- WatershedSegmentationExp [▶ 2509]
- SauvolaThreshold [▶ 2510]

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel, elements of type USINT)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 1663]	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 [▶ 1706]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate destination image will be created if required.)
aLowerBounds	TcVnVector4_LREAL [▶ 1662]&	Channel-wise lower bounds (Unused channels are ignored.)
aUpperBounds	TcVnVector4_LREAL [▶ 1662]&	Channel-wise upper bounds (Unused channels are ignored.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (For Otsu and Triangle threshold types, only 1 channel of TCVN_ET_USINT is supported)
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 1706]	Threshold type to be applied

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (DINT, 1 channel. An appropriate image will be created if required.)
ipMarkers	ITcVnImage* [▶ 1871]	Marker image (DINT, 1 channel)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
pipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 1706]	Threshold type to be applied. Supported are TCVN_TT_BINARY and TCVN_TT_BINARY_INV.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18 Keypoint Features

This group contains functions for finding and matching distinctive features in an image.

NOTICE

Long computing duration

Many functions for finding keypoint features require a long computing duration. Therefore, pay attention to the set cycle time and if necessary use a watchdog [▶ 146] to secure the real-time behavior.

Functions

Key Points

- KeyPointsAGAST [▶ 2530]
- KeyPointsFAST [▶ 2535]
- KeyPointsGFTT [▶ 2536]
- KeyPointsMSER [▶ 2537]

- KeyPointsSB [▶ 2538]

Key points and descriptors

- KeyPointsAndDescriptorsAKAZE [▶ 2531]
- KeyPointsAndDescriptorsBRISK [▶ 2532]
- KeyPointsAndDescriptorsaKAZE [▶ 2533]
- KeyPointsAndDescriptorsORB [▶ 2534]

Descriptor matching

- FilterGoodMatches [▶ 2512]
- GetMatchCoordinates [▶ 2529]
- MatchDescriptorsBF [▶ 2539]
- MatchDescriptorsFlannLsh [▶ 2540]
- MatchDescriptorsKnnBF [▶ 2541]
- MatchDescriptorsKnnFlannLsh [▶ 2544]

Key Points Referenz Matching

- FindReferenceKeyPointsInImage [▶ 2513]
- FindReferenceKeyPointsInImageAKAZE [▶ 2517]
- FindReferenceKeyPointsInImageBRISK [▶ 2521]
- FindReferenceKeyPointsInImageORB [▶ 2525]

Miscellaneous

- RegionsMSER [▶ 2545]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMatches	ITcVnContainer* [▶ 1836]	Container with descriptor matches (ContainerType_Vector_TcVnDMatch or ContainerType_Vector_Vector_TcVnDMatch)
ipGoodMatches	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcKeyPoints	ITcVnContainer* [▶ 1836]		KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage* [▶ 1871]		Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer* [▶ 1836]		KeyPoints of the reference image (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		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 [▶ 1696]		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 [▶ 1662]&		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 [▶ 1688]	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* [▶ 1836]*	nullptr	Returns a container with good matches (optional, set to 0 if not required; ContainerType_Vector_TcVnDMatch)
pipInlierMask	ITcVnContainer* [▶ 1836]*	nullptr	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1836]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		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 [▶ 1662]&		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 [▶ 1696]	NT_HAMMING	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)

Name	Type	Default	Description
stAkazeParams	TcVnParamsAKAZE [▶ 1719]&	{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 [▶ 1688]	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1836]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		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 [▶ 1662]&		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 [▶ 1696]	NT_HAMMING	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)

Name	Type	Default	Description
stBriskParams	TcVnParamsBRISK [▶ 1723]&	{30, 3, 1}	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1688]	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1836]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1871]		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* [▶ 1836]&		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 [▶ 1662]&		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 [▶ 1696]	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* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
stOrbParams	TcVnParamsORB [▶ 1726]&	{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 [▶ 1688]	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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryKeyPoints	ITcVnContainer* [▶ 1836]	Container with query Keypoints (i.e. keypoints of a reference/ template image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipTrainKeyPoints	ITcVnContainer* [▶ 1836]	Container with training Keypoints (i.e. keypoints of a source/input image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipMatches	ITcVnContainer* [▶ 1836]	Container with matches between query and train Keypoints (CTcVnContainer_Vector_TcVnDMatch).
ipQueryCoordinates	ITcVnContainer* [▶ 1836]&	Returns a container with the coordinates of keypoints from the query Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
ipTrainCoordinates	ITcVnContainer* [▶ 1836]&	Returns a container with the coordinates of keypoints from the training Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsAGAST [▶ 1718]&	{10, true, KDT_AGAST_9_16}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1871]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsAKAZE [▶ 1719]&	{FDT_AKAZE_MLDB, 0, 3, 0.001f, 2, 1, DT1_KAZE_PM_G2}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1871]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsBRISK [▶ 1723]&	{30, 3, 1}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1871]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsKAZE [▶ 1725]&	{false, false, 0.001f, 4, 2, DT1_KAZE_PM_G2}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1871]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsORB [▶ 1726]&	{500, 1.2f, 8, 31, 0, 2, FST_ORB_HARRIS, 31, 20}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsFAST [▶ 1724]&	{10, true, KDT_FAST_9_16}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsGFTT [▶ 1724]&	{1000, 0.01, 1, 3, false, 0.04}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsMSER [▶ 1725]&	{5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	nullptr	Mask to specify, where to look for keypoints (optional, set to 0 if not required)
stParams	TcVnParamsSB [▶ 1727]&	{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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1871]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1871]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1871]	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 [▶ 1696]	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1871]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1871]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1871]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1871]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1836]&		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* [▶ 1871]	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 [▶ 1696]	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1871]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1871]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image
ipRegions	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)
ipBoundingBoxes	ITcVnContainer* [▶ 1836]&		Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
stParams	TcVnParamsMSER [▶ 1725]&	{5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}	Additional expert parameters

Return value

HRESULT [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19 Machine Learning

This group contains functions for creating, training and applying Machine Learning models as well as functions for feature pre-processing. A brief overview of the possible applications of Machine Learning [▶ 143] and further descriptions can be found under the software concept [▶ 137].

Functions for creating a model:

- CreateBoostClassifier [▶ 2547]
- CreateKmppModel [▶ 2548]
- CreateKnnModel [▶ 2549]
- CreateLbgModel [▶ 2550]
- CreateLdaTransform [▶ 2552]
- CreateLdaTransformViaComponentNum [▶ 2553]
- CreateNbcModel [▶ 2554]
- CreatePcaTransform [▶ 2554]
- CreatePcaTransformViaComponentNum [▶ 2555]
- CreatePcaTransformViaVariance [▶ 2556]
- CreateRTreesModel [▶ 2557]
- CreateStaModel [▶ 2559]
- CreateSvmModel [▶ 2562]
- CreateSvmSgdClassifier [▶ 2565]

Functions for training a model:

- TrainBatch [▶ 2581]
- TrainBatchClusters [▶ 2582]
- TrainSample [▶ 2583]
- TrainSampleClass [▶ 2584]
- TrainSampleCluster [▶ 2585]
- TrainSampleScalar [▶ 2585]
- TrainSampleVector [▶ 2586]

Functions for obtaining information on the cluster models:

- GetClusterCenter [▶ 2570]
- GetClusterNum [▶ 2571]

Functions for applying models to a sample:

- GetSampleCluster [▶ 2573]
- GetSampleNovelty [▶ 2573]
- PredictSampleClass [▶ 2578]

- PredictSampleScalar [▶ 2579]
- PredictSampleVector [▶ 2580]

Functions for applying models to a batch:

- GetBatchClusters [▶ 2569]
- GetBatchNovelties [▶ 2569]
- PredictBatch [▶ 2577]

Functions for applying models for feature transformation / dimension reduction:

- FeatureTransform [▶ 2568]
- InverseFeatureTransform [▶ 2576]

Functions for feature normalization:

- GetFeatureScales [▶ 2572]
- FeatureScaling [▶ 2567]
- InverseFeatureScaling [▶ 2574]
- InverseFeatureScaling_REAL [▶ 2576]

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 1665]	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* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 1701]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created model (Non-zero interface pointers are reused.)
eKnnType	ETcVnKnn [▶ 1694]	k-nearest neighbors model type
nK	ULONG	Parameter k used for prediction (number of considered neighbors)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 1701]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer* [▶ 1836]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created model (Non-zero interface pointers are reused.)
eNbcType	ETcVnNbc [▶ 1696]	Normal Bayes classifier type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1885]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1885]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 1702]		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* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 1703]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 1704]		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 [▶ 1704]		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* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]&		Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnSvmSgdClassifierType [▶ 1705]	SSCT_ASGD	Learning algorithm type (default: TCVN_SSCT_ASGD)
eMarginType	ETcVnSvmSgdClassifierMarginType [▶ 1705]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1836]		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* [▶ 1836]		Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFeatureTransform	ITcVnMlModel* [▶ 1879]	Feature transform instance to be used
ipSamples	ITcUnknown* [▶ 1885]	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* [▶ 1885]*	Returns the transformed sample(s) using the same type like ipSamples

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]		Clusterer to be used
ipSamples	ITcUnknown* [▶ 1885]		Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	ITcVnContainer* [▶ 1836]&		Returns the cluster IDs (ContainerType_Vector_DINT)
pipNovelties	ITcVnContainer* [▶ 1836]*	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNoveltyDetector	ITcVnMlModel* [▶ 1879]	Novelty detector to be used
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipNovelties	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]	Clusterer to be used
nCluster	LONG	Cluster ID of the cluster the center of which is requested
ipCenter	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]	Clusterer to be used
nClusterNum	ULONGLONG&	Returns the number of clusters

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1836]	Container holding input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer* [▶ 1836]&	Returns a container with the scaling parameters for each feature (Vector_REAL or Vector_LREAL depending on the type of ipSamples).
eFeatureScalingType	ETcVnFeatureScalingType [▶ 1689]	Feature scaling type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]		Clusterer to be used
ipSample	ITcUnknown* [▶ 1885]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNoveltyDetector	ITcVnMlModel* [▶ 1879]	Novelty detector to be used
ipSample	ITcUnknown* [▶ 1885]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1836]		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* [▶ 1836]		Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fSample	float		Sample value
ipScales	ITcVnContainer* [▶ 1836]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFeatureTransform	ITcVnMlModel* [▶ 1879]	Feature transform instance to be used
ipSamples	ITcUnknown* [▶ 1885]	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* [▶ 1885]*	Returns the transformed sample(s) using the same type like ipSamples

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]		Classifier or regressor to be used
ipSamples	ITcUnknown* [▶ 1885]		Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	ITcVnContainer* [▶ 1836]&		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* [▶ 1836]*	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClassifier	ITcVnMlModel* [▶ 1879]		Classifier to be used
ipSample	ITcUnknown* [▶ 1885]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1879]		Regressor to be used
ipSample	ITcUnknown* [▶ 1885]		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1879]		Regressor to be used
ipSample	ITcUnknown* [▶ 1885]		Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	ITcVnContainer* [▶ 1836]&		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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]	Classifier, clusterer, novelty detector, or regressor to be used
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipOutputs	ITcVnContainer* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]	Clusterer to be used
ipSamples	ITcUnknown* [▶ 1885]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1879]	Clusterer or novelty detector to be used
ipSample	ITcUnknown* [▶ 1885]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClassifier	ITcVnMlModel* [▶ 1879]	Classifier to be used
ipSample	ITcUnknown* [▶ 1885]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nClass	LONG	Class label to be learnt

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1879]	Clusterer to be used
ipSample	ITcUnknown* [▶ 1885]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1879]	Regressor to be used
ipSample	ITcUnknown* [▶ 1885]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fOutput	float	Scalar output to be learnt

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1879]	Regressor to be used
ipSample	ITcUnknown* [▶ 1885]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipOutput	ITcVnContainer* [▶ 1836]	Vectorial output to be learnt (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20 Measurement

The group contains functions for the measurement of objects in images.

Functions

Edge localization

- LocateCircularArc [▶ 2595] for localizing proportional circular arcs
- LocateEdge [▶ 2598] for localizing linear edges
- LocateEdges [▶ 2602] for localizing several linear edges
- LocateEllipse [▶ 2605] for localizing complete ellipses

Geometrical features

- ClosestPointsBF [▶ 2594] for calculating the closest points in two point clouds
- MeasureAngleBetweenEdges [▶ 2608] for measuring the angle between two edges
- MeasureEdgeDistance [▶ 2611] for measuring the average distance between two edges
- MeasureMinEdgeDistance [▶ 2615] for measuring the minimum distance between two edges

Miscellaneous

- AdjustSearchWindowOrientationToLinearEdge [▶ 2593] for aligning the search field to an edge

Parameters for edge localization

The borders of an image object often don't lie exactly on the border between two pixels. Instead, they run independently of pixel grids. Therefore edge localization with sub-pixel accuracy is necessary. This not only considers the transition between two pixels, but also determines the edge position within a pixel on the basis of the overall intensity gradient of an object border.

NOTICE

Don't process the image before measuring

Don't process the image before calling the measurement function. Processing can lead to loss of information and lower accuracy accordingly.

This basic functionality of edge localization is used by all measurement functions. The functions differ mainly by the geometric arrangement in which they perform the edge localization (e.g. LocateEllipse [▶ 2605] for round objects) and the properties that you calculate from them (e.g. MeasureAngleBetweenEdges [▶ 2608] for angle measurement).

Certain parameters must be defined for each function that uses edge localization. These include:

- Definition of the search area: where is the edge sought?
- Definition of edge contrast: What is detected as an edge?
- Definition of the measurement algorithm: How to determine the exact position of the edge.

These parameters are described below. In the case of expert parameters, standard values are specified in addition.

Search area

Parameter	Description
Position of the search lines	The position of the search lines is defined according to the feature to be localized. The position of search lines for a linear edge is defined as a single line with the start point `aStartPoint` and end point `aEndPoint`. The position of search lines for an ellipse is defined by a center point `aCenter` and a radius `fSearchRadius`. The position of search lines for angles is defined by an inner point `aInnerPoint` and two outer points `aOuterPoint1` and `aOuterPoint2`. The position of the search lines should always be set such that the search lines cut the edges to be localized orthogonally. A rectangular search area can be adjusted orthogonally to an edge with the function AdjustSearchWindowOrientationToLinearEdge [▶ 2593].
Number of search lines `nSearchLines`	This specifies how many search lines are to be used for the edge localization. The simplest edge localization has one search line on which the edge must be found. Depending on the geometric feature to be determined, multiple search lines are distributed locally so that several edge transitions are localized. This can be necessary in the following cases: • Not only the position, but also the gradient or the rotation of the edge are required. • The geometrical feature (e.g. ellipse) necessitates several search lines. • An average value is required. When localizing linear edges, the search lines are arranged symmetrically on both sides of the central search line, the central search line being included in `nSearchLines`. Thus, an odd number of search lines is always required.
Distance between the search lines `fSearchlineDist` *(Expert)* Default: `1`	The search lines distance defines the distance between neighboring search lines in pixels (> 0, default 1). This allows flexible balancing between accuracy and computing time.
Inversion of the search direction `bInvertSearchDirection` *(Expert)* Default: `false`	The localization algorithm always searches for the edge from one end of the search line to the other. e.g. searching takes place from aStartPoint to aEndPoint. If it is necessary to invert this search direction due to the image properties (i.e. to start from the other end of the search lines), this parameter helps.

Edge contrast

Parameter	Description
Edge direction `eEdgeDirection`	The edge direction `ETcVnEdgeDirection` determines whether an edge transition from light to dark or from dark to light should be found. The following values are available for this: • `ED_DARK_TO_LIGHT` • `ED_LIGHT_TO_DARK`
Minimum strength `fMinStrength`	The minimum strength defines the minimum difference in intensity required within an edge for it to be found. This makes it possible to control how much contrast an edge must have in order to be found.
Maximum thickness `nMaxThickness` *(Expert)* Default: `10`	The maximum thickness specifies the number of pixels within which `fMinStrength` must be reached. This allows you to control whether only sharp edges or also very blurred edges are to be found.

Algorithm

Parameter	Description
Algorithm eAlgorithm *(Expert)* Default: EDA_INTERPOLATION	This parameter ETcVnEdgeDetectionAlgorithm defines the localization algorithm used. The available options are: • EDA_INTERPOLATION • EDA_APPROX_ERF • EDA_APPROX_GAUSSIAN EDA_APPROX_ERF and EDA_APPROX_GAUSSIAN are slower than the standard algorithm EDA_INTERPOLATION, but can achieve higher accuracy. The prerequisite for this, however, is that the intensity gradient of the edge also corresponds as well as possible to the specified model. The error function erf or the Gaussian function is used as a model. The erf function is well suited for edges if the transition from bright to dark extends over several pixels and remains approximately constant on both sides of the light or dark area respectively.  A Gaussian function is well suited for lines, i.e. a dark-light-dark transition or vice versa. However, the brightness plateau may only be a few pixels wide, otherwise the erf function should be used. 
Subpixels / Iterations nSubpixelsIterations *(Expert)* Default: 10	The meaning of this parameter depends on the localization algorithm used. At EDA_INTERPOLATION: the number of subpixels considered defines the fine granularity with which the search line evaluates the underlying pixel intensities. A value of 10, for example, specifies that 10 measured values are taken over a length of 1 pixel. Recommended: 5 - 10 At EDA_APPROX_ERF and EDA_APPROX_GAUSSIAN: the number of iterations is an abort criterion that defines the maximum number of iterations with which the edge model is to be approximated for the pixel intensities. More iterations increase the runtime; if this is too high the interpolation method should be used. Recommended: 50 - 100
Approximation accuracy fApproxPrecision *(Expert)* Default: 0.001	The approximation accuracy is a further abort criterion of the edge localization. It is only observed if the localization algorithm is EDA_APPROX_ERF or EDA_APPROX_GAUSSIAN. The approximation of the edge or line is aborted if either the number of iterations of nSubPixelIterations is reached or the relative

Parameter	Description
	deviation in pixels is less than `fApproximationPrecision` for two consecutive iterations. Recommended: `0.01` - `0.0001`

Interpretation of the HRESULT

The HRESULT [▶ 141] return values have the following meaning for measurement functions:

Code	Name	Description
0x000	S_OK	Function was successfully executed and an edge was located on all search lines.
0x001	S_FALSE	Function was successfully executed, but edges were not located on each search line.
0x256	S_WATCHDOGTIMEOUT	Function was aborted by the watchdog. In this case, all measurement functions can return partial results.
0x70C	NOTFOUND	Function was executed successfully, but no edge could be found on any search line. It is assumed that a measurement function is only called if it is ensured that the image contains an object with the parameterized edge properties. If none of the search lines are found, this indicates incorrect parameterization or an unsuitable call of the function. Therefore, an error code is output in this case.
0x7xx	All error codes except NOTFOUND	Function was not executed successfully. See: ADS Return Codes [▶ 2879]

The following query, for example, can be used for the processing of the return value:

```
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
}
```

Reducing execution time

The execution time for a fixed search window varies slightly depending on the position of the component in the image. If the edge that is found is closer to the starting point, the execution time is shorter than for an edge further away.

A larger fluctuation can occur with the two approximation algorithms, since these require different iterations in order to approximate the model parameters using the surrounding pixel intensities.

In general, fewer iterations are required if the search lines of the search window hit the edge as orthogonally as possible. If the position and orientation of the objects in the image are subject to little or no fluctuation, the search window can be adjusted accordingly to achieve shorter execution times. In addition, the maximum time required can be reduced by reducing the maximum number of iterations, but this may lead to less accurate results.

In addition, the time can be limited with an external watchdog, which terminates the execution of the function if necessary and returns the existing partial results.

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
aStartPoint	TcVnPoint2_REAL [▶ 1660]&	Position from which to start the search process (in the direction of aEndPoint), which is not changed
aEndPoint	TcVnPoint2_REAL [▶ 1660]&	Position where the search process ends, which is adjusted by this function
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]	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 [▶ 141]

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 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet1	ITcVnContainer* [▶ 1836]	Container with 1st set of 2d points
ipPointSet2	ITcVnContainer* [▶ 1836]	Container with 2nd set of 2d points
fMinDist	double&	Returns the minimum distance between 2 points of the different point sets
aPoint1	TcVnPoint2_LREAL [▶ 1660]&	Returns the point out of ipPointSet1, for which fMinDist is achieved
aPoint2	TcVnPoint2_LREAL [▶ 1660]&	Returns the point out of ipPointSet2, for which fMinDist is achieved

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
stCircularArc	TcVnCircularArc [▶ 1713]&		Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 1660]&		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 [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipContourPoints	ITcVnContainer* [▶ 1836]*	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
ipEdgePoints	ITcVnContainer* [▶ 1836]&		Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
aStartPoint	TcVnPoint2_REAL [▶ 1660]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1660]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
ipEdgePoints	ITcVnContainer* [▶ 1836]&		Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
aStartPoint	TcVnPoint2_REAL [▶ 1660]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1660]&		Position where the search process ends
nNumEdges	ULONG		The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
stEllipse	TcVnRotatedRectangle [▶ 1730]&		Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 1660]&		The expected ellipse center
fSearchRadius	float		Radius around aCenter to search for edges
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipContourPoints	ITcVnContainer* [▶ 1836]*	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 [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
fAngle	float&		Returns the angle between the detected edges (radians or degrees, depending on bAngleInDegrees)
aInnerPoint	TcVnPoint2_REAL [▶ 1660]&		Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 1660]&		Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 1660]&		Position where the search process for edge 2 ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1836]*	nullptr	Returns the detected edge points between aInnerPoint and aOuterPoint1 (ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1836]*	nullptr	Returns the detected edge points between aInnerPoint and aOuterPoint2 (ContainerType_Vector_TcVnPoint2_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
fAvgDistance	float&		Returns the average distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 1660]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1660]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1836]*	nullptr	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1836]*	nullptr	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipDistances	ITcVnContainer* [▶ 1836]*	nullptr	Returns the distances between the detected edge points (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 141]

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 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (1 channel)
fMinDistance	float&		Returns the minimum distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 1660]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1660]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1687]		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 [▶ 1660]&		Returns the point on the edge near aStartPoint, for which the minimum distance is achieved
aPoint2	TcVnPoint2_REAL [▶ 1660]&		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 [▶ 1687]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1836]*	nullptr	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1836]*	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 [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21 Miscellaneous

This group contains further functions that can be used across groups.

Functions

Geometrical auxiliary functions

- ClipLineToBoundary [▶ 2621]
- ClipLineToBoundary_ITcVnImage [▶ 2622]
- ClipLineToBoundary_TcVnRectangle_DINT [▶ 2623]
- LineIntersectionPoint [▶ 2630]
- LineIntersectionPointAndAngle [▶ 2631]
- RotatedRectangleCorners [▶ 2633]
- RotatedRectangleIntersection [▶ 2634]

Timestamp

- GetTimestamp [▶ 2627]
- UpdateTimestamp [▶ 2635]

Auxiliary functions for matrices

- InitMatrixStruct [▶ 2629]
- InvertMatrix3x3 [▶ 2629]
- MultiplyMatrices [▶ 2632]

Converting interfaces

- ConvertITcUnknownToITcVnBitmapExport [▶ 2624]
- ConvertITcUnknownToITcVnContainer [▶ 2625]
- ConvertITcUnknownToITcVnImage [▶ 2625]
- ConvertITcUnknownToITcVnMlModel [▶ 2626]

Initialization function

- CheckFunctionInitialization [▶ 2620]
- DeinitializeFunction [▶ 2627]

Miscellaneous

- HuMomentInvariants [▶ 2628]
- SetRngSeed [▶ 2634]

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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
eFunction	ETcVnInitializableFunction [▶ 1692]	Initializable function
nOptions	ULONGLONG	Initialization options for the function

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 1660]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1662]&	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* [▶ 1871]	Image, from which to derive the boundary
aStartPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 1729]&	Rectangular boundary
aStartPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1660]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1885]&	Source pointer
ipDest	ITcVnBitmapExport* [▶ 1857]&	Destination pointer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1885]&	Source pointer
ipDest	ITcVnContainer* [▶ 1836]&	Destination pointer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1885]&	Source pointer
ipDest	ITcVnImage* [▶ 1871]&	Destination pointer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1885]&	Source pointer
ipDest	ITcVnMlModel* [▶ 1879]&	Destination pointer

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
eFunction	ETcVnInitializableFunction [▶ 1692]	Initializable function
nOptions	ULONGLONG	Initialization options for the function

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipUnknown	ITcUnknown* [▶ 1885]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
nTimestamp	LONGLONG&	Returns the timestamp

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stMoments	TcVnMoments [▶ 1717]&	Moments from which to compute the Hu invariants
aHuInvariants	TcVnArray7_LREAL [▶ 1662]&	Returns the Hu moment invariants

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pSrcBuffer	PVOID	Source buffer
stDestMatrix	TcVnMatrix [▶ 1717]&	Returns completed matrix struct
nRows	ULONG	Matrix rows
nCols	ULONG	Matrix columns
eElementType	ETcVnElementType [▶ 1687]	Type of the matrix elements

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Source matrix
aInvertedMatrix	TcVnMatrix3x3_LREAL [▶ 1662]&	Returns inverted matrix

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine1	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 1662]&	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 [▶ 1660]&	Returns the intersection point of aLine1 and aLine2.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine1	TcVnVector4_LREAL [▶ 1662]&	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 [▶ 1662]&	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 [▶ 1660]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stSrcMatrix1	TcVnMatrix [▶ 1717]&	First source matrix (A)
stSrcMatrix2	TcVnMatrix [▶ 1717]&	Second source matrix (B)
stDestMatrix	TcVnMatrix [▶ 1717]&	Destination matrix (C) (The destination matrix is filled by this function, but the required memory needs to be provided.)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRect	TcVnRotatedRectangle [▶ 1730]&	Rotated rectangle
aCorners	TcVnArray4_Point2_REAL [▶ 1662]&	Returns the 4 corner points

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRect1	TcVnRotatedRectangle [▶ 1730]&	Rotated rectangle 1
stRect2	TcVnRotatedRectangle [▶ 1730]&	Rotated rectangle 2
ipIntersection	ITcVnContainer* [▶ 1836]&	Returns the intersection points (ContainerType_Vector_TcVnPoint2_REAL, non-zero interface pointers are reused)
eIntersection	ETcVnRectangleIntersection [▶ 1701]&	Returns the intersection type

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipUnknown	ITcUnknown* [▶ 1885]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22 OCR

This group contains functions for **O**ptical **C**haracter **R**ecognition.

The OCR functions [▶ 2638] identify characters in an image and return the recognized characters as a string. The classification is based on classic Machine Learning models. The models are provided in a trained form so that the functions can be used directly. No additional settings or training are required. The supported characters and fonts depend on the data used in the training. For this reason, there are various models whose range of functions, as well as the general preconditions and requirements, are described below.

For all characters that are recognized in an image, the models assign each character to a known class with the highest match. The result therefore always consists of the known characters of the model used. Rejection or removal of unknown characters is not included.

The functions only support single-line character strings; several lines must be divided into individual ROIs and read with several function calls.

General requirements for recognizing characters:

- Character height min. 20 pixels
- Line width min. 3 pixels
- Points min. 3 x 3 pixels
- Lines min. 3 x 6 pixels
- Character spacing / separation min. 4 pixels
- Characters must not overlap
- Only horizontal alignment / arrangement of characters max. ± 6° deviation
- The line / contour of a character must not be interrupted
- The characters must not be mirrored or reversed

General requirements for the image:

- ROI only with the characters to be recognized and an interference-free zone around the characters
- Within the ROI, the characters must not be enclosed by another contour such as a rectangle
- Good contrast between character and background
- Homogeneous, non-noisy or disturbed, non-transparent background

Requirements for the fonts:

- Only proportional fonts with equal character widths
- Larger gaps are always recognized as a single space
- Only sans serif fonts such as Arial, Tahoma, Courier, Univers, Frutiger, Verdana, OCR-B
- No mixed fonts
- No dot print or italic fonts

Models:

The enum ETcVnOcrModelType [▶ 1697] allows access to the following models:

- `OMT_NUMBERS`
  - enables the classification of numbers
  - contains the characters of `0-9`
- `OMT_NUMBERS_SC`
  - enables the classification of numbers and special characters

    - contains the characters from 0-9 and 6 special characters . / - : = +
- OMT_UCLETTERS
    - enables the classification of capital letters
    - contains the characters of A-Z
- OMT_NUMBERS_SC_UCLETTERS
    - enables the classification of numbers, special characters and capital letters
    - contains the characters from 0-9, the 6 special characters . / - : = + and A-Z

When using the combined model (OMT_NUMBERS_SC_UCLETTERS), confusion can occur due to the great similarity of certain characters. Examples of this are O and 0, S and 5 and B and 8.

If the position of a number or letter is known, the optional formatting specification (sPattern) and the combination of the separate models should be used as an alternative. Alternatively, the characters of the result string (ipCharacters) can also be analyzed individually, so that you can decide individually whether, for example, a 0 is also accepted instead of O.

Initialize functions

In order to be able to use the OCR function with one or more models, the function must first be initialized with the respective models using the function block InitializeFunction [▶ 2667].

The time required to load the models depends on the performance of the IPC used and the model size and can take several hundred milliseconds or even several seconds. Therefore, cycle time overruns are to be expected with shorter task cycle times.

If the combined model OMT_NUMBERS_SC_UCLETTERS is to be used, the router memory must be set to at least 512 MB due to the file size.

The exact size depends on the other use of the router memory and any existing fragmentation. If the function block InitializeFunction returns a return code 0x80004005, for example, this indicates insufficient router memory. The router memory should therefore be checked first and increased if necessary. After loading a model, a large part of the memory is released again and is available for the application.

Interpretation of the HRESULT

- The standard function returns S_OK if characters are recognized on the image. When using optional parameters, it depends on whether a sPattern has been specified. If no formatting specification has been defined, the return corresponds to the standard function. If a formatting specification has been passed, S_OK is only returned if the recognized characters match the pattern specification.
- If the standard function was executed successfully and no characters were found or if the characters found do not match the optional pattern specification, S_FALSE is returned.
- If the transferred model is missing for an optional sPattern specification, a HRESULT = ADSERR_DEVICE_NOTINIT is returned.
- If several models are passed to the standard function, a HRESULT = ADSERR_DEVICE_INVALIDPARM is always returned, as the function only supports one model per call. When using optional parameters, it depends on whether a sPattern has been specified. If sPattern is empty, only the TCVN_OMT_NUMBERS_SC_UCLETTERS model is used. If this model has not been passed, a HRESULT = ADSERR_DEVICE_INVALIDPARM is returned.

Examples for the use of sPattern

Additional possibilities arise when using the optional parameters sPattern and eOcrOptions. Depending on the combination of characters on the input image, the information from sPattern and eOcrOptions and the recognized characters, S_FALSE is returned if there is a mismatch. The characters of the result string (ipCharacters) can then also be analyzed individually, as S_FALSE is also returned if the length differs, for example. The samples refer to the use of the OMT_NUMBERS_SC_UCLETTERS model.

Characters on the ip-SrcImage	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

6.2.3.22.1 OCR

Detects and recognizes characters in a binary image (white characters on black background).

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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (USINT, 1 channel, binary)
eModel	ULONGLONG		Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	ITcVnContainer* [▶ 1836]&		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* [▶ 1836]*	nullptr	Returns the corresponding bounding boxes of the recognized characters (ContainerType_Vector_TcVnRectangle_DINT, optional, set to 0 if not required)
pipConfidences	ITcVnContainer* [▶ 1836]*	nullptr	Returns the corresponding classification confidences of the recognized characters (ContainerType_Vector_REAL, optional, set to 0 if not required)

Return value

HRESULT [▶ 141]

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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.23 Texture Analysis

This group contains functions for analyzing the texture of an image.

Functions

Granulometric properties

- Granulometry [▶ 2640]

Haralick features

- HaralickFeatures [▶ 2641]

6.2.3.23.1 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Source image (1 channel)
ipIntensityDiffs	ITcVnContainer* [▶ 1836]&	Returns the intensity differences between successive structuring element sizes (CTcVnContainer_Vector_LREAL), which can be used as features for texture description.
eMorphType	ETcVnMorphologicalOperator [▶ 1695]	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.23.2 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1871]	Source image (1 channel, USINT)
ipFeatures	ITcVnContainer* [▶ 1836]&	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 [▶ 141]

Required License

TC3 Vision Machine Learning

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24 Neural Network

This group contains functions for pre- and post-processing, as well as for the execution of neural network models.

Functions

Pre- & Postprocessing

- ConvertDataLayout [▶ 2642]
- ReshapeImage [▶ 2650]

Execution of neural networks

- ExecuteNeuralNetwork [▶ 2643]
- ExecuteNeuralNetwork_SingleLayerOutput [▶ 2645]
- ExecuteNeuralNetwork_MultiLayerOutput [▶ 2644]

Postprocessing

- Sigmoid [▶ 2651]
- SoftMax [▶ 2652]
- NonMaxSuppression [▶ 2647]
- Reduce [▶ 2649]
- ReduceArg [▶ 2650]

Get the layer names

- GetNeuralNetworkLayerNames [▶ 2646]

6.2.3.24.1 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.2 ExecuteNeuralNetwork

Execute a neural network using the provided image as input and provide the output from the last output layer. The function doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1887]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1871]	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* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.3 ExecuteNeuralNetwork_MultiLayerOutput

Execute a neural network using the provided image as input and provide the output from the specified output layers. The function doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork_MultiLayerOutput(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnContainer*&    ipDestImages,
    ITcVnContainer*     ipLayerNames
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1887]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1871]	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* [▶ 1836]&	Container of images containing the output from the specified layers.
ipLayerNames	ITcVnContainer* [▶ 1836]	Container of layer names whose outputs are required (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.4 ExecuteNeuralNetwork_SingleLayerOutput

Execute a neural network using the provided image as input and provide the output from a specified output layer. The function doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork_SingleLayerOutput(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage,
    PCCH                sLayerName
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1887]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1871]	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* [▶ 1871]&	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 the last output layer is provided.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.5 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1887]	Neural network model
ipLayerNames	ITcVnContainer* [▶ 1836]&	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	bool	If true, only the unconnected output layer names are returned.

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Neural Network

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.6 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipBoundingBoxes	ITcVnContainer* [▶ 1836]		Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer* [▶ 1836]		Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	ITcVnContainer* [▶ 1836]&		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* [▶ 1836]	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.7 Reduce

Reduce ipSrcImage by applying the specified operation across rows/columns to obtain a single row/column with the resulting values.

Syntax

Definition:

```
HRESULT Reduce(
    HRESULT         hrPrev,
    ITcVnImage*     ipSrcImage,
    ITcVnImage*&    ipDestImage,
    LONG            nDim,
    ETcVnReduceType eReduceType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (USINT or REAL elements, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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
eReduceType	ETcVnReduceType [▶ 1710]	Reduction operation

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.8 ReduceArg

Finds indices of max/min elements along provided axis and reduce the ipSrcImage to a single row/column.

Syntax

Definition:

```
HRESULT ReduceArg(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    LONG         nDim,
    LONG         eReduceType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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
eReduceType	LONG	Reduction operation REDUCE_MAX or REDUCE_MIN

Return value

HRESULT [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.9 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	Destination image (An appropriate single channel destination image will be created if required.)
aNewShape	TcVnArray33_UDINT [▶ 1662]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.10 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 [▶ 141]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]		Source image (REAL)
ipDestImage	ITcVnImage* [▶ 1871]&		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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.11 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 [▶ 141]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1871]	Source image (REAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1871]&	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 [▶ 141]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4 Function Blocks

Communication with GigE Vision cameras or with the file system as well as complex image processing functions is mapped by function blocks.

Grouping of function blocks

All function blocks of the TwinCAT Vision library are sorted thematically into the following groups:

- Camera Register Access [▶ 2654]
- File Access [▶ 2665]
- Image Acquisition [▶ 2680]
- Image Processing [▶ 2699]

6.2.4.1 License Overview

The following list shows the assignment of TwinCAT Vision function blocks to available licenses (see license model [▶ 12]). It shows which license is required for the application of certain function blocks.



The basic license TC3 Vision Base (TF7100) is always required.

TC3 Vision Base

FileSourceControl [▶ 2682]

GevCameraControl [▶ 2685]

InitializeFunction [▶ 2667]

ReadCalibrationPattern [▶ 2668]

ReadCalibrationResult [▶ 2670]

ReadContainer [▶ 2671]

ReadImage [▶ 2672]

ReadMemory [▶ 2655]

ReadMlModel [▶ 2673]

ReadNeuralNetwork [▶ 2679]

ReadRegister_REAL [▶ 2657]

ReadRegister_UDINT [▶ 2658]

ReadRegister_ULINT [▶ 2659]

SimpleCameraControl [▶ 2696]

WriteCalibrationResult [▶ 2674]

WriteContainer [▶ 2675]

WriteImage [▶ 2677]

WriteMemory [▶ 2660]

WriteMlModel [▶ 2678]

WriteRegister_REAL [▶ 2662]

WriteRegister_UDINT [▶ 2663]

WriteRegister_ULINT [▶ 2664]

TC3 Vision Matching

GeneralizedHoughBallard [▶ 2699]

SSIM [▶ 2702]

6.2.4.2 Camera Register Access

This group contains function blocks for reading and writing camera registers or memory areas.

Function blocks

UDINT register (4 bytes)

- ReadRegister_UDINT [▶ 2658]
- WriteRegister_UDINT [▶ 2663]

ULINT register (8 bytes)

- ReadRegister_ULINT [▶ 2659]
- WriteRegister_ULINT [▶ 2664]

REAL register (4 bytes)

- ReadRegister_REAL [▶ 2657]
- WriteRegister_REAL [▶ 2662]

Larger memory areas

- ReadMemory [▶ 2655]
- WriteMemory [▶ 2660]

Parameter

The following parameters work in a similar way with all function blocks in this group:

Address

The address `nAddress` specifies the register of the camera to be read or saved. The type GVCP_REGISTER_ADDRESS is merely an alias for UDINT; therefore, the address must be 4 bytes (or 32 bits) long. For larger memory areas of the camera (see ReadMemory [▶ 2655]) the address points to the beginning of the area to be read or saved.

Execute the function

The triggers for write and read commands are `bWrite` and `bRead` respectively. They are triggered by a rising edge (change from `false` to `true`). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks

Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout

So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with `nTimeout`. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs `bError` and `nErrorId`.

Working state

The output `bBusy` indicates whether the function block is busy. No new command can be triggered if `bBusy == true`.

By means of a suitable `if` query it is possible to determine whether the execution of the function block is completed:

```
if (!fbReadOrWrite.bBusy)
{
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
}
```

Error state

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [▶ 2879]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING (0x71E)` if the function block is busy (`bBusy == true`).

The hexadecimal value of the error code can be extracted as follows:

```
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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 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 [▶ 1660]		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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3 File Access

This group contains function blocks for reading and writing data on the file system.

Function blocks

Images

- ReadImage [▶ 2672]
- WriteImage [▶ 2677]

Container

- ReadContainer [▶ 2671]
- WriteContainer [▶ 2675]

Machine learning models

- ReadMlModel [▶ 2673]
- WriteMlModel [▶ 2678]

Calibration pattern

- ReadCalibrationPattern [▶ 2668]
- ReadCalibrationResult [▶ 2670]
- WriteCalibrationResult [▶ 2674]

Function initialization

- InitializeFunction [▶ 2667]

Neural Network

- ReadNeuralNetwork [▶ 2679]

General

These function blocks work asynchronously; the time from the first execution to the desired result extends over several cycles. The actual duration of each read/write operation depends on several factors, such as the size of the image and the behavior of the operating system. The following parameters are the same for all function blocks and are therefore described here once centrally.

Parameter

File path

The file path `sFilePath` specifies where to save a file or which file to load. Absolute and relative paths can be used. In the case of relative paths, the default paths set in the Service Configuration [▶ 66] are used as the basis.

The file path is stored internally on a rising edge of the write command `bWrite` so that changes to the path in later cycles have no effect.

Execute the function

The triggers for write and read commands are `bWrite` and `bRead` respectively. They are triggered by a rising edge (change from `false` to `true`). The relevant input parameters are saved internally so that a parameter change during the execution has no effect. A falling edge of the trigger likewise has no effect.

Calling function blocks

Please note that the description of parameters for function blocks only has an effect if the function blocks are actually called.

Timeout

So as not to have to wait a long time for the execution of the function block in the case of an unknown error, the maximum permitted execution time can be set with `nTimeout`. The execution will be aborted if this time is exceeded without a successful result after triggering the command trigger. The timeout error is then recognizable by the two outputs `bError` and `nErrorId`.

Working state

The output `bBusy` indicates whether the function block is busy. No new command can be triggered if `bBusy == true`.

By means of a suitable `if` query it is possible to determine whether the execution of the function block is completed:

```
if (!fbReadOrWrite.bBusy)
{
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
}
```

Error state

The error state is indicated by `bError` binarily indicating an error and `nErrorId` outputting the error code (ADS Return Code [▶ 2879]) if applicable. The error state is reset as soon as a new command is triggered. The error code is `PENDING (0x71E)` if the function block is busy (`bBusy == true`).

The hexadecimal value of the error code can be extracted as follows:

```
if (fbReadOrWrite.bError)
{
    // Show relevant error code for debugging purposes:
    nErrorCode = fbReadOrWrite.nErrorId & 0xFFF;
}
```

The most frequent error codes are:

Hex	Dec	Name	Meaning
0x006	6	TARGETPORTNOTFOUND	ADS server not reachable. Essentially occurs when the TwinCAT Vision Service has not started. See Vision Service doesn't start [▶ 2864].
0x700	1792	ERROR	General error if saving or loading fails. This usually indicates that the specified file format is not supported or that the file path is unreachable. The error usually occurs in the TwinCAT Vision Service [▶ 66].
0x70B	1803	INVALIDPARM	Invalid parameter. This usually indicates that the interface pointer [▶ 137] is invalid.
0x719	1817	TIMEOUT	Timeout. Loading or saving has exceeded the maximum execution time `nTimeout`. In general, this occurs mainly with a heavy load on ADS.
0x71E	1822	PENDING	Not really an error, therefore the output `bError` is not set here. This code signals that the function block is busy. It occurs in parallel to the output `bBusy` if a new rising edge is registered while the function block is busy.

6.2.4.3.1 InitializeFunction

This FB initializes selected functions. Before, it needs to initialize itself, i.e. call the FB until the output bInitialized is true before using it for function initialization.

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 [▶ 1692]		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 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* [▶ 1836]*		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 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 [▶ 1662]*		Returns the camera matrix
paDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]*		Returns the distortion coefficients
paRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]*		Returns the rotation matrix
paTranslationVector	TcVnVector3_LREAL [▶ 1662]*		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 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* [▶ 1836]*		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 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* [▶ 1871]*		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 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* [▶ 1879]*		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 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 [▶ 1662]*		Returns the camera matrix
paDistortionCoefficients	TcVnArray8_LREAL [▶ 1662]*		Returns the distortion coefficients
paRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1662]*		Returns the rotation matrix
paTranslationVector	TcVnVector3_LREAL [▶ 1662]*		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 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* [▶ 1836]*		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 [▶ 1682]	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 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* [▶ 1871]*		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 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* [▶ 1879]*		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 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* [▶ 1887]*		Returns the neural network
paInputShape	TcVnArray33_UDINT [▶ 1662]*		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 PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.4 Image Acquisition

This group contains function blocks for acquiring images with Vision devices (GigE Vision cameras [▶ 69] and File Source [▶ 125] objects).

Function blocks

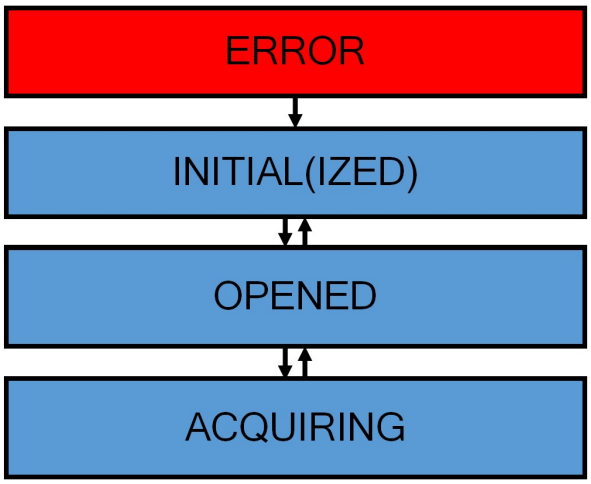
- SimpleCameraControl [▶ 2696] for cameras and File Sources
- GevCameraControl [▶ 2685] for GigE Vision cameras
- FileSourceControl [▶ 2682] for File Sources

General

The function blocks in this group represent a proxy for the Image Provider TcCOM objects. Accordingly, they need not be called cyclically. Instead, the TcCOM object is controlled via the methods of the function block.

State machine

All function blocks for image acquisition include a state machine to manage the state of the represented device. Essentially, a function block must be placed in the ACQUIRING state in order to receive images continuously. The INITIAL state represents the starting point, after a restart or reset, from which the INITIALIZING can be triggered. From the OPENED state, the communication channel to the device is open in order, for example, to read or write register values.



All states are represented by the enum ETcVnCameraState [▶ 1667] and are explained below:

Main states

The main states are stationary states. They essentially indicate whether there is a connection to a Vision device (`OPENED`), whether the device is capturing images (`ACQUIRING`) and whether there is an error with the connection to the device (`ERROR`). You can switch between these states with the methods of the FBs. As a rule, the transition between two main states does not take place immediately, but over at least one cycle. In this intermediate time, the Vision device is in an intermediate state [▶ 2681].

State	Description
ERROR	Error state from which every device can only be reactivated by calling the method `Reset`.
INITIAL	Initial state of each device.
INITIALIZED	Similar to INITIAL, where the camera was manually initialized (e.g. with Force-IP and the initialization commands)
OPENED	There is a connection to the camera, individual images can be triggered and camera registers can be read and written. **This state can be skipped with the method StartAcquisition.**
ACQUIRING	Device is in the acquisition state and is sending images depending on the settings.

Intermediate states

The intermediate states are necessary as most actions cannot be completed immediately (e.g. image acquisition). If the function blocks are in one of these intermediate states, the respective method must be called again in order to complete the transition to the respective main state.

State	Description
INITIALIZING	INITIAL > INITIALIZED
OPENING	INITIAL > OPENED and INITIALIZED > OPENED
STARTACQUISITION	OPENED > ACQUIRING.
STOPACQUISITION	ACQUIRING > OPENED
RESETTINGFEATURES	OPENED > OPENED
TRIGGERING	OPENED > OPENED and ACQUIRING > ACQUIRING
CLOSING	OPENED > INITIALIZED

Methods must be called in intermediate states!

A respective method for state transitions must also be called in the intermediate states. The state transition is only complete and the method no longer needs to be called when a stationary state is detected using the `GetState` method after a corresponding method call. Triggering a method once and subsequently waiting does not result in the desired state transition.

A state machine in C++ can then be implemented either with a `switch` structure or with a `if … else if …` structure.

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 [▶ 2683]	Gets the current available image (if any).
GetCurrentImageAndFileName [▶ 2683]	Gets the current available image (if any) and the corresponding file name.
StartAcquisition [▶ 2683]	Activates the File Source. If not in trigger mode, it will start sending images.
StopAcquisition [▶ 2684]	Deactivates the File Source. It will stop sending images.
Reset [▶ 2684]	Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
GetState [▶ 2684]	Gets the current state of the internal camera control state machine.
TriggerImage [▶ 2684]	If in trigger mode, this function triggers the next image in the list.
TriggerImageExp [▶ 2684]	If in trigger mode, this function triggers a specific image in the list.
TriggerImageByName [▶ 2685]	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 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* [▶ 1871]&	Returns the image

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the image
sFileName	PCHAR	Returns the file name
nMaxLen	USHORT	Define the maximum allowed length for the file name

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.2.4.4.1.4 StopAcquisition

Deactivates the File Source. It will stop sending images.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 1667]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 2688]	Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.
CloseCamera [▶ 2688]	Close the control channel to the camera.
GetCalibPatternRef [▶ 2688]	Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)
GetCameraMatrix [▶ 2689]	Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetCurrentImage [▶ 2689]	Gets the current available image (first in receive queue).
GetCurrentImageUndistorted [▶ 2690]	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 [▶ 2690]	Gets the current available image (first in receive queue) and its corresponding GVSP info.
GetCurrentImageWithGvspInfo Undistorted [▶ 2690]	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 [▶ 2691]	Gets the current available image (first in receive queue) and corresponding timestamps.
GetDistortionCoefficients [▶ 2691]	Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetLastImageFromQueue [▶ 2692]	Gets the last received image from the queue.
GetOmittedImagesNum [▶ 2692]	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 [▶ 2692]	Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
GetState [▶ 2693]	Gets the current state of the internal camera control state machine.
GetTranslationVector [▶ 2693]	Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
InitializeCamera [▶ 2693]	Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
OpenCamera [▶ 2694]	Open a control channel to the camera.
Reset [▶ 2694]	Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
ResetCameraFeatures [▶ 2694]	Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
SetCameraMatrix [▶ 2694]	Sets the camera matrix to the TcVnGevImageProvider TcCOM module
SetDistortionCoefficients [▶ 2695]	Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Name	Description
SetRotationMatrix [▶ 2695]	Sets the rotation matrix to the TcVnGevImageProvider TcCOM module
SetTranslationVector [▶ 2695]	Sets the translation vector to the TcVnGevImageProvider TcCOM module
StartAcquisition [▶ 2696]	Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
StopAcquisition [▶ 2696]	Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
TriggerImage [▶ 2696]	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 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 [▶ 141]

6.2.4.4.2.2 CloseCamera

Close the control channel to the camera.

Syntax

Definition:

```
HRESULT CloseCamera()
```

Return value

HRESULT [▶ 141]

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* [▶ 1836]&	Returns the reference calibration pattern points (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	Returns the camera matrix

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the image

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the undistorted image

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the image
stGvspInfo	GVSP_IMAGE_INFO [▶ 1710]&	Returns the GVSP info

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the undistorted image
stGvspInfo	GVSP_IMAGE_INFO [▶ 1710]&	Returns the GVSP info

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	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 [▶ 1710]&	Returns the GVSP info

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	Returns the distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 141]

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* [▶ 1871]&	Returns the image

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 1662]&	Returns the rotation matrix

Return value

HRESULT [▶ 141]

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 [▶ 1667]

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 [▶ 1662]&	Returns the translation vector

Return value

HRESULT [▶ 141]

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 [▶ 141]

6.2.4.4.2.17 OpenCamera

Open a control channel to the camera.

Syntax

Definition:

```
HRESULT OpenCamera()
```

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 1662]&	The camera matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The distortion coefficients to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The rotation matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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 [▶ 1662]&	The translation vector to be copied to the TcCOM module

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 2697]	Get the current available image (if any).
GetState [▶ 2697]	Get the current state of the internal camera control state machine.
Reset [▶ 2698]	Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)
StartAcquisition [▶ 2698]	Start the image acquisition.
StopAcquisition [▶ 2698]	Stop the image acquisition.
TriggerImage [▶ 2698]	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 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* [▶ 1871]&	Returns the image

Return value

HRESULT [▶ 141]

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 [▶ 1667]

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 [▶ 141]

6.2.4.4.3.4 StartAcquisition

Start the image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 141]

6.2.4.4.3.5 StopAcquisition

Stop the image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 141]

6.2.4.4.3.6 TriggerImage

Trigger the next image.

Syntax

Definition:

```
HRESULT TriggerImage()
```

Return value

HRESULT [▶ 141]

6.2.4.5 Image Processing

This group contains function blocks for image processing. In these comprehensive function blocks, the values once set are retained after an execution. Furthermore, the function blocks can also contain several methods.

The functions contained in the chapter Functions [▶ 1890], on the other hand, have no internal status information and the variable values are not retained.

Function blocks

- GeneralizedHoughBallard [▶ 2699]
- SSIM [▶ 2702]

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 [▶ 2699]	Detect the template in an image (SetTemplate must have been called before).
SetCannyThreshold [▶ 2700]	Sets the canny edge detection thresholds.
SetInvAccuRatio [▶ 2700]	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 [▶ 2701]	Sets the number of R-table levels.
SetMinDist [▶ 2701]	Sets the minimum distance between the centers of different objects.
SetTemplate [▶ 2701]	Sets the template image to search for in the Detect method.
SetVotesThreshold [▶ 2702]	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 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* [▶ 1871]	Source image (1 channel, USINT)
ipPositions	ITcVnContainer* [▶ 1836]&	Returns the centers of the detected template positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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 [▶ 141]

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* [▶ 1871]	Template image (1 channel, USINT)

Return value

HRESULT [▶ 141]

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 [▶ 141]

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 [▶ 2703]	Compute the structural similarity (SetReference must have been called before).
SetReference [▶ 2703]	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 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* [▶ 1871]	Source image (USINT)
ipSSIM	ITcVnImage* [▶ 1871]&	Returns the SSIM (REAL, values from 0 to 1)

Return value

HRESULT [▶ 141]

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* [▶ 1871]	Reference image (USINT)

Return value

HRESULT [▶ 141]

7 TwinCAT HMI packages

This chapter deals with the interaction of TwinCAT Vision and the TwinCAT HMI. This consists of a server extension [▶ 2704], with which images [▶ 149] from the PLC can be displayed in the TwinCAT HMI, and a controls package [▶ 2706] for visualization and interaction with images and colors.

Integration of packages

The integration of packages takes place in the case of TwinCAT HMI versions from 1.12 via the package management NuGet. In a TwinCAT HMI project, right-click **References** and select **Manage NuGet Packages…**. Then click on **Browse** and select "TwinCAT HMI Official" or "Nuget.org" as **Package source**. You will now see a list of the available TwinCAT HMI packages, from which you can select **Beckhoff.TwinCAT.HMI.Vision** or **Beckhoff.TwinCAT.HMI.Vision.Controls**, for example.

From TwinCAT HMI version 1.14, the **TwinCAT.HMI.Vision** and **TwinCAT.HMI.Vision.Controls** packages can be installed via the TwinCAT Package Manager. The packages must then be integrated into the HMI project via the "Beckhoff Offline Packages" **package source**, as described above.

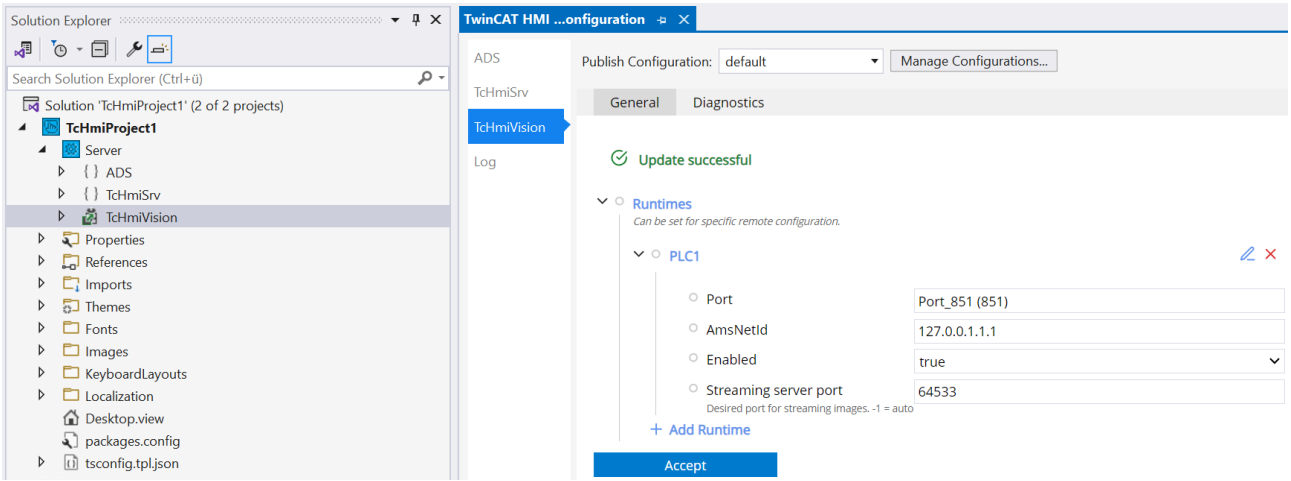
General information can be found in the TwinCAT HMI documentation, e.g. with the detailed description of package management.

7.1 Server extension

The NuGet package Beckhoff.TwinCAT.HMI.Vision (TwinCAT package 'TwinCAT.HMI.Vision') contains the HMI server extension `TcHmiVision`, which can be used to display images from the PLC in the TwinCAT HMI. For this purpose, all ADS symbols of the type ITcVnDisplayableImage [▶ 411] from the configured runtimes are listed and can be linked in various formats with an HMI symbol. See System requirements [▶ 15] for the required .NET version.

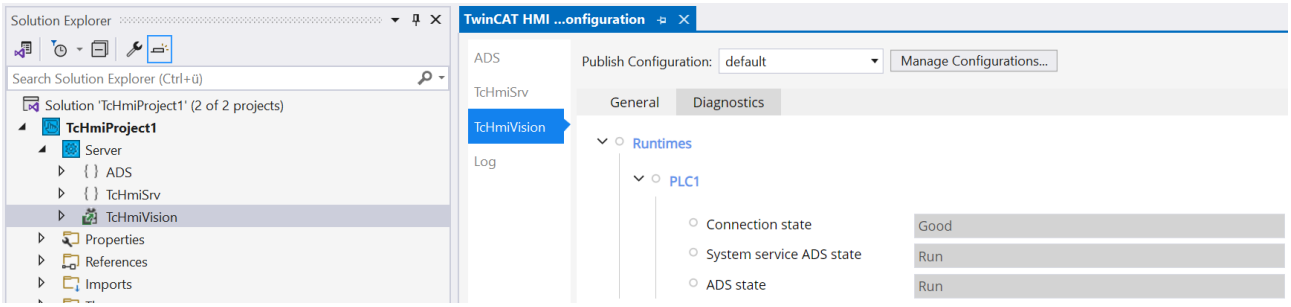
Configuration

Once the package has been successfully integrated into the HMI project, enter the target PLC under **Server > TcHmiVision > General**:



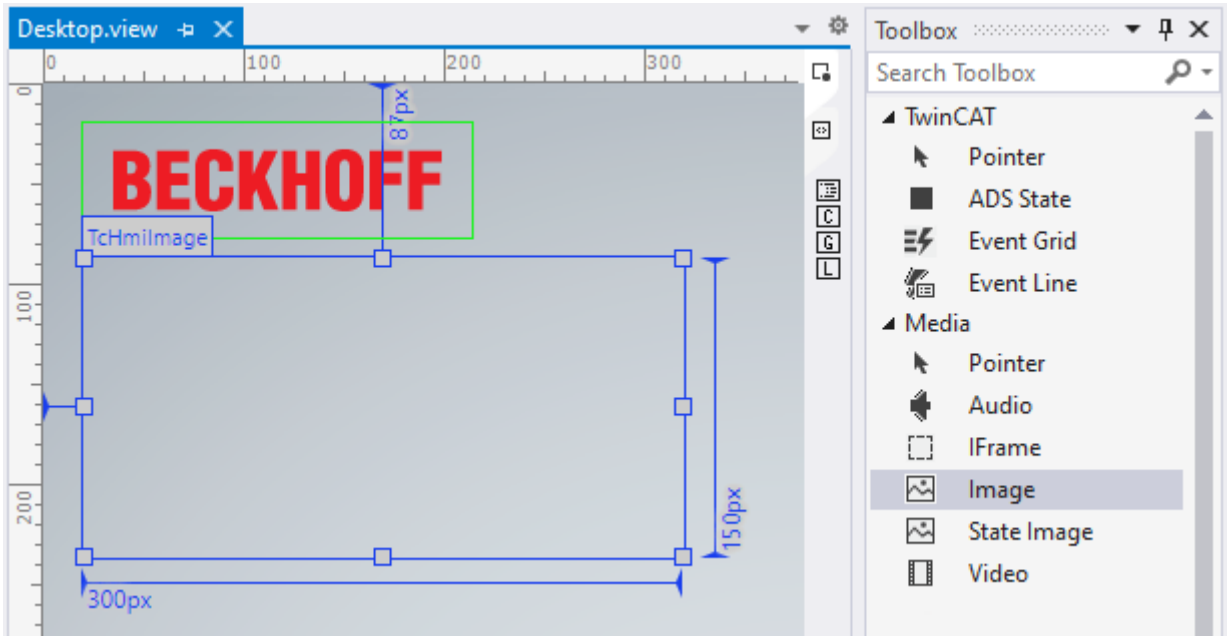
Diagnostics

Under **Server > TcHmiVision > Diagnostics**, you can obtain information on the connection status of the configured target system:

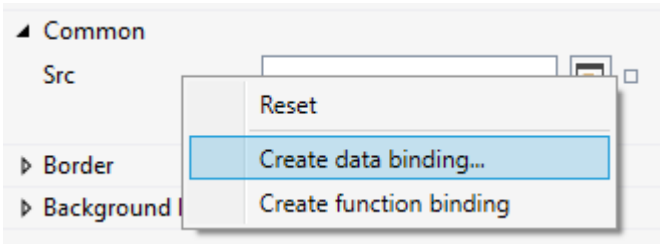


Application

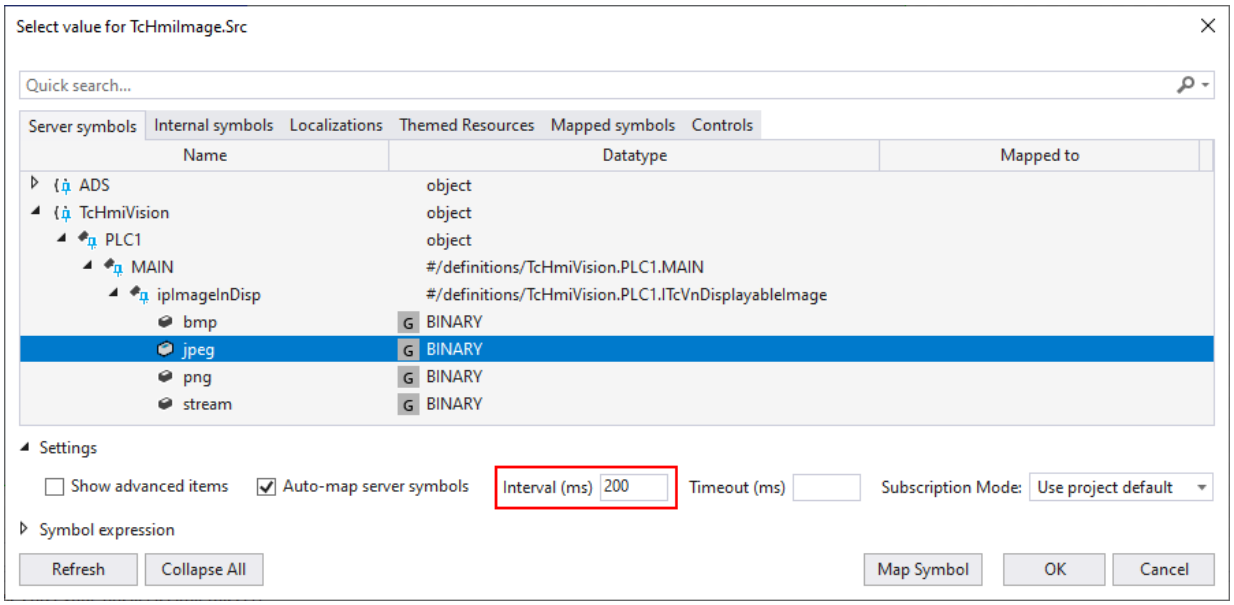
1. Navigate to an HMI view and add an **Image** control.



2. Open the properties of the control element and click on **Create data binding…** under the parameter **Src…**



3. Link the Src parameter of the Image control to the corresponding image symbol at the following location:
   Server symbols > TcHmiVision > PLC > MAIN > ipImage > <image format>



Linking images

Linking images is only possible if the process image of the configured target system contains corresponding image variables.

Interval

The speed at which images can be retrieved from the PLC depends on the system, image size, and workload. We therefore recommend specifying a fixed interval in the Binding dialog at **Settings** to prevent the system from being overloaded.

If several images are to be displayed on one page, it is also recommended to use different intervals, e.g. 200 ms and 201 ms. This way, the images are retrieved separately from each other, which can be advantageous for larger amounts of data.

Transmission formats

The server extension contains the image data from the PLC in BMP format and forwards them as a Base64 string to the TwinCAT HMI client. In order to reduce the data quantity transmitted to a client, images can be compressed in the server extension. The following image formats are available:

- **BMP**: uncompressed
- **JPEG**: lossy compressed, good for natural images
- **PNG**: compressed (lossless), good for artificially created images
- **Stream**: for continuous image streams (e.g. live camera picture, not recommended!)

For transmission as a stream the HMI server must be started as administrator. Otherwise no image transmission is possible.

7.2 Controls

The NuGet package Beckhoff.TwinCAT.HMI.Vision.Controls (TwinCAT package "TwinCAT.HMI.Vision.Controls") provides some controls for displaying and manipulating Vision data. This documentation assumes basic knowledge of TwinCAT HMI concepts see TE2000 and refers to them. A detailed description of the use of controls can also be found there.

Controls

The package contains the following controls that can be added to HMI pages via the HMI Toolbox:

Control	Category	Description
Image Watch [▶ 2707]	Vision	• Displaying images • Zoom and drag options • Extracting image information • Marking image areas with different shapes • Multi-image selection • Overlay graphics • Toolbar with operating functions • Infobar for displaying image and shape information
Color [▶ 2731]	Vision	Displaying and setting colors
Rectangle (Interactive) [▶ 2744]	Shapes	Interactive display and drawing of a rectangle.
Polygon (Interactive) [▶ 2746]	Shapes	Interactive display and drawing of a polygon.

Shape controls

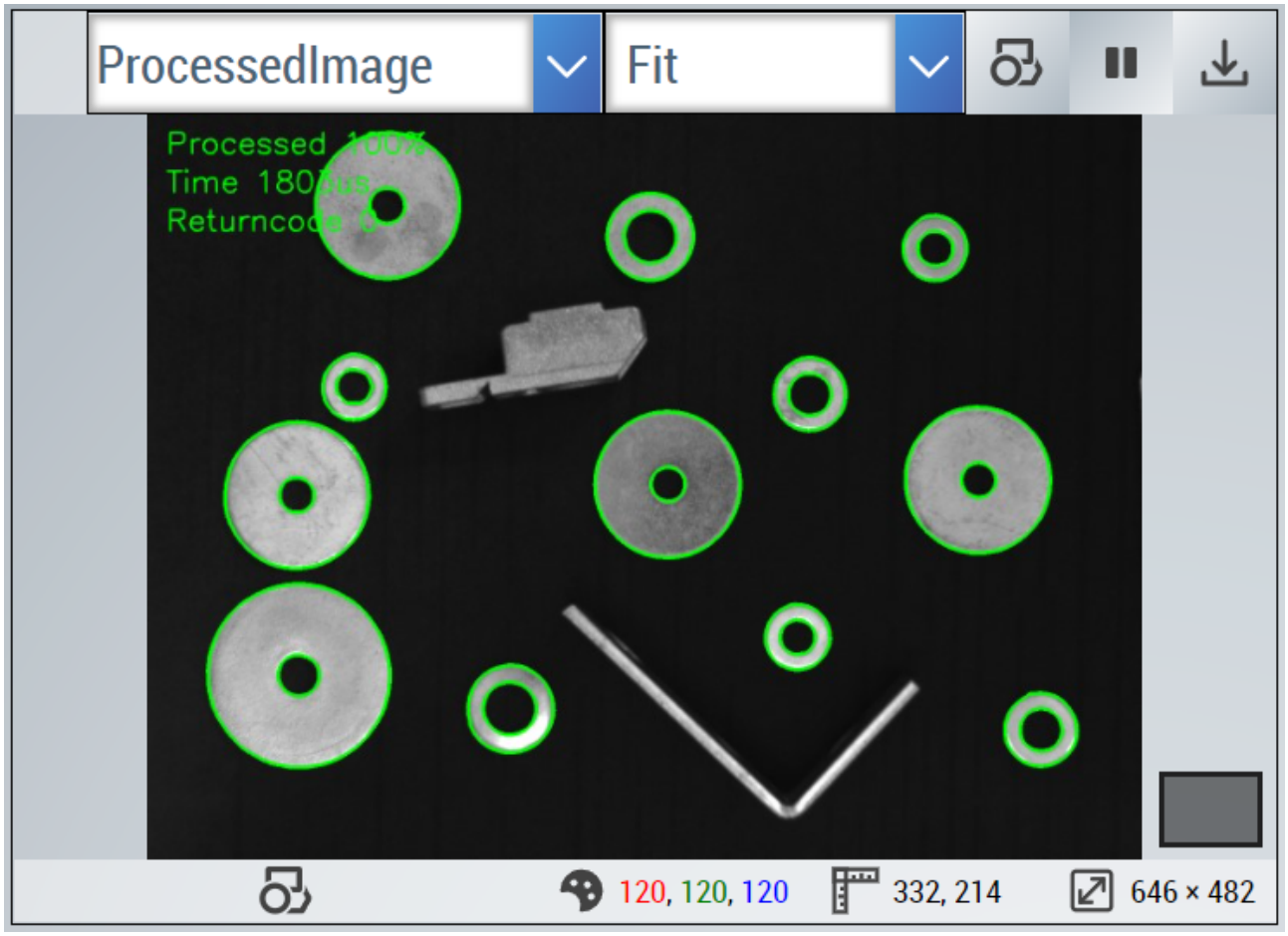
The Rectangle (Interactive) and Polygon (Interactive) controls are primarily auxiliary controls that are reused in the Image Watch control. If necessary, they can also be used alone. In addition, the controls can also be used for derived shapes such as a point, a line, a circle, an ellipse or a square with the help of their attributes.

In addition, the package contains the following help functions:

Function	Category	Description
BoxColorConverter [▶ 2747]	Vision	Converts the `ColorValue` attribute of the Color [▶ 2731] control for display in the color box and vice versa.
ConvertColor [▶ 2748]	Vision	Converts color values between different formats.
PixelColorFormatter [▶ 2749]	Formatting	Formats the `PixelColor` attribute of the Image Watch [▶ 2707] control to text for display in the toolbar [▶ 2721].
ShapeValueFormatter [▶ 2749]	Formatting	Formats the `ShapeValue` attribute of the Image Watch [▶ 2707] control to text for display in the toolbar [▶ 2721].
ToRotatedRectangle [▶ 2749]	Data Conversion	Converts a rectangle of type TcVnRectangle [▶ 2751] to type TcVnRotatedRectangle [▶ 2751].
ToUprightRectangle [▶ 2750]	Data Conversion	Converts a rectangle of type TcVnRotatedRectangle [▶ 2751] to type TcVnRectangle [▶ 2751].

7.2.1 Image Watch

The **Image Watch** Control offers many other tools and information displays in addition to the image display. An overview of the areas and components can be found in the subchapter Structure of the control [▶ 2715]. The description of each configuration option can be found under the following categories on this page. For each category there are sub-chapters with further information, details and application examples. In order to directly display TwinCAT Vision images from the PLC, the Vision extension [▶ 2704] for the HMI server is also required.



For all the following attributes, the control has a getter and a setter method. For example, the two methods `getBarColor(): SolidColor` and `setBarColor(value: SolidColor): void` exist for the attribute `BarColor`. The use of the attributes is explained in more detail in the linked subchapters.

All further attributes, events and permissions provided by the base class **Controls** can be found in the TwinCAT HMI documentation.

Category: Colors

Attribute	Type	Description
`BarColor`	`SolidColor`	Toolbar and Infobar color (see categories **Toolbar** and **Infobar**).
`ShapeStrokeColor`	`SolidColor`	Color of the contours of the shapes that can be drawn on the image (see category **Shapes**).
`ShapeHandleColor`	`SolidColor`	Color of the handles of the shapes that can be drawn on the picture (see category **Shapes**).

Category: Common

The following attributes specify the resource [▶ 2716] from which the images to be displayed are loaded.

Attribute	Type	Description
`Image`	`String`	Resource for the image to be displayed. Can be either a path to an image file or a data URI.
`ImageList`	`ImageList`	List of named image resources. If the list is not empty, the binding of `Image` will be replaced with the resource selected by `ImageIndex`.
`ImageIndex`	`Number`	Selects which of the image resources from the list will be displayed if `ImageList` is not empty.
`Alt`	`String`	Alternative text that will be displayed if the current image resource is not available.

Function	Description
`downloadImage(fileName: string): void`	Triggers a browser-specific download of the currently displayed image in its original size. Can alternatively be triggered via a UI element in the toolbar [▶ 2721].

Category: View

The following attributes and functions determine the section of the displayed image [▶ 2718] and how it can be changed.

Attributes	Type	Description
ImageFreeze	Boolean	If active, the currently displayed image is fixed and will not be updated until ImageFreeze is disabled.
Offset	[Number, Number]	Position of the upper left corner of the image in the coordinate system of the control.
Scale	Number (%)	Scaling of the image in percent based on ScaleReference.
ScaleReference	ScaleReference	At ImageSize, 100% corresponds to the original image size. At ControlWidth, 100% corresponds to the width of the control. At ControlHeight, 100% corresponds to the height of the control. At ControlSize, 100% corresponds to either the width or the height of the control, depending on which limits the size of the image first.
Draggable	Boolean	If active, the image can be moved using the mouse or touch. In the process, Offset is automatically adjusted.
Scalable	Boolean	If active, the image can be scaled using the mouse wheel or multitouch. In the process, Offset and Scale are automatically adjusted.
ScaleMin	Number	Minimum Scale when scaling via mouse wheel or multitouch.
ScaleMax	Number	Maximum Scale when scaling via mouse wheel or multitouch.
ScalingCenter	ScalingCenter	Point that remains fixed when scaled via the Scale attribute . At TopLeftCorner the upper left corner of the image remains fixed. At ImageCenter the center point of the image remains fixed. At ControlCenter the center of the control remains fixed. When scaling using the mouse wheel, the position of the cursor always remains fixed.
View (readonly)	Number[4]	Specifies the displayed section of the image in terms of a region of interest. The format is: [X, Y, Breite, Höhe].

Function	Description
`centerImageToControl(): void`	Adjusts the `Offset` so that the current image is in the centre of the control.
`fitImageToControl(center: boolean): void`	Adjusts `Offset` and `Scale` so that the current image is centered and filled in the control.
`setView(view: number[]): void`	Changes `Offset` and `Scale` so that the specified section of the image is displayed. Does not necessarily have to match View if the aspect ratio of the clipping does not match the control.

Category: Toolbar

The following attributes can be used to customize the toolbar [▶ 2721] of the control.

Attribute	Type	Description
`ToolbarVisible`	`Boolean`	Toolbar display.
`ToolbarItems`	`ToolbarItems`	Selection of elements to be displayed in the toolbar.
`ScaleOptions`	`Number[]`	Scaling values displayed and selectable in the `ScaleSelection` combo box in addition to the "Fit" option.
`ToolbarHeight`	`Number (px)`	Height of the toolbar.
`ImageSelectionWidth`	`Number (px)`	Width of the `ImageSelection` combo box.
`ScaleSelectionWidth`	`Number (px)`	Width of the `ScaleSelection` combo box.
`ScaleSelectionEditable`	`Boolean`	If active, values can be manually typed into the `ScaleSelection`. If inactive, selection is only possible from the offered `ScaleOptions`.

Category: Infobar

The following attributes can be used to customize the infobar [▶ 2722] of the control.

Attribute	Type	Description
`InfobarVisible`	`Boolean`	Display of the infobar.
`InfobarItems`	`InfobarItems`	Selection of elements to be displayed in the infobar.
`PixelColorFormat`	`Function`	Function that generates a text for the infobar from the value of the `PixelColor` attribute.
`ShapeValueFormat`	`Function`	Function that generates a text for the infobar from the value of the `ShapeValue` attribute.
`InfobarHeight`	`Number (px)`	Height of the infobar.

Category: Shapes

The following attributes help to draw shapes [▶ 2725] on the image.

Attribute	Type	Description
ShapeSelectionActive	Boolean	If active, a shape (based on ShapeType) can be drawn, displayed and edited on the image.
ShapeType	ShapeType	Shape type. Possible are: • Dotted • Line • Rectangle / square • Ellipse / circle • Polygon
ShapeIsRotatable	Boolean	If active, suitable shapes can be rotated using a handle in the UI. (line, rectangle, square, ellipse, polygon)
ShapeData	ShapeData	Geometric description of the shape. Can be used to query the data of a drawn shape or to display a specific predefined shape.
ShapeAutoClear	Boolean	If active, ShapeData is reset to the base value of the respective ShapeType when ShapeSelectionActive is deactivated.
ShapeStrokeThickness	Number (px)	Thickness of the lines of the displayed shape.
ShapeHandleSize	Number (px)	Size of the handles of the displayed shape.
ShapeClickableSize	Number (px)	Size of the clickable area for the handles of the displayed shape. Can be used to simplify editing the shape on touch panels without displaying the handles unnecessarily large.
ShapeAngleInterval	Number (°)	Step size in which the rotation of the shape can be changed. For example, with a value of 90, there are only four possible alignments. At 0 the rotation is arbitrary.
ShapeIsInImageArea (readonly)	Boolean	Indicates whether the displayed shape is completely within the image area or not. Can be useful to check the validity of ShapeData for certain PLC functions. Only the corner points are used for the calculation.
ShapeLimitToImage	Boolean	If active, the shape can only be drawn or edited within the image.
ShapeDataUpdate	DataUpdate	Specifies whether ShapeData should be updated during (onChanged) or at the end (onConfirmed) of a graphical interaction.

Function	Description
`clearShape(): void`	Deletes the displayed shape, also resetting `ShapeData` to the base value of the respective `ShapeType`.

Category: Thumbnail

The following attributes determine the display of a thumbnail. The thumbnail [▶ 2729] gives an overview of which section of the image is visible.

Attribute	Type	Description
`ThumbnailVisible`	`Boolean`	Display of the thumbnail.
`ThumbnailPosition`	`ThumbnailPosition`	Position of the thumbnail.
`ThumbnailSize`	`Number (px)`	Size of the thumbnail. Determines the width of the thumbnail; the height is automatically calculated based on the aspect ratio.

Category: Overlay

The following attributes allow objects to be overlaid [▶ 2729] on the image (e.g. as positioning or marking aids).

Attribute	Type	Description
`OverlayVisible`	`Boolean`	Display overlays.
`OverlayImage`	`String`	Image overlaid on the displayed image.
`OverlayElements`	`OverlayElements`	Geometric elements overlaid on the displayed image.

Category: Info

The following attributes can be used to obtain information [▶ 2731] about the image or specific pixels of the image.

Attribute	Type	Description
`PixelInfoUpdate`	`PixelInfoUpdate`	Determines when `PixelColor` and `PixelPosition` are updated. • At `Never` they are never updated. • At `AtClick`, they are updated when you click on a specific part of the image. • At `AtCursor` they are updated every time the cursor moves on the image. This always displays the values of the current cursor position.
`PixelColor` (readonly)	`[Number, Number, Number, Number]`	Color of the selected image pixel.
`PixelPosition` (readonly)	`[Number, Number]`	Position of the selected image pixel in image coordinates.
`ImageSize` (readonly)	`[Number, Number]`	Original size of the displayed image.
`ImageDisplaySize` (readonly)	`[Number, Number]`	Display size of the displayed image.

Events

All events are listed in the "Vision" category.

Event	Description
`.onPixelInfo`	This event is triggered if `PixelInfoUpdate <> Never` and the user clicks on a position in the image or moves the mouse pointer over it to select a pixel. Internally, the event updates the values of the attributes `PixelColor` and `PixelPosition`.
`.onImageReceived`	This event is triggered when the control receives an image.
`.onShapeChanged`	This event is triggered as soon as the attribute `ShapeData` changes.
`.onShapeConfirmed`	This event is triggered when a shape is set or an interaction with it is completed.

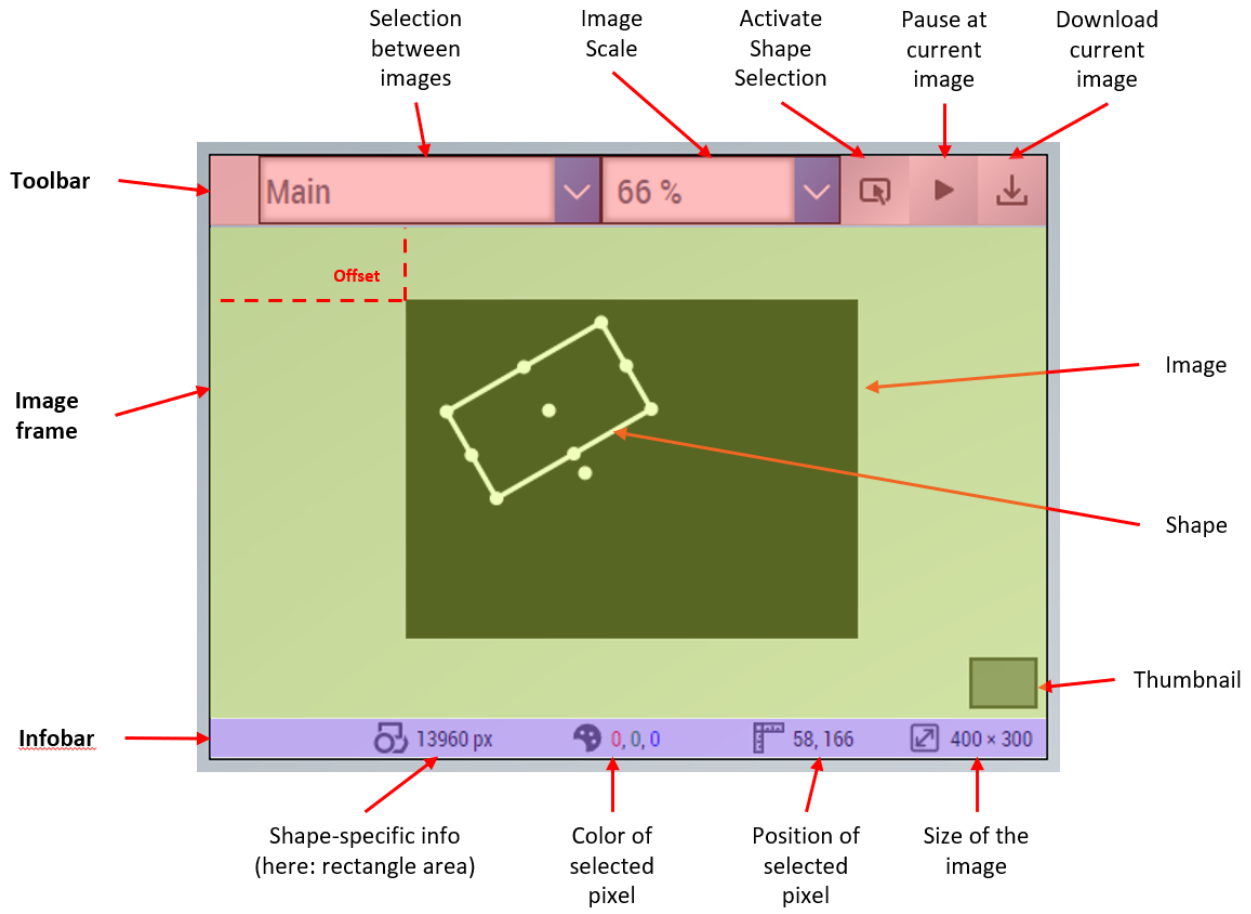
Themed Resources

The following attributes can be used to change the appearance of sub-controls (the combo boxes in the toolbar) at the Themes settings. A detailed description of the individual attributes can be found in the standard HMI documentation.

Attribute	Type
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 Structure of the control

The **Image Watch** Control consists of areas for displaying tools and information (Toolbar [▶ 2721] and Infobar [▶ 2722]), and a main area for displaying the image. Within this area, the image can be moved and scaled programmatically, by mouse or by touch. Other features like the thumbnail or the selection of shapes can be used application specific.



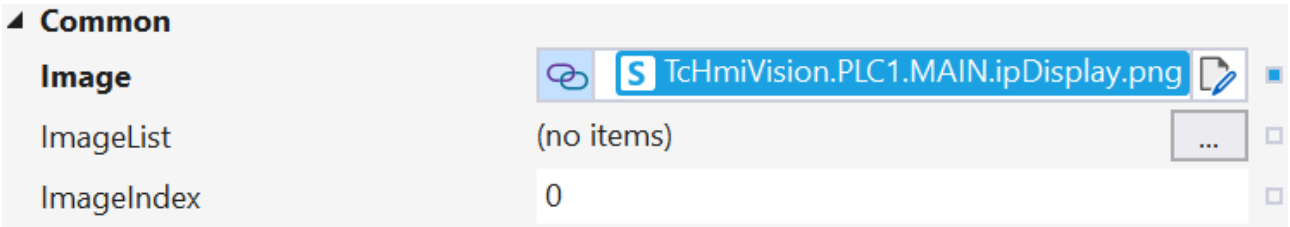
7.2.1.2 Image resources

The attributes and functions explained on this page can be found in the category **Common**.

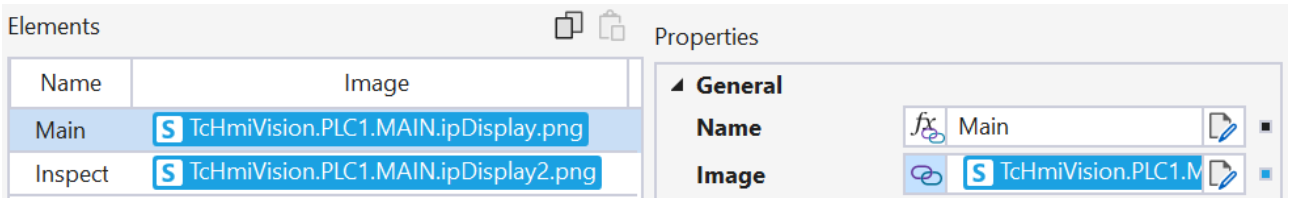
Image resources

The image resource defines which image will be displayed in the Image Viewer. The intended use is to link as an image resource a `ITcVnDisplayableImage` symbol using the HMI.Vision extension [▶ 2704]. In general, however, the image resource only needs to be a valid image URL. This can be either a direct path in the project, a URL to an external server or a Data URL. This works analogously to the Src attribute of the standard image control.

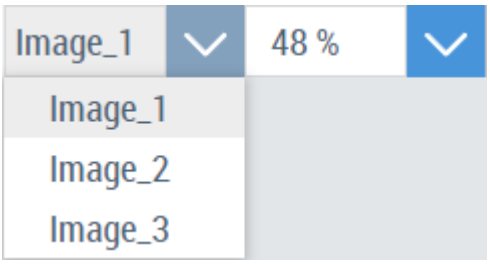
In the simplest case, the image resource is specified via the `Image` attribute:



Furthermore, it is possible to store multiple image resources in the `ImageList` attribute:



The `ImageIndex` attribute can be used to select which of the resources is actually used. This can be used, for example, to easily switch between image symbols from the PLC. The switch can also be made graphically via the `ImageSelection` combo box in the toolbar [▶ 2721]. For this purpose, not only the respective image resource but also an associated image name can be assigned in the `ImageList` .



Simultaneous use of Image and ImageList is mutually exclusive

Internally, using the `ImageList` property causes the respective image symbol of the selected `ImageList` element to be bound to the `Image` symbol. Therefore, the use of `Image` and `ImageList` are mutually exclusive; you should not bind an image symbol to the `Image` attribute if you are already using `ImageList`.

Receive event

Each time the control receives a new image, the event `onImageReceived` is triggered.

Alternative text

If no image resource is specified or no valid image could be loaded from the specified resource, an alternative text is displayed. This can be changed by the `Alt` attribute and is "No image available." by default.

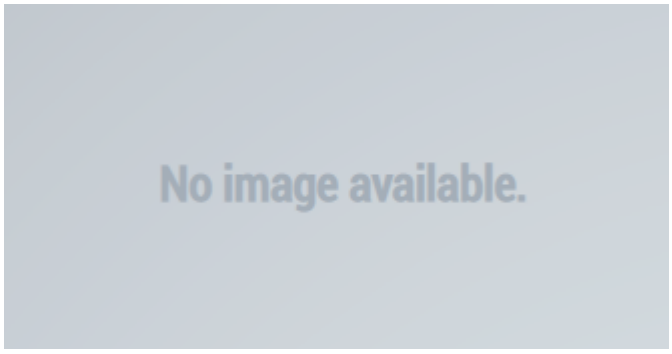


Image download

The currently displayed image can be saved locally on the client device via browser-specific download. The download can be triggered either by the `downloadImage` function or the download button in the toolbar [▶ 2721].

The image is saved in its original size.

The file name can be specified as a parameter in programmatic calls. A file extension does not need to be specified as it is automatically appended. If the parameter remains empty, the file name is also created automatically. For a symbol, the symbol name is used first; for a data URL, the control name is used accordingly:

<symbol name/control name>_<date>_<time>-<MS>.<file extension>

For an image path, the file name and file extension from the specified path are used without appending a timestamp.

The selection of the save path is not possible within the control, but corresponds to the download path setting of the used browser. The usual download path is **C:\User\<User>\Downloads**.

Requirements and restrictions of the image download

Since the image download is done via a browser-specific functionality, the possibilities, settings and functionality depend on the used web browser for the TwinCAT HMI Client.

The method `downloadImage` requires the authorization **operate** of the TwinCAT HMI user management.

This method does not work with images that are linked as a stream, as well as in the live view of HMI engineering.

7.2.1.3 Image view

The attributes and functions explained on this page can be found in the **View** category.

Internally, the size and position of the displayed image is precisely defined by the properties `Offset`, `Scale` and `ScaleReference`. However, the typical use case for changing size and position is using interactive features such as touch and mouse gestures. Thereby the above mentioned properties are automatically changed internally by those gestures.

Property: `Offset`

The `Offset` property specifies the image position relative to the control itself. The upper left corner is defined as the origin, with the X-axis to the right and the Y-axis down. The offset of the image is also measured from the upper left corner of the image. The offset coordinates are specified in pixels.

As mentioned earlier, the `Offset` property is changed by the control itself, for example, when the user interacts with the image using touch or mouse gestures.

By default, the offset property is associated with a `TwoWay` binding.

Properties: `Scale` **and** `ScaleReference`

The property `Scale` specifies a scaling factor in percent to calculate the pixel size in which the image is displayed on the screen. The size is always calculated in relation to the original image size or in relation to the size of the control, depending on the value of the attribute `ScaleReference`.

The `ScaleReference` can be one of the following values:

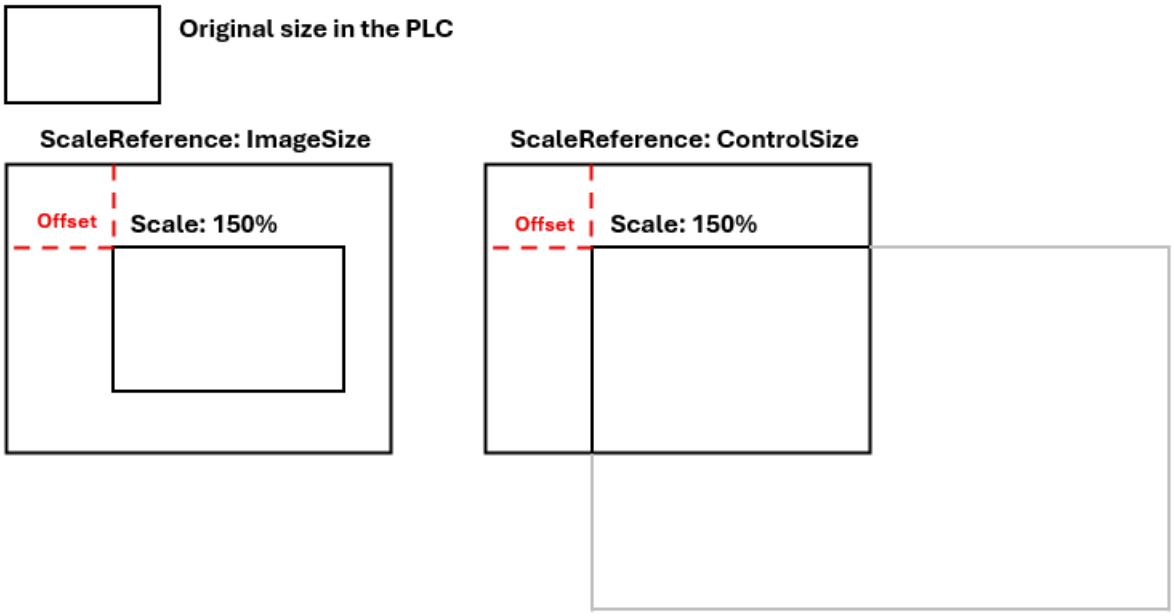
ImageSize	The size of the displayed image is equal to the original image size multiplied by the value of the property `Scale`. Example: the original size of the image is 300x200, the property `Scale` is 150%. The resulting size of the displayed image is 450x300.
ControlSize Width	The size of the displayed image is equal to the width of the control (or more precisely, the image frame within the control) multiplied by the value of the `Scale` property.
ControlSize Height	The size of the displayed image is equal to the height of the control (or more precisely, the image frame within the control) multiplied by the value of the `Scale` property.
ControlSize	The size of the displayed image is equal to the size of the control (or more precisely, the image frame within the control) multiplied by the value of the `Scale` property. If the aspect ratio of the control does not match the aspect ratio of the image, either the height or the width is used as the reference size, depending on which dimension differs more.

The advantage of the `ImageSize` setting is the direct relationship between the displayed size and the original size of the image. Two images with different sizes in the PLC always have different sizes in the HMI. The advantage of the `ControlSize` setting is the automatic adjustment of the image to the control. The images are always the same size, regardless of how large they actually are within the PLC.

Aspect ratio

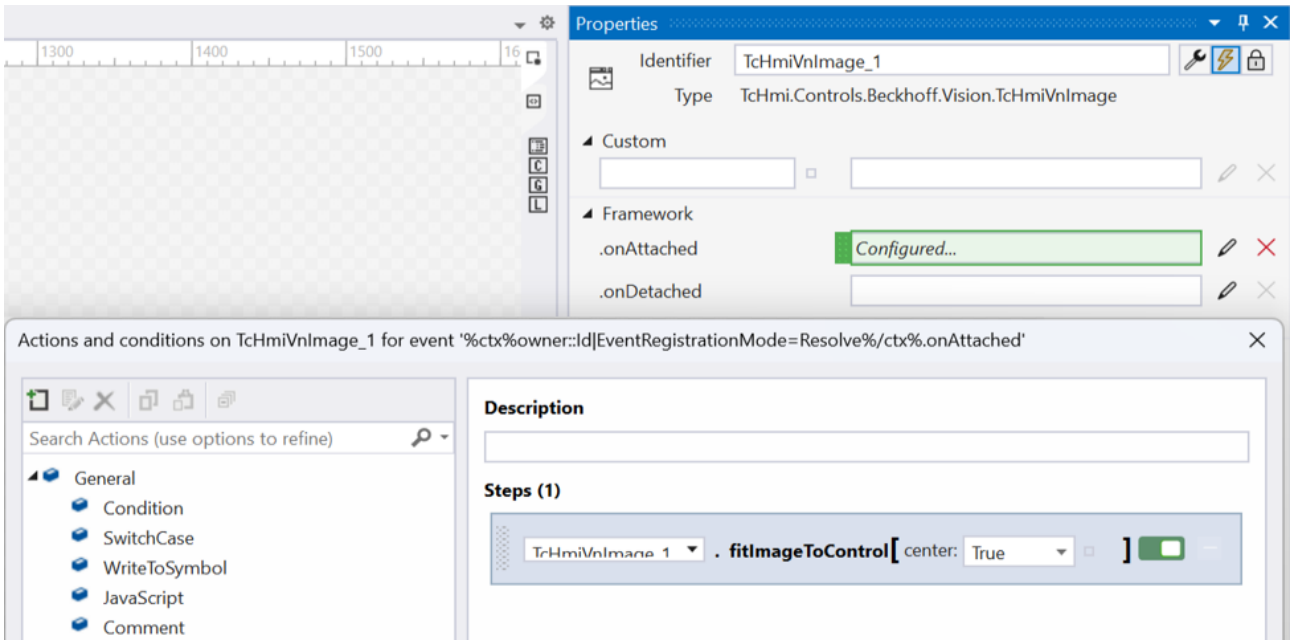
The height and width of the image are always calculated to maintain the aspect ratio of the original image.

Example: `Offset`, `Scale` **and** `ScaleReference`



Function: `FitImageToControl`

To maximize and center the currently displayed image within the Image Viewer, the FitImageToControl function can be called. If this is to be done from the beginning, the function should be linked to the .onAttached event. Once this function has been activated, all subsequent images will also be centered and maximized until the view is changed elsewhere, thereby canceling the mode.



Property: `ScalingCenter`

The `ScalingCenter` offers a possibility to adjust the scaling behavior. An individual change of the `Scale` properties always results in a change of the lower right corner of the image, while the upper left corner remains fixed. To change this and fix other points, the property `ScalingCenter` can be used. In this case, the property `Offset` is automatically adjusted. The following options are available:

ScaleCenter value	Description
TopLeftCorner	Upper left corner of the image stays centered when zooming.
ImageCenter	Center of the image remains centered when zooming.
ControlCenter	Center of the control remains centered when zooming.

Gestures

The user can change the size and position of the image by the following gestures/interactions:

- Hold the left mouse button anywhere in the image frame and drag the mouse to move the image and thus change the property `Offset`. The property `Scale` remains unaffected.
- Touch the image frame with one finger and move your finger to move the image to change the `Offset` property. The property `Scale` remains unaffected.
- Scroll with the mouse wheel to change the size of the image and thus the property `Scale`. The center of the scaling will always be the position of the mouse pointer. As a result, the `Offset` property may also change accordingly. The mouse pointer must always be positioned over the control.
- Touch the picture frame with two fingers and move both fingers in one direction, towards or away from each other, to change the position and size of the picture. The two properties `Offset` and `Scale` are adjusted.

To prevent these user interactions, the two properties `Draggable` and `Scalable` can be set to `false` to prevent a user-induced change of position or size. Changing the value (e.g. by writing to the symbol) is always possible and is not affected by the two properties.

Properties: `ScaleMin` and `ScaleMax`

The control positioning and scaling system applies limiting rules to prevent the user from creating confusing or disorienting views of the image.

The properties `ScaleMin` and `ScaleMax` represent limits on the setting of the `Scale` property for user interactions.

In addition, the control automatically restricts the image to its borders with a certain outer padding (currently 10px). Positions outside this limit are not useful as you cannot see the image.

Property: `View` **(read-only)**

The properties `Offset` and `Scale` are good for internal handling of the image view, but are not necessarily intuitive for the user (e.g. to define an image area). People often tend to think in terms of regions of interest or similar representations. Therefore, after changing the size or position of the image, the currently visible area of the image is calculated and provided via the read-only property `View`.

The internal calculation always sets the coordinate origin to the upper left corner of the image and calculates from there the upper left position as well as the width and height of the image frame.

The `View` property is provided as an array in the form `[X, Y, Breite, Höhe]`.

To set the position and size using a view description, see the next section on the method `SetView`.

Method: `SetView`

The `SetView` method allows the user to display a specific area of the image. When this method is called, the image area specified by a parameter is maximized and centered within the image frame.

The area must be specified in the original image coordinates as `[x, y, Breite, Höhe]`.

Differing views

Depending on the aspect ratios of the specified area and the image frame, the requested view may differ from the actual view. This in turn can be retrieved via the read-only property `View`.

Separate action

The view functionality is not accessible via a property, but via a read-only property and a setter method. The reason for this is the difference between an ordered view and the actual view mentioned above, and also that a **View** property would conflict with the properties `Offset` and `Scale` as to which view should be achieved. This is prevented by the view setter being a one-time action instead of a bound property.

Stop image flow

If the property `ImageFreeze` is true, the currently displayed image does not change even if new images are received. The intended use case is to freeze the current image for further review without adding additional logic within the PLC.

If the property `ImageFreeze` is changed from `true` to `false`, the control immediately displays the last image again and ignores the intermediate images.

The property `ImageFreeze` can be controlled by the Freeze button in the toolbar (see `FreezeButtonVisible` property in the category Toolbar [▶ 2721]).

7.2.1.4 Toolbar

The attributes and functions explained on this page can be found in the category **Toolbar**.

The controls in the toolbar can be used to interactively perform various actions that are otherwise only accessible via the attributes. Each element can be shown or hidden. In addition, the entire toolbar can be adjusted in height or completely hidden.

The positioning of the elements in the toolbar is fixed and cannot be customized. If such adjustments are nevertheless necessary, you can hide the corresponding elements using the property `ToolbarItems` and implement them yourself with your own controls and place them on the Image Control toolbar. The available toolbar elements from left to right are:



ImageSelection	Combo box for selecting images from the image list to be displayed (see property `ImageList` in the category **Common**). Only displayed if the image list contains at least one element. When switching between images, expect a short loading time.
ScaleSelection	Combo box for selecting specific scaling values to adjust the current image. The selectable scaling values can be adjusted via the property `ScaleOptions`. If `ScaleSelectionEditable` is deactivated while a user-defined scale is set, the combo box will no longer display a value. This is the default behavior of the combo box control.
ShapeButton	Button for setting the value of the `ScaleSelectionActive` property. If active, an existing shape is displayed and can be edited. If no shape is displayed, a new one can be drawn.
FreezeButton	Button for setting the value of the `ImageFreeze` property. If active, the currently displayed image is fixed and will not be updated.
DownloadButton	Button for downloading the current image with default name (see function `downloadImage` under Image download [▶ 2716]). To assign a different file name, this button can be implemented by the customer and then call the function with a self-defined name.

Functions also programmatically accessible

Any action that is accessible through the interface elements of the toolbar is also accessible programmatically through the properties of the control.

Size settings

The toolbar offers customization options for its own height via the `ToolbarHeight` property and for the width of the two combo boxes properties `ImageSelectionWidth` and `ScaleSelectionWidth`. The width of the three additional buttons is automatically adjusted to the height of the toolbar so that they are square.

The width of the dropdown button of the combo boxes cannot be adjusted, because this functionality is not available in the standard combo box control.

Internally, the property `ToolbarHeight` is limited to a maximum of 150 pixels.

Background color

The background color of the toolbar is a semi-transparent white `"rgba(255, 255, 255, 127)"` by default and can be customized using the `BarColor` property in the category **Colors**.

Advanced settings

Further graphical settings (e.g. font of the combo boxes) can be changed via Themed Resources. Basically, attributes of the combo boxes are provided. Themed Resources can be customized in the theme editor at `TcHmiVnImage > Themed Resources`. These settings will then apply to all instances of the Image Viewer control.

7.2.1.5 Infobar

The attributes and functions explained on this page can be found in the category **Infobar**.

The display elements in the infobar can be used to display various information about shape values, pixel values or the image size. Each element can be shown or hidden. In addition, the entire infobar can be adjusted in height or completely hidden.

The positioning of the elements in the infobar is fixed and cannot be customized. If such adjustments are nevertheless necessary, you can hide the corresponding elements using the **InfobarItems** property and implement them yourself with your own controls and place them on the Image Control infobar. The available infobar elements from left to right are:



ShapeValue	Calculated value to the currently displayed shape. This value is calculated by the function bound to the `ShapeValueFormat` property. The standard formatting function [▶ 2749] `ShapeValueFormatter` returns a specific value depending on the selected shape type: • Coordinate from the point • Length of the line • Area of the square, rectangle, circle or ellipse • Number of points of the polygon This example [▶ 2723] describes the implementation and use of a custom formatter function.
PixelColor	Color value(s) of the selected pixel. This value is calculated by the function bound to the `PixelColorFormat` property. The standard formatting function [▶ 2749] `PixelColorFormatter` returns a specific value depending on the selected color space. Selectable color formats are: `Gray`, `RGB`, `RGBa`, `HSV`, `HSV FULL`, `HLS`, `HLS FULL`, `Lab`, `YUV`, `YCrCb`. You can also set whether and how rounding should take place. You can choose between 0-3 `Decimals` and `Unrounded`. The PiyelType can also be set. Possible values here are `8-Bit (USINT)` and `16-Bit (UINT)`. This example [▶ 2724] describes the implementation and use of a custom formatter function.
PixelPosition	Position of the selected pixel.
ImageSize	Original size of the currently displayed image.

Size settings

The infobar offers customization options for its own height via the property `InfobarHeight`, which is internally limited to a maximum of 150 pixels.

Background color

The background color of the **infobar** is by default a semi-transparent white "`rgba(255, 255, 255, 127)`" and can be customized using the `BarColor` property in the category **Colors**.

7.2.1.5.1 ShapeValueFormatting function

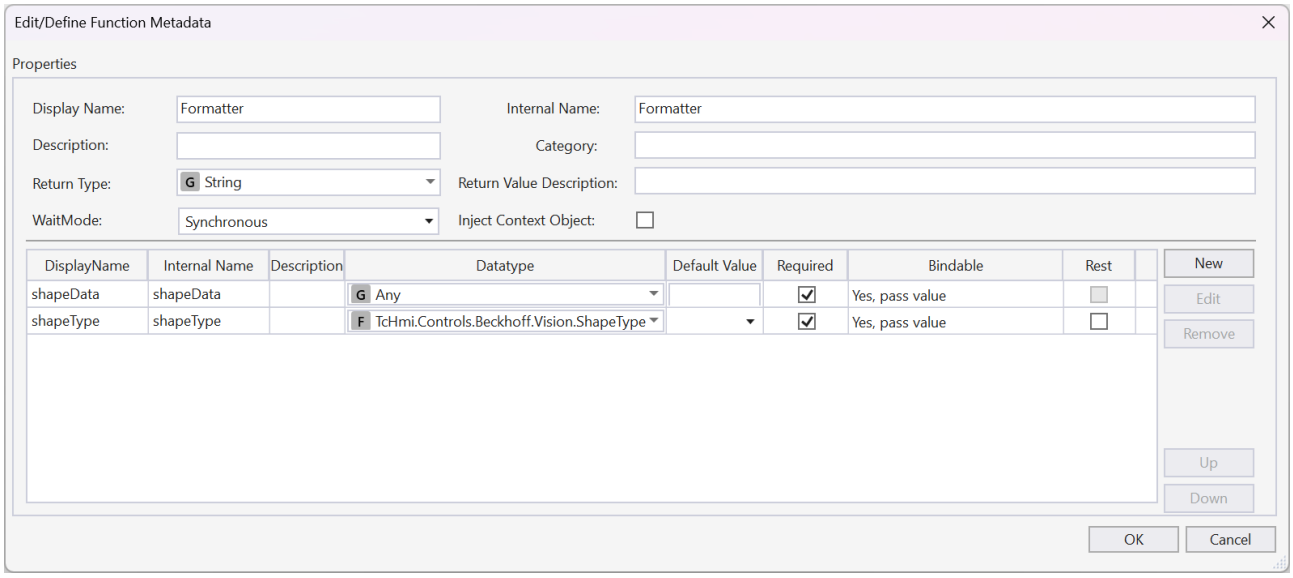
As an alternative to the `ShapeValueFormatter` function, a custom formatter function can be created to display other shape information in the infobar. The following example shows how such a function is created and which interfaces it must have.



The general use of functions is explained in the HMI documentation.

After creating the TypeScript function, two parameters must be created. The first one gets the Shape description passed in the same way as the ShapeData property, as it can be retrieved from the control. This parameter must be created as type `Any`. The second parameter is passed the shape type analogous to the ShapeType property. This parameter must be specified as type `TcHmi.Controls.Beckhoff.Vision.ShapeType`. It should be noted that these passed parameters may differ from the values of the corresponding control properties at the time of function execution for timing reasons.

The function must return a value of type `String`, which is then displayed in the infobar. The function is called whenever the shape description changes and thus the string in the infobar needs to be updated.

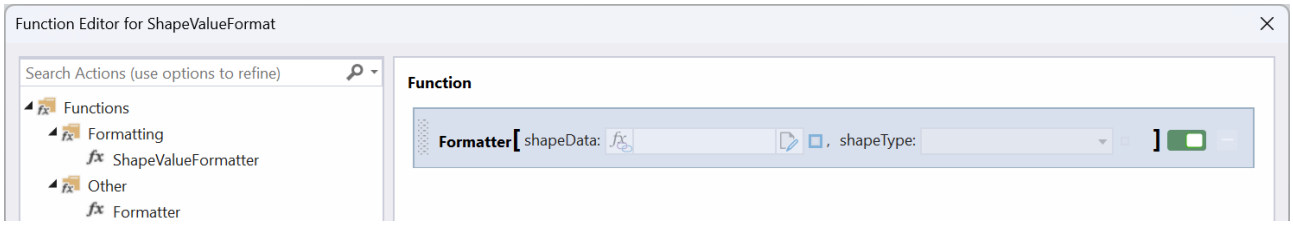


In the following code sample, the width and height are returned if the objects are included in the shape data:

```
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);
```

To use a formatter function you have created, click on the Select button at the `ShapeValueFormat` property and select the function you have just created in the Function Editor.



7.2.1.5.2 PixelColorFormatting function

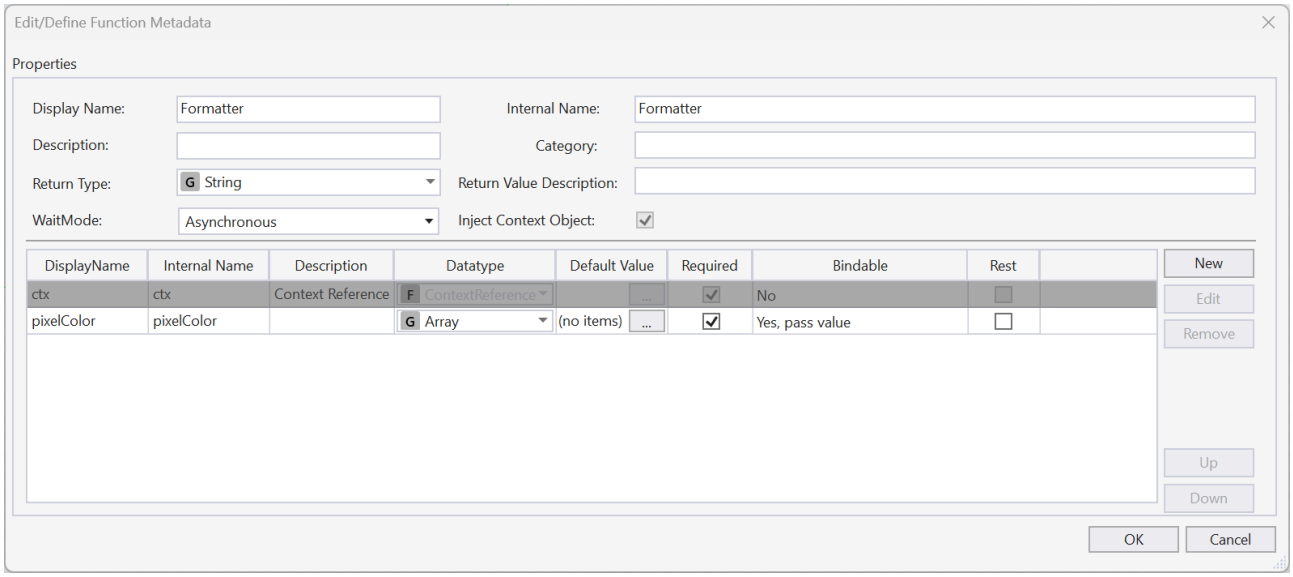
As an alternative to the `PixelColorFormatter` function, a user-defined formatter function can be created to format the color of the selected pixel differently. The function gets the color value of the pixel and returns a string that is displayed in the infobar. The following example shows how such a function is created and which interfaces it must have.



The general use of functions is explained in the HMI documentation.

After creating the TypeScript function, the WaitMode must be changed to `Asynchronous`. As a result, the parameter "ctx" is created, which is required for the return. A further parameter of the type “Array” must then be created, which receives the color value as an array in the same way as the PixelColor property of the control. It should be noted that the parameter passed may differ from the value of the corresponding control properties at the time of function execution for timing reasons.

The function must return a value of type `String` so that it can be displayed in the infobar. The call is always made when the value of the `PixelColor` property changes and the string in the infobar therefore needs to be updated.

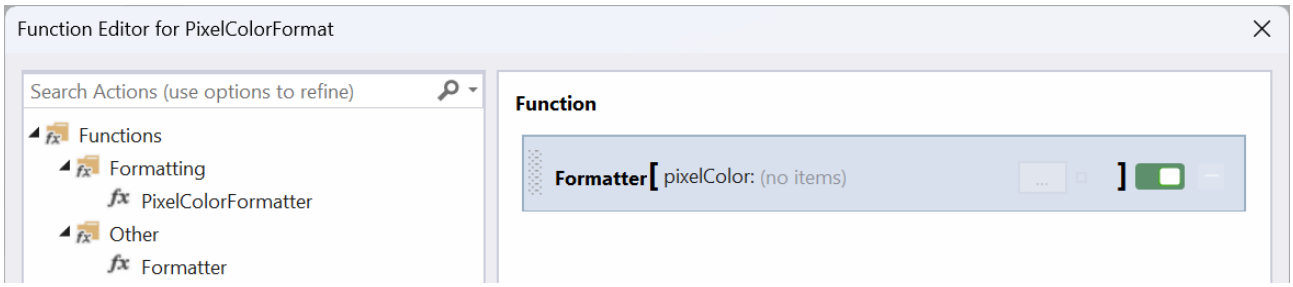


In the following code block, 3 color values in red, green and blue are returned as an example, provided at least 3 elements are present in pixelColor. The values are multiplied by 256 to convert them into 16 bits:

```
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);
```

To use a formatter function you have created, click on the Select button at the `PixelColorFormat` property and select the function you have just created in the Function Editor.



7.2.1.6 Shapes

The attributes and functions explained on this page can be found in the category **Shapes**.

A common use case is the selection of certain image areas by the user. This can be a region of interest or the position of an object. Another related use case is user-defined measurement of distances or areas within an image.

To accommodate all these use cases, the properties in the **Shapes** category offer flexible functionality for drawing and editing different shapes on the image and retrieving their respective geometric descriptions.

Property: ShapeSelectionActive

The selection of a shape can be activated via the property `ShapeSelectionActive` or via the **Shape** button in the Toolbar [▶ 2721]. As long as this property is active, the shape can be drawn by clicking on the image surface and edited by dragging and dropping the corner points or the shape itself. If this property is set to `false` and the `ShapeAutoClear` property is active, the current shape will be reset and lost, so any data needed must be extracted first.

The geometric description of the drawn shape can be retrieved via the property `ShapeData`. Depending on the property `ShapeType` (see below) the shape data have a different format. You can expect exactly the same format used in the interactive controls themselves (for example, `TcHmiVnPolygon` and `TcHmiVnRectangle`), because the symbol is bound directly to the shape control used internally. The property `ShapeLimitToImage` can be used to restrict the shape to the image area. If this is not the case (default), please note that the shapes can also assume negative values. This can be queried with the function `shapeIsInImageArea` .

Properties: ShapeType and ShapeData

Different types of shapes are available, based on two different data types. You can select one of the following shapes using the `ShapeType` property:

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`

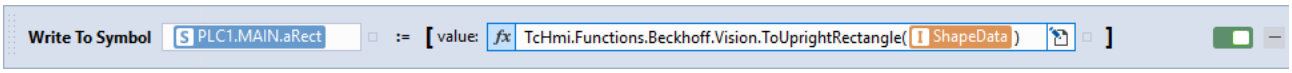
Square,
Rectangle,
Circle,
Ellipse
	{
	fAngle: angle,
	aCenter: [cx, cy],
	stSize: {
	fWidth: width,
	fHeight: height,
	},
	}
		`TcVnRotatedRectangle`

Polygon	[[x1, y1], [x2, y2], …, …, [xn, yn]]	`ARRAY[0..n] OF TcVnPoint2_REAL or ARRAY[0..n] OF TcVnPoint2_LREAL`

The values in `ShapeData` refer to the image coordinates of the original image.

Since the shape types square and circle always have the same height and width by definition, they will not be drawn exactly to this endpoint when the second position is clicked if the aspect ratio is not the same. In this case, the desired position can be reached by moving the corresponding corner.

The conversion function ToUprightRectangle [▶ 2750] is available to transfer the data of the shape types Square and Rectangle of type TcVnRotatedRectangle [▶ 2751] directly to a PLC variable of type TcVnRectangle_UDINT [▶ 245] or a TcVnRectangle [▶ 2751] and use it as an upright bounding rectangle without a rotation angle. For the opposite case there is the ToRotatedRectangle [▶ 2749] function.



Programmatically, the `ShapeData` property can be changed via Java script as follows:

```
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}})
```

In the first sample, a polygon with an array of 4 points is written. To write a point you need only the first element and for a line the first two. In the second sample, a rectangle is written. Since the shape types square, circle and ellipse are based on the same data type, the same notation can be used for these types as well.

The data can also be written to `ShapeData` via WriteToSymbol action. To do this, copy the part inside the round brackets of the `setShapeData()` function and paste it as an object value.

Property: `ShapeAutoClear`

If the `ShapeAutoClear` property is active and the `ShapeType` is changed while `ShapeSelectionActive` is `true`, the current shape is reset and a new shape of the selected shape type can be drawn. If `ShapeAutoClear` is not active and you switch between shape types of the same base control, the data is

retained when the ShapeType is switched and the new shape is displayed directly. The point, line and polygon types belong to one group that use the same base control, while the square, rectangle, circle and ellipse types belong to the other.

Appearance

The appearance of the shape and the control points can be changed by adjusting the color and size/ thickness of the contour:

- With ShapeStrokeThickness you can set the thickness of the lines in pixels.
- With ShapeHandleSize you can set the size of the displayed control points of the shape in pixels
- With ShapeClickableSize the size of the clickable area of a control point can be set in pixels. This can simplify operation on touch panels without having to display the handles unnecessarily large.
- Via ShapeStrokeColor and ShapeHandleColor the color of the lines and that of the control points can be set independently.

Property: ShapeIsRotatable

If ShapeIsRotatable is active, an additional handle is displayed for suitable shapes (line, rectangle, square, ellipse, polygon), with which the shape can be rotated. Using the ShapeAngleInterval property, the step size in which the rotation of the shape can be changed can be set in degrees. With the default value 0°, the shape can be rotated as desired. For example, if the shape may only be placed in 4 positions, you would enter 90° as the value.

Property: ShapeLimitToImage and ShapeIsInImageArea

It is often necessary for a shape to lie completely within the image, e.g. to be able to set a region of interest. If the property ShapeLimitToImage is active, a shape can only be drawn and moved within the image. This restriction does not affect the programmatic setting of a shape, but only graphical interactions. It should also be noted that if ShapeIsRotatable and ShapeAngleInterval are set, the shape can still be rotated without restriction.

If ShapeLimitToImage is not active, the positioning of the shape can be queried via the property ShapeIsInImageArea. This only returns true if valid ShapeData is currently available and if all corner points of the shape lie within the image area. The corner points are also used to check the circle and ellipse shapes. In order to be able to distinguish between a shape that is not present and one that is not in the image, you should also query ShapeSelectionActive and valid ShapeData.

Property: ShapeDataUpdate

The property ShapeDataUpdate can be used to specify when the property ShapeData should be updated. If onChanged is selected, ShapeData is continuously updated during a graphical interaction. If, on the other hand, onConfirmed is chosen, ShapeData is only updated at the end of an interaction.

Properties: ShapePolygonNumPoints and ShapePolygonClosed

If the property ShapeType is set to Polygon, the number of points from which a polygon is considered complete can be defined via the property ShapePolygonNumPoints. For example, if three points are specified, the polygon is automatically completed when the third point is drawn. If the default value 0 is used, the number of points is arbitrary. The polygon is then only completed when you click on the first point. The property ShapePolygonClosed can also be used to specify whether the first and last points of the polygon should be connected, i.e. whether the polygon should be displayed closed or not.

Method: clearShape()

The drawn shape can be deleted programmatically by calling the method clearShape.

Display element

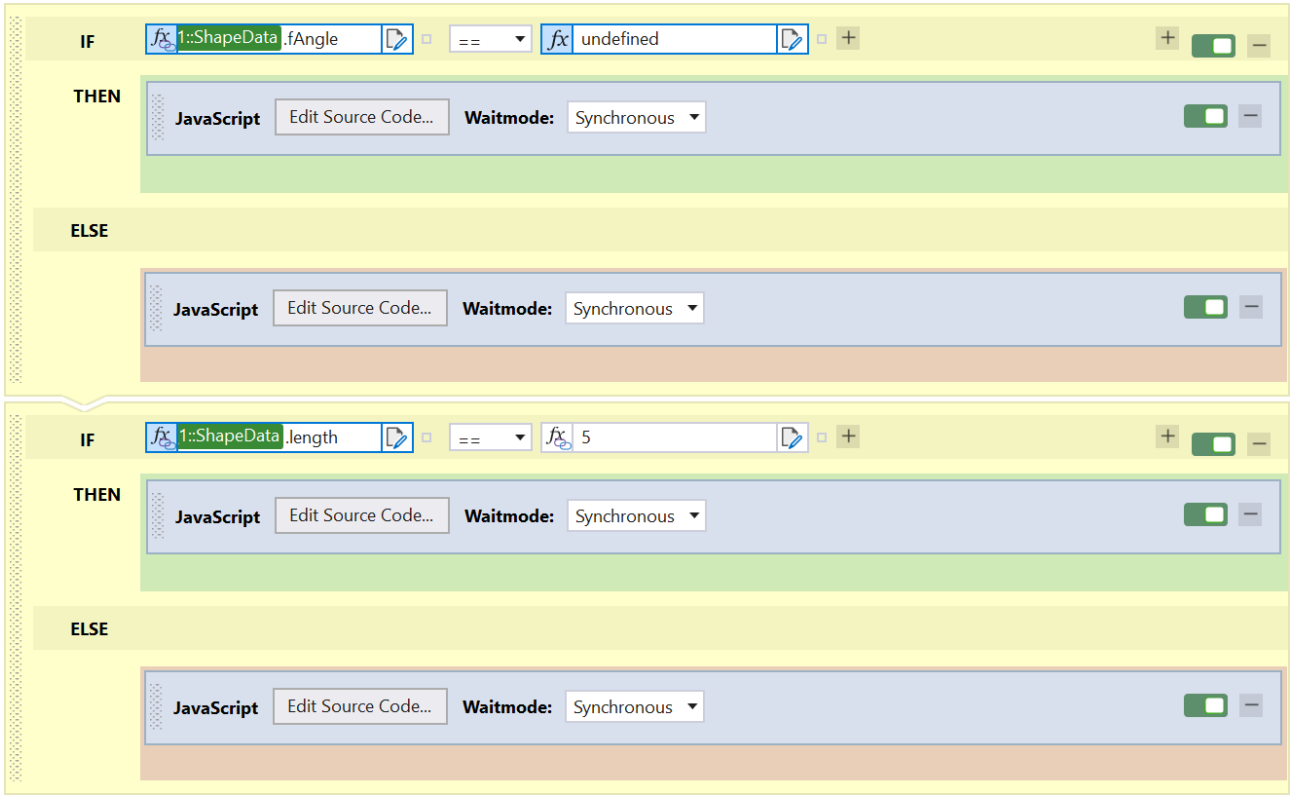
To support the use case of measuring distances or areas within the image, the Infobar [▶ 2722] provides an element for displaying text that represents the currently drawn shape. By default, the coordinate is displayed for the "Point" shape type, the line length for "Line", the number of points for "Polygon" and the area for the "Square", "Rectangle", "Circle" and "Ellipse" shape types. If further or other information is to be displayed, this can be implemented using a separate ShapeValueFormatting function [▶ 2723].

Query the shape types and data values

Depending on the selected shape type, the data in `ShapeData` can have different formats. To determine which format is present, the current `ShapeType` can be queried or it can be determined from the data values, e.g. whether the fAngle property is defined. If this is the case, a TcVnRotatedRectangle type is present, otherwise an array of points. To query how many points are contained in the array, the length can be queried. The number can then be used to determine whether there is a point, a line or a polygon with what number of points.

Furthermore, it is possible to check whether valid data are available by querying the individual values. When deleting or resetting the shape data, the individual values are written with 0, which represents an invalid value.

These queries can be used to ensure that the various shape types are each written to a designated PLC variable of the same type and that only valid data can be transferred to the control program.



Events: onShapeChanged and onShapeConfirmed

The two events `onShapeChanged` and `onShapeConfirmed` are available to inform you of a change in shape.

The event `onShapeChanged` is triggered as soon as `ShapeData` changes. A change can occur when `ShapeData` is set programmatically, when the `clearShape()` method is called and during graphical interactions. This event is used to track the change of `ShapeData` live, provided that `ShapeDataUpdate` is set to the value `onChanged`.

The event `onShapeConfirmed` is triggered when the shape is set programmatically or after a graphic interaction with the shape has been completed. This event is triggered on the condition that `ShapeData` is valid.

Advanced use of shapes

The Shape functionality is implemented via separate controls that are integrated into the Image Watch control. These shape controls have some properties that are not directly available from the ImageWatch control. This includes, for example, the exact aspect ratio of a rectangle. These properties can be accessed either programmatically or by using the shape controls separately.

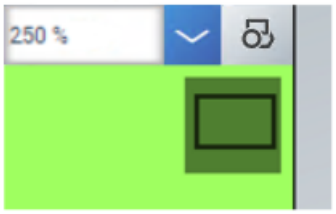
7.2.1.7 Thumbnail

The attributes and functions explained on this page can be found in the category **Thumbnail**.

The thumbnail indicates which section of the image is visible in the control. It is closely related to the **View** of the image (see read-only property `View` in the **View** category), but differs in some respects.

Whether the thumbnail is visible is determined by the `ThumbnailVisbile` property. Its general position and size is determined by the properties `ThumbnailPosition` and `ThumbnailSize` and is always square. If the aspect ratio of the image is not 1:1, the thumbnail size is the full width of the image, while the height is adjusted to the aspect ratio.

When the entire image frame is covered with the image, the thumbnail display is virtually identical to the read-only property `View`. Otherwise, it differs in that areas outside the image are included in the `View` but not in the thumbnail.



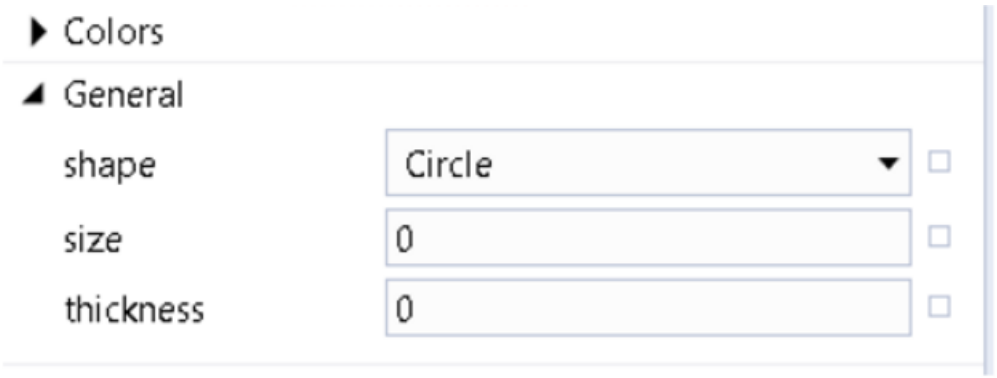
7.2.1.8 Overlays

The attributes and functions explained on this page can be found in the category **Overlay**.

Overlays can be used to help set up the camera or position objects within the image, for example.

Predefined geometric shapes

The control provides predefined geometric objects (such as circles, rectangles, crosses, frames) that can be displayed in the center of the image. These shapes can serve, for example, as crosshairs or for frames and marks at certain positions in the image. Several objects can also be displayed at the same time (e.g. a circle and a crosshair cursor). This is done using the array property `OverlayElements`. The configuration of an object looks like this:



The objects are always centered in the image. They have a fixed scale defined by **size** and do not scale with the image.

Overlay image

If this functionality is not flexible enough for a particular application, you can specify an overlay image using the `OverlaySource` property. This is displayed in exactly the same place and size as the main image, i.e. the overlay image is stretched/squeezed to fit the main image. We therefore recommend that the overlay image has at least the same aspect ratio and preferably the same size as the main image and is transparent except for relevant parts.

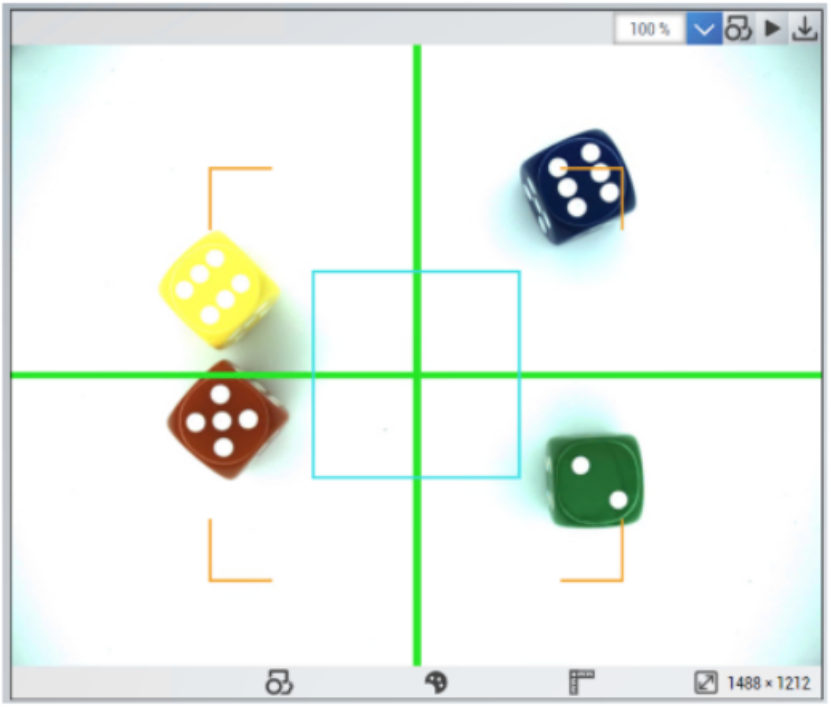
As an example, you could draw a user-defined crosshair cursor with additional accessories on a transparent image and use it as an overlay. To create a transparent image, a 4-channel image must be created with the color value 0 at the fourth channel of the background color. The cursor lines to be drawn in then get the color value 255 at the fourth channel, so that these elements overlay the background image.

View

For example, an overlay image with a rectangle and a text might produce the following result:



In the following example, an overlay image from the PLC (green lines) was combined with two objects defined in the control:



Please note that the PNG or BMP image format must be used when transmitting transparent images with the "TwinCAT.HMI.Vision" server extension, as JPEG and Stream do not support transparent image content with an alpha channel.

7.2.1.9 Information

The attributes and functions explained on this page can be found in the category **Info**.

You can use the following attributes to query information about the displayed image or specific pixels in that image.

ImageSize	Original size of the displayed image. This corresponds to the image size in the PLC.
ImageDisplaySize	Displayed size of the image in pixels.
PixelColor	Color of a specific selected pixel (see below for details). The color is provided as an array of four elements representing the color channels [R, G, B, A].
PixelPosition	Position of a specific selected pixel (see below for details). The position is specified as an array with two elements [X, Y] in original image coordinates, not in display coordinates.

The **PixelColor** and **PixelPosition** are calculated for a specific pixel. The selection of this pixel is done by a click, a touch or by the position of the cursor on the displayed image.

The property **PixelInfoUpdate** determines whether the calculation is performed at all or by which event it is triggered.

Never	**PixelColor** and **PixelPosition** are not updated at all.
AtClick	**PixelColor** and **PixelPosition** are updated when the user clicks or touches a specific pixel. The values are reset when the user clicks or touches a point on the image frame outside the current image.
AtCursor	**PixelColor** and **PixelPosition** are continuously updated as the user moves the mouse pointer over the image. The values are reset when the mouse pointer is not on the image.

Each time the pixel color and position are updated, the event **onPixelInfo** is triggered.

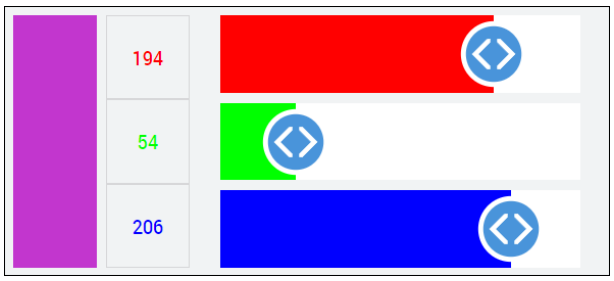
Display

The current PixelColor, PixelPosition and ImageSize information can be displayed in the infobar (see property `InfobarItems` in the **Infobar** category).



7.2.2 Color

The Color Control displays colors and makes them adjustable if necessary. It contains various display elements to represent the color values, which, depending on the configuration, can also be used to set the color values. An overview of the components and the structure can be found in the subchapter Display elements [▶ 2733]. The description of each configuration option can be found under the following categories on this page. For each category there are sub-chapters with further information, details and application examples.



For all the following attributes, the control has a getter and a setter method. For example, the two methods `getOrientation` and `setOrientation` exist for the attribute `Orientation`. The use of the attributes is explained in more detail in the linked subchapters.

All further attributes, events and permissions provided by the base class **Controls** can be found in the TwinCAT HMI documentation.

Category: Common

The following attributes represent a data interface for the [▶ 2735] color value of the control. It is used on the one hand to set the displayed color of the control, and on the other hand to retrieve a color value entered by the user.

Attribute	Type	Description
`ColorSymbol`	ColorSymbol [▶ 2750]	HMI symbol containing the current color value.
`ColorChannels`	ColorChannels [▶ 2750]	Defines the format of the color. This includes: • Number of channels • Min/max values of the channels • Integer or Float Display colors of the channels
`BoxColorConversion`	`Function`	Function that converts the respective color value into a string that can be interpreted by the browser and vice versa. This is necessary to tell the control the meaning of each color channel if it differs from the default RGB format.
`ColorValue` (readonly)	ColorValue [▶ 2751]	Color value. This can be retrieved without a symbol being linked as `ColorSymbol`.

The attribute `ColorValue` always adapts to the data type of the mapped PLC variable. The default is a 4-channel array (`TcVnVector4_LREAL`) which is also used without mapping. Other possible PLC data types are `INT, REAL, TcVnVector3_LREAL`.

Function	Description
`setColorValue(colorValue:` ColorValue [▶ 2751]`): void`	Sets the color value. Can be called without a symbol being linked as `ColorSymbol`.

Category: Usage

The following attributes specify the behavior of the control.

Attribute	Type	Description
BoxUsage	UsageMode [▶ 2751]	Determines whether the color box is displayed and whether it is editable.
ValuesUsage	UsageMode [▶ 2751]	Determines whether the individual color values are displayed and whether they are editable.
SliderUsage	UsageMode [▶ 2751]	Determines whether the individual color values are displayed as sliders and whether they are editable.
SliderValueUpdate	DataUpdate	Specifies whether the color symbol should be updated during onChanged or at the end of onConfirmed of an interaction with the sliders.

Category: Appearance

The following attributes determine the appearance [▶ 2733] of the control.

Attribute	Type	Description
Orientation	Orientation [▶ 2751]	Orientation of the control; analogous to the orientation of the LinearGauge controls. Horizontal or Vertical.
BoxSize	Number (px)	Determines the width or height of the color box depending on the orientation (value of Orientation).
ValuesSize	Number (px)	Determines the width or height of the displayed color values depending on the orientation (value of Orientation).

Themed Resources

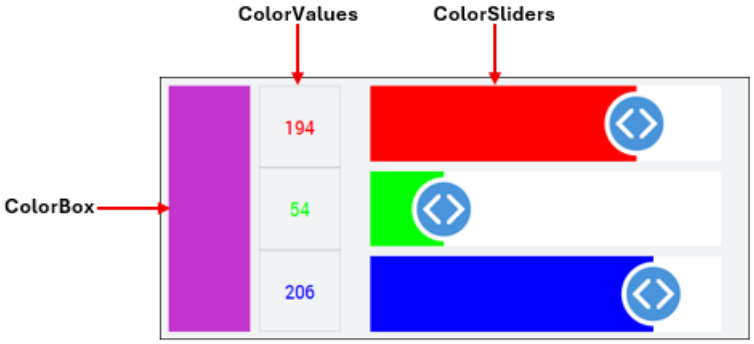
The following attributes can be used to change the appearance of sub-controls (the text boxes for the color values) at the Themes settings. A detailed description of the individual attributes can be found in the standard HMI documentation.

Attribute	Type
TextFontFamily	FontFamily
TextFontSize	MeasurementValue
TextFontSizeUnit	MeasurementValue
TextFontStyle	FontStyle
TextFontWeight	FontWeight
TextPadding	Padding

7.2.2.1 Display elements

Elements

The following elements can be displayed in the color control. The corresponding usage attributes (BoxUsage, ValuesUsage und SliderUsage) determine in each case whether the element is hidden (Hide) or displayed (ViewOnly) and whether a user can change the color with the help of this element (Editable).



Name	Position	Description
ColorBox	Left or top	This is a rectangular area where the current color is displayed. If the `BoxUsage` property is set to "Editable", clicking on the `ColorBox` opens a browser-specific dialog box for selecting the color. Depending on the browser, this may look like a separate window or an overlay, or it may show nothing. Since each browser controls the appearance and functionality of the dialog itself, we have no influence on it. For example, the color pipette takes a screenshot of the screen, so it doesn't work with changing images. Further information can be found in chapter Color boxes in different browsers [▶ 2734].
ColorValues	Center	This area includes text fields for viewing and editing the color values stored in the control. The number of value elements and their text color are taken from the property **ColorChannels**.
ColorSliders	Right or bottom	This area includes linear sliders corresponding to **ColorValues**.

Arrangement

The arrangement of the display elements depends on the `Orientation` attribute. "Horizontal" places the elements next to each other and "Vertical" places them on top of each other. Depending on the orientation, the attributes `BoxSize` and `ValuesSize` define the width or height of the respective elements. The width or height of the slider automatically corresponds to the rest of the available space.

NOTICE

Small sizes

If the control is very small in the selected orientation, it may not be possible to display all elements completely and correctly. This mainly affects the sliders. Above a certain size, a warning is displayed in the JavaScript console that the size may affect the function of the control.

Number

The number of displayed color values and sliders always corresponds to the number of channels in the set color format [▶ 2738]. The color format is set by means of the `ColorChannels` attribute.

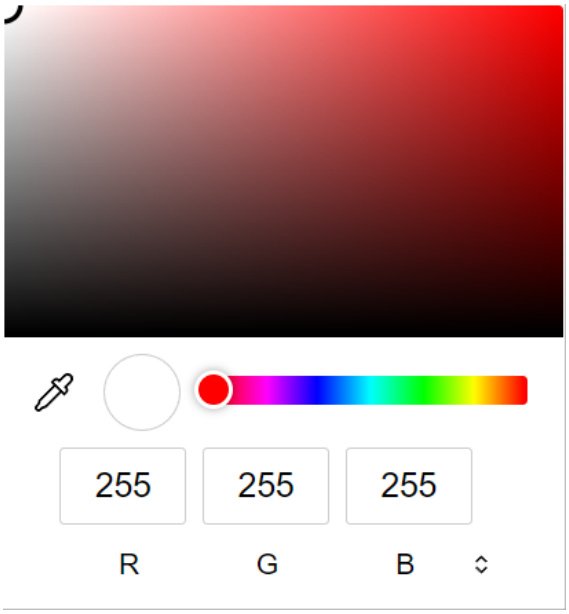
Text

The text boxes for the color values can be customized via various font properties using Themed Resources in the theme editor.

7.2.2.1.1 Color boxes in different browsers

If the `BoxUsage` attribute is set to `Editable`, another selection dialog for the color can be opened by clicking on the color box. This dialog is browser dependent. Following are some dialogs of the most used browsers:

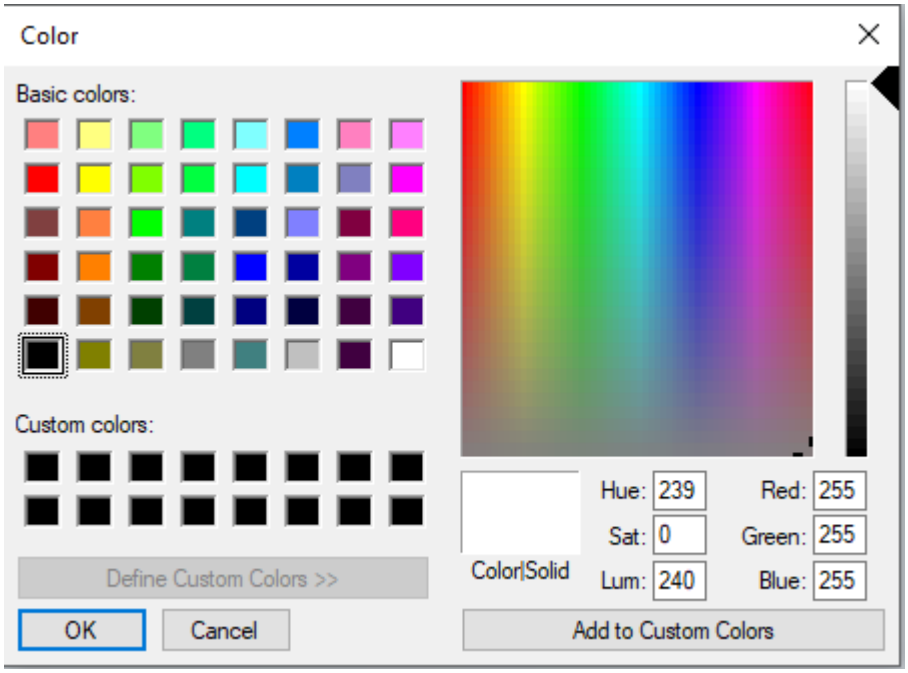
Google Chrome and Microsoft Edge



Pipette

The pipette function in Chromium-based browsers takes a screenshot of the browser window when the tool is selected. The selected color then comes from this screenshot. This means that when images change, the selected color will not be the one displayed at the time of the click. Therefore, it is recommended to stop the scrolling by means of the `ImageFreeze` attribute before using the pipette function.

Mozilla Firefox



7.2.2.2 Color data

The `ColorSymbol` attribute can be bound to different symbols. To avoid inconsistencies between the output color value and the data type of the symbol, the Color Control reads the type of the symbol and tries to automatically derive the correct return type.

For example, if three color channels are to be displayed, but an array of type `TcVnVector4_LREAL` is linked in the PLC, these types do not match. The Color Control automatically recognizes this and writes an array with four values instead of three into the symbol, so that no error occurs. The rest of the channel is left empty.

The read-only attribute `ColorValue` is used to make the color programmatically accessible without binding a symbol to the property `ColorSymbol`. Other than that, there is no difference. It is possible to use both versions at the same time. Please note that `ColorValue` also outputs the data type of the symbol that is bound to `ColorSymbol`.

The attribute `SliderValueUpdate` is used to specify whether the color symbol should be updated during `onChanged` or at the end of `onConfirmed` of an interaction with the sliders. If the color value is changed during a slider movement (`onChanged`) and a symbol is bound to the attribute `ColorSymbol`, it can happen that the value jumps back to an older value after the slider is released. If this behavior is not desired, `SliderValueUpdate` should be set to `onConfirmed` .

In analogy to the property `ColorValue`, the method `SetColorValue` provides a way to set the color values of this control without linking a symbol with the property `ColorSymbol`.

The two ways of accessing the color value are explained in the following samples:

- Linking with PLC [▶ 2736]
- Programmatic use [▶ 2737]

The example Conversion to HMI color object [▶ 2737] shows how the color data can also be used for other use cases.

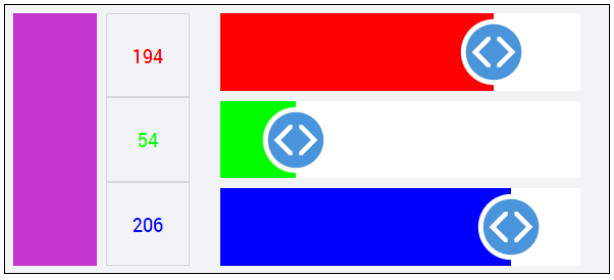
7.2.2.2.1 Linking with PLC

In the simplest case, the Color Control can be used to represent a color variable from the PLC. For this purpose, for example, a PLC variable of type `TcVnVector4_LREAL` is linked to the `ColorSymbol` attribute. Other permitted data types are `INT`, `REAL`, `TcVnVector3_LREAL`.



Now PLC variable and the control are synchronized with each other:

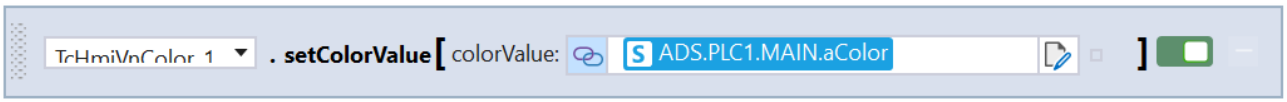
Expression	Type	Value
aColor	TcVnVector4_LREAL	
aColor[0]	LREAL	194
aColor[1]	LREAL	54
aColor[2]	LREAL	206
aColor[3]	LREAL	0



The fourth element of the PLC array is ignored, because by default only three channels are displayed as RGB in the control. This can be set via the color format [▶ 2738].

7.2.2.2.2 Programmatic use

If the color value of the Color Control is not to be linked directly to a symbol, it can be accessed programmatically instead. To set a color in the control, use the `setColorValue` function, either via JavaScript code or via the **Actions and conditions** dialog:




```
TcHmi.Controls.get("Background").setColorValue([0, 0, 255])
```

To query the color of the control, use the ReadOnly attribute `ColorValue`:



In JavaScript, the attribute must be retrieved using the `getColorValue` method:

```
const color = TcHmi.Controls.get("Background").getColorValue()
```

7.2.2.2.3 Conversion to HMI color object

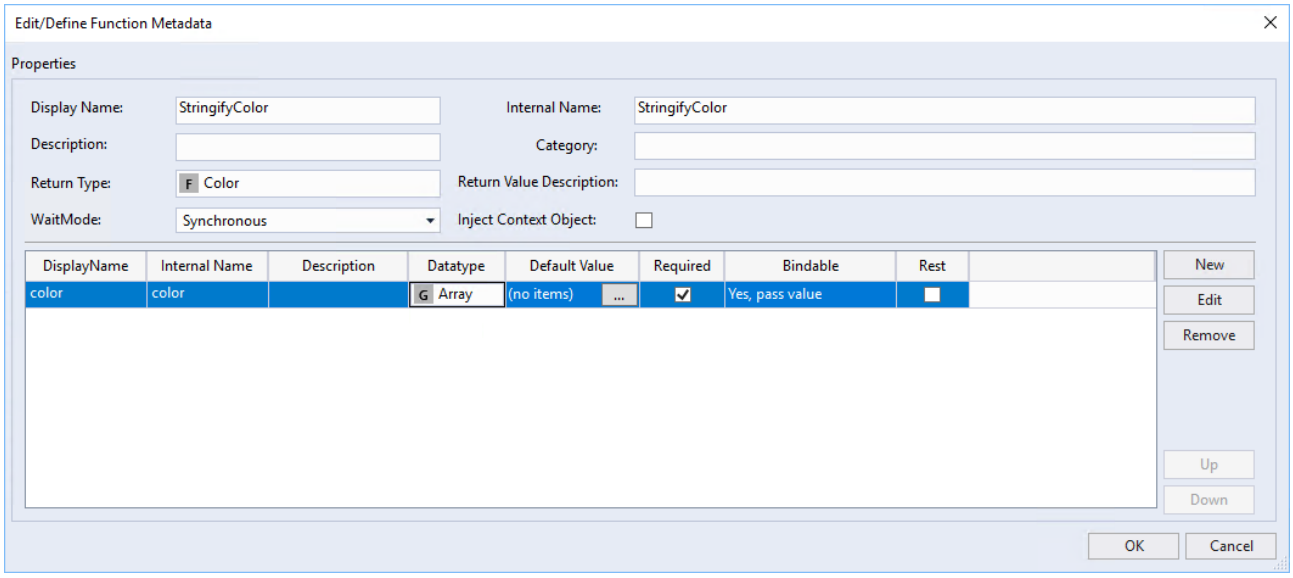
In order to use a color in the TwinCAT HMI, e.g. as a background or for other properties such as text or frames, it must be converted into an object of type TcHmi.Color. The following example shows how such a function is created and which interfaces it must have.



The general use of functions is explained in the HMI documentation.

After creating the TypeScript function, a parameter must be created. This gets the color value e.g. from the PixelColor property of the ImageControl or the ColorValue from the Color Control and must therefore be created of the type `Array`. It should be noted that the parameter passed may differ from the value of the corresponding control properties at the time of function execution for timing reasons.

The function must return a value of type `Color`, which can then be used for other properties in the HMI.



The following code sample shows the conversion of an RGB color value:

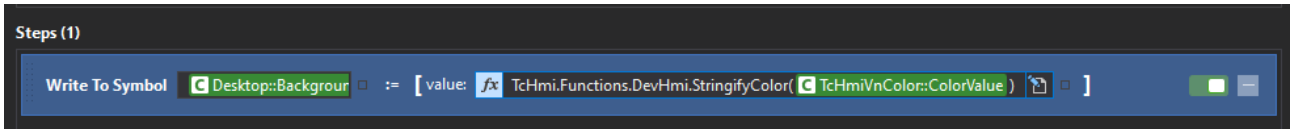
```
module TcHmi {
    export module Functions {
        export module Custom {
            export function StringifyColor(color: number[]): TcHmi.Color {
                if (color.length >= 3) {
                    return { color: `rgb(${color[0]}, ${color[1]}, ${color[2]})` };
                }
```

```
                throw new Error("Could not stringify color.");
            }
        }
        registerFunctionEx('StringifyColor', 'TcHmi.Functions.DevHmi', DevHmi.StringifyColor);
    }
}
```

To be accepted as a color object, the return value must be of type `Color` and must be structured as follows:

```
{
    color: 'rgb(<red>, <green>, <blue>)'
}
```

Such a function can be used, for example, by converting the color value of a TcHmiVnColor control with the function and then setting it as the background color of another UI element.



7.2.2.3 Color format

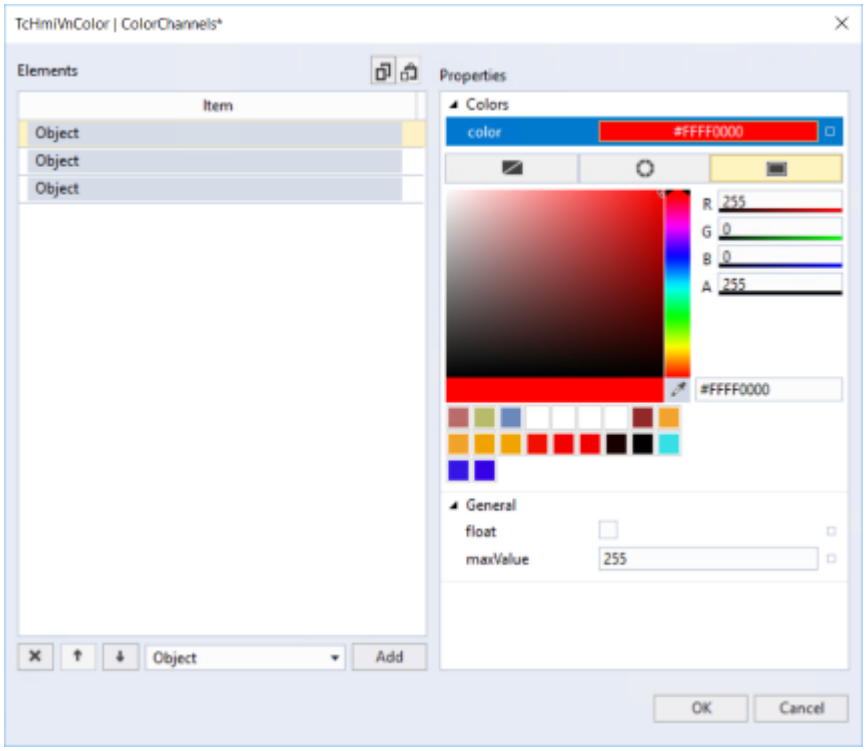
There are many different ways to represent a color. The most common formats such as Mono (1 channel grayscale value), RGB color space (3 channels color value), HSV / HLS color space (3 channels color value) are described in more detail below. However, the control offers many configuration options so that special color formats can also be set and used.

Setting the color format

To adapt the control to any color format, the general conditions can be set in detail:

- Number of channels (objects)
- Maximum values of each channel
- Data type (float or integer) of each channel
- Display color of each channel. This is used as the color of the respective slider and the displayed color value.

The adjustments can be made via the dialog of the attribute `ColorChannels`. The RGB format is preset as default. For example, to set a Mono 8 format, two of the three existing channels (Object) must first be deleted. After that, the color representation can be adjusted to an average grayscale value like 128 in the R, G and B channels.



Configuration examples are available for the following color formats:

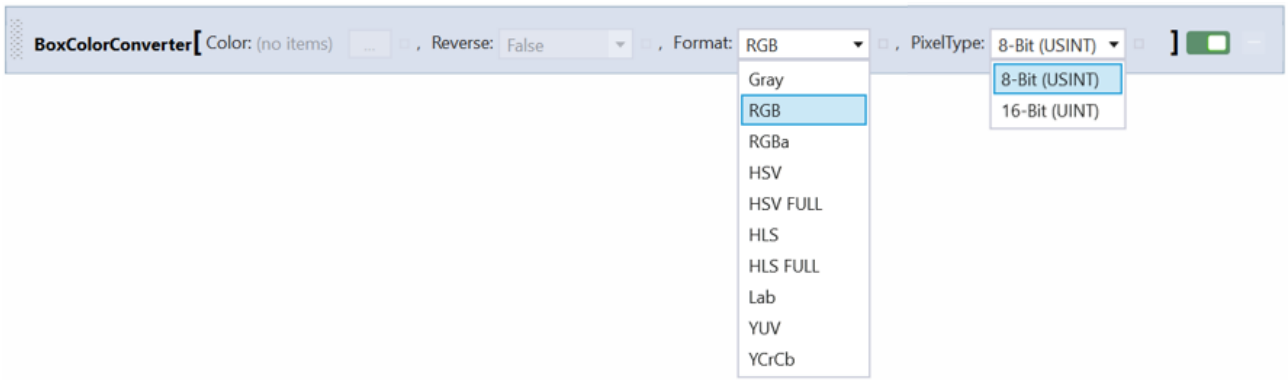
- RGB (standard) [▶ 2740]
- HSV / HLS (value adjustments) [▶ 2741]
- Percent RGB (custom format) [▶ 2742]

Color box conversion

This section is only relevant if the color box is used and the color does not correspond to the standard RGB format.

For the display of the color value by means of numbers and sliders no deeper knowledge about the color format is needed, because it is only the display of numerical values. However, in order to display the color in the color box, it must be defined how the individual color channels and their values are to be interpreted. For example, RGB values must be displayed differently than HSV values.

Since the browser-specific color input elements process RGB colors by default, all other color formats must first be converted to RGB to enable display in the color box. This conversion is to be set via the attribute `BoxColorConversion`. Conversion functions already exist for the formats Gray, RGBa, HSV, HSV FULL, HLS, HLS FULL, Luv, YUV and YCrCb, as well as for 8- and 16-bit data types.



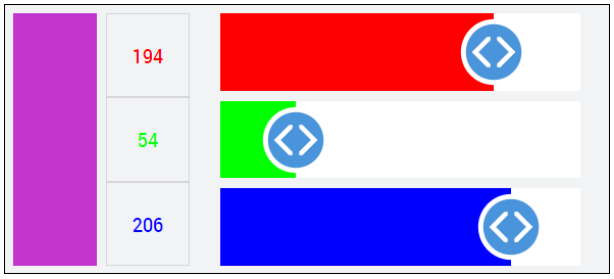
General color converter

In addition to this special conversion function for the color box, a general color converter with the same set of color formats for general use in the HMI is also available as ConvertColor [▶ 2748].

For completely different color formats or color formats with different scaling, user-defined functions [▶ 2742] can be created.

7.2.2.3.1 Standard RGB(a) colour format

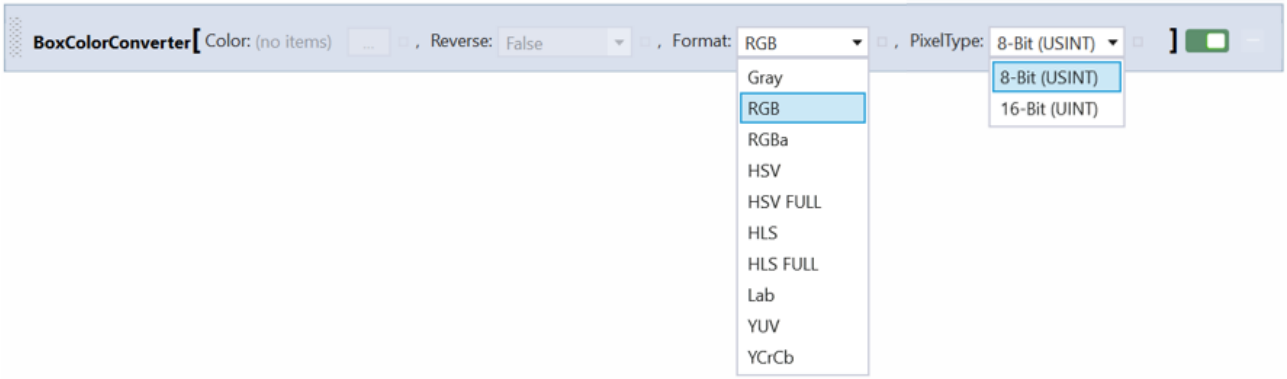
By default, three channels are defined, all of which are configured as integers with a maximum value of 255 and whose display colors correspond to the RGB colour format.



The values of the `ColorChannels` attribute can be written by script as follows:

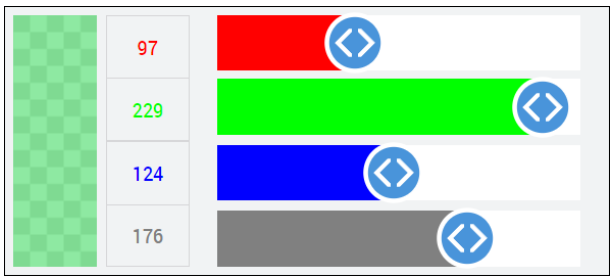
```
[
  {
    "color": {
      "color": "rgba(255, 0, 0, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(0, 255, 0, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(0, 0, 255, 1)"
    },
    "float": false,
    "maxValue": 255.0
  }
]
```

The `BoxColorConversion` attribute is set to the supplied `BoxColorConverter` function and this is configured with the format “RGB” and the pixel type “8-bit (USINT)”.



RGBa color format

To be able to display transparent colors, an additional color channel must be added in the attribute `ColorChannels`. It is advisable to configure this with the display color "rgba(128, 128, 128, 1)". As the `BoxColorConverter` already has a conversion to RGBa, this must then be selected as the color format.

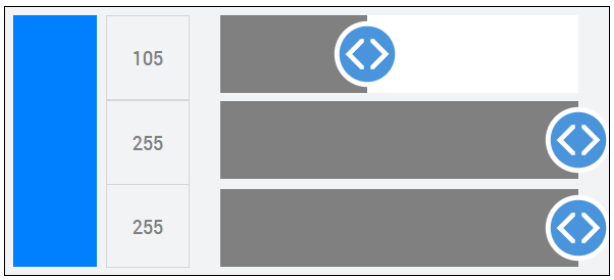


7.2.2.3.2 HSV/ HLS (FULL) color format

If a color is to be displayed in HSV format with the Color Control, some adjustments are necessary. The value range of the "Hue" channel is only defined from 0 to 180, therefore the `maxValue` of the first channel must be reduced to 180. The other two channels are scaled from 0 to 255. Since the individual channels of the HSV format cannot be assigned to specific colors, the colors for displaying the sliders and numerical values, for example, should be set to a uniform grayscale value so that no confusion arises.

The same settings and value ranges can be used for the HLS color space, since only the calculation of the L and S channels differs.

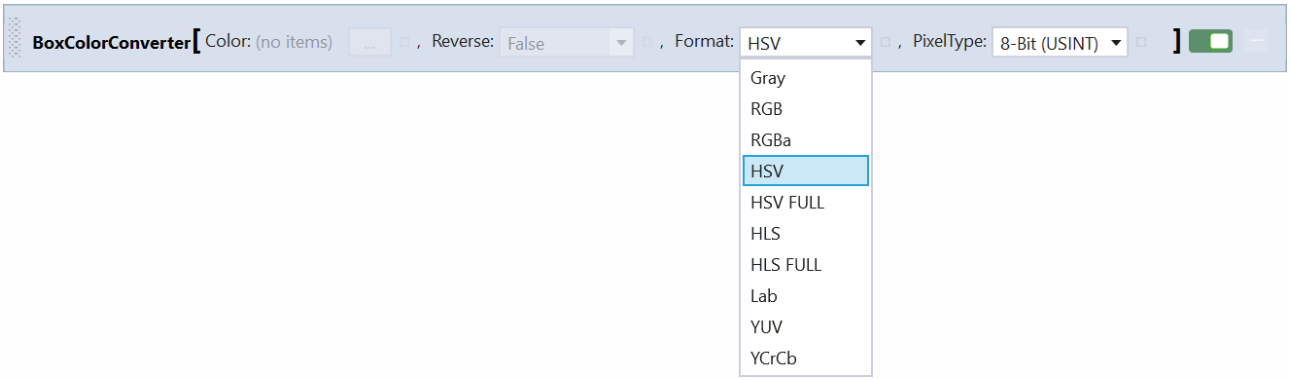
When using the "HSV FULL" and "HLS FULL" color formats, all color channels are defined from 0 to 255.



The values of the `ColorChannels` attribute can be written by script as follows:

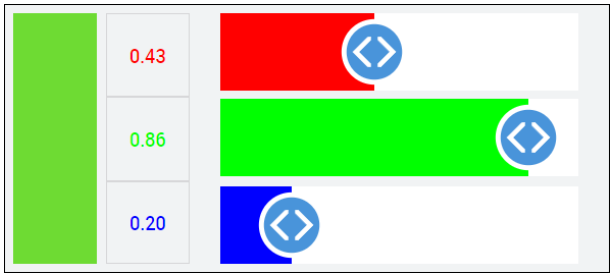
```
[
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 180.0
  },
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 255.0
  }
]
```

Finally, the conversion for the color box in the `BoxColorConversion` attribute must be adjusted. To do this, the corresponding color format must be selected in the `BoxColorConverter` function:



7.2.2.3.3 User-defined color format

As another example of the Color Control configuration, an RGB color format with a value range between 0 and 1 is considered. To display this in the Color Control, the channels must be set in the `ColorChannel` attribute as `float` and with a `maxValue` of `1`.



The values of the `ColorChannels` attribute can be written by script as follows:

```
[
  {
    "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 function

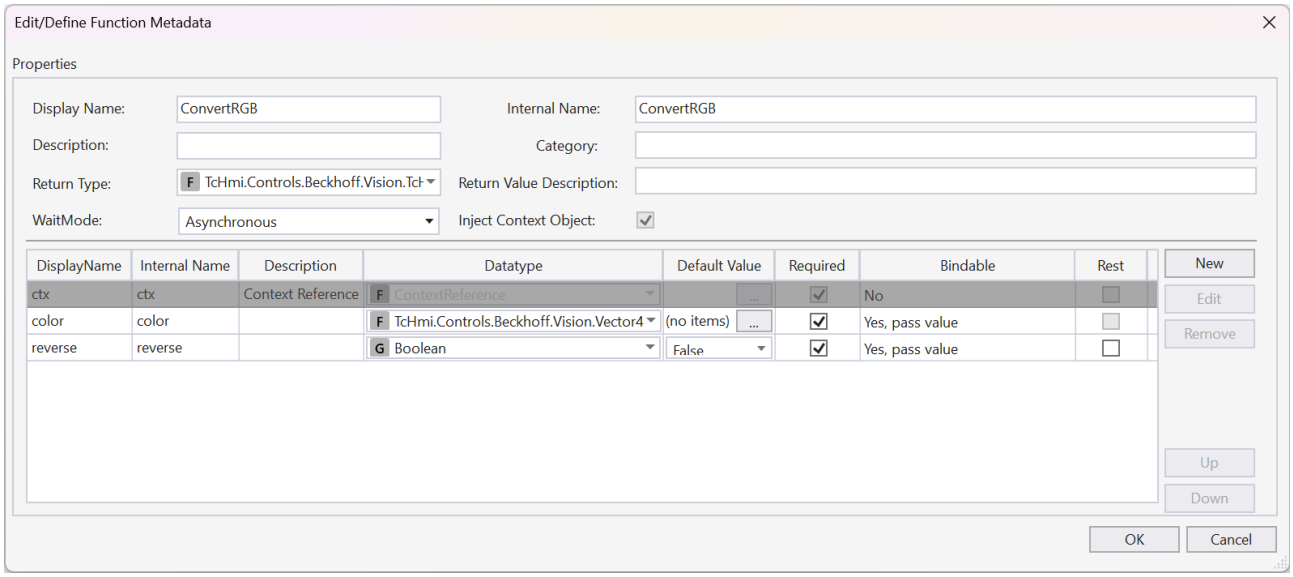
In order for the color box to display the correct color despite the changed value range, the `BoxColorConversion` attribute must be adjusted. Unlike the standard conversions, there is no ready-made function for this. Instead, the creation of a separate function is necessary. The following sample shows how such a function is created for the use case mentioned above and which interfaces it must have.



The general use of functions is explained in the HMI documentation.

After creating the TypeScript function, the WaitMode must be changed to Asynchronous. As a result, the parameter "ctx" is created, which is required for the return. Two parameters must then be created. The first one gets the color value to be converted and therefore must be created of type `TcHmi.Controls.Beckhoff.Vision.Vector4`. The second parameter specifies the conversion direction and must be created as `Boolean`. With `True` the values are converted to RGB and with `False` from RGB back to the user-defined color format. It should be noted that these passed parameters may differ from the values of the corresponding control properties at the time of function execution for timing reasons.

The function must return a value of type `TcHmi.Controls.Beckhoff.Vision.TcHmiVnColor.ColorValue`, which is then displayed in the color box.



In the following code sample, the scaling of the color values is adjusted. If the parameter is `reverse = false`, the color values are transferred from the control to the function in the user-defined format. The color format must then be converted to RGB(a) in order to be displayed in the ColorBox. The color values transferred in the "color" parameter are always 4-channel. If no transparent color is to be displayed in the ColorBox, the returned color values must be 3-channel. Alternatively, the fourth channel can also be set to 255.

If the parameter is `reverse = true`, color values from the color selection dialog are transferred to the function in RGBa format. The color format must then be converted to the user-defined color format. The fourth channel is taken from the color values of the control because the color selection dialog does not support transparency.

```
module TcHmi {
    export module Functions {
        export module 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);
```

This ensures that the color box displays the correct color and that manually changing the color in the color box results in correct color values.

7.2.3 Rectangle (Interactive)

The Rectangle (Interactive) Control provides an environment to display or draw a rectangle by clicks. Mainly, the control is used within the Image Watch [▶ 2707] control; however, it can also be used stand-alone. Thus, for example, image regions can be selected to mark objects or define ROIs. In addition to a rectangle, you can also draw squares, circles or ellipses. To do this, the StrokeShape and AspectRatio attributes must be adjusted accordingly.



For all the following attributes, the control has a getter and a setter method. All further attributes, events and permissions provided by the base class **Controls** can be found in the TwinCAT HMI documentation.

Category: Colors

Attribute	Type	Description
`StrokeColor`	`SolidColor`	Color of the contours of the drawn rectangle.
`HandleColor`	`SolidColor`	Color of the handles of the drawn rectangle.

Category: Common

Attribute	Type	Description
`ShapeData`	`TcVnRotatedRectangle`	Contains the description of the rectangle. Can be used both to retrieve a drawn rectangle and to set a specific rectangle.
`AspectRatio`	`Number`	Specifies the aspect ratio that is enforced when drawing the rectangle. The `0` value does not force a specific aspect ratio.

Function	Description
`clear`	Deletes the drawn rectangle and description from the `ShapeData` attribute.

Category: Usage

Attribute	Type	Description
InitialSelection	InitialSelection	Specifies whether the rectangle can be drawn with two clicks or with a sweeping motion, or using both options.
CenterHandle	Boolean	Displays an element in the center of the rectangle to move it.
CornerHandles	Boolean	Displays elements at the corners of the rectangle to change its size.
EdgeHandles	Boolean	Displays elements on the edges of the rectangle to resize it.
RotationHandle	Boolean	Displays an element on the outside of the rectangle to rotate it.
ShapeDragging	Boolean	Determines whether the rectangle can be moved at the edges.
AngleInterval	Number	Specifies the steps in which the rectangle can be rotated. At 0, the rectangle can be rotated as desired.
Boundary	Boundary	Defines an area in which the shape can be drawn and edited using the five properties "X", "Y", "Width", "Height" and "IsActive". The shape cannot leave this area.
DataUpdate	DataUpdate	Specifies whether ShapeData should be updated during "onChanged" or at the end "onConfirmed" of a graphical interaction.

Category: Appearance

Attribute	Type	Description
StrokeThickness	MeasurementValue	Line thickness of the rectangle.
HandleSize	MeasurementValue	Size of the elements to move and change the rectangle.
StrokeShape	StrokeShape	Specifies whether a rectangle or an ellipse is painted. The description is identical for both. To get a square or a circle, the AspectRatio can be set to 1.
ClickableSize	MeasurementValue	Size of the clickable area around above elements to move and change the rectangle. This can simplify touch screen use without changing the appearance.
Offset	Vector2	Specifies the offset with which the shape is to be displayed.
Scale	Number	Specifies the scaling in which the shape is to be displayed. The value 1 corresponds to 100%.

Events

Event	Description
.onShapeChanged	This event is triggered as soon as the attribute ShapeData changes.
.onShapeConfirmed	This event is triggered when a shape is set or an interaction with it is completed.

7.2.4 Polygon (Interactive)

The Polygon (Interactive) Control provides an environment to display a polygon or draw it by clicks. The control is mainly used within the Image Watch [▶ 2707] control, but it can also be used stand-alone. In addition to a polygon, lines or individual points, for example, can also be displayed or drawn. To do this, the NumPoints property must be set to 1 or 2 accordingly.

> For all the following attributes, the control has a getter and a setter method. All further attributes, events and permissions provided by the base class **Controls** can be found in the TwinCAT HMI documentation.

Category: Colors

Attribute	Type	Description
StrokeColor	SolidColor	Color of the contours of the drawn polygon.
HandleColor	SolidColor	Color of the handles of the drawn polygon.

Category: Common

Attribute	Type	Description
ShapeData	PolygonData	Contains the description of the polygon. Can be used both to retrieve a drawn polygon and to set a specific polygon.
NumPoints	Number	Specifies how many points the complete polygon should have. At 0, any number of points are possible. This specification is important to be able to handle the polygon as complete.
Closed	Boolean	Indicates whether the polygon is closed or whether it is a polyline.

Function	Description
clear	Deletes the painted polygon and description from the ShapeData attribute.

Category: Usage

Attribute	Type	Description
RotationHandle	Boolean	Displays an element on the outside of the polygon to rotate it.
AngleInterval	Number	Specifies the steps in which the polygon can be rotated. At 0, the polygon can be rotated as desired.
Boundary	Boundary	Defines an area in which the shape can be drawn and edited using the five properties "X", "Y", "Width", "Height" and "IsActive". The shape cannot leave this area.
DataUpdate	DataUpdate	Specifies whether ShapeData should be updated during "onChanged" or at the end "onConfirmed" of a graphical interaction.

Category: Appearance

Attribute	Type	Description
StrokeThickness	MeasurementValue	Line thickness of the polygon.
HandleSize	MeasurementValue	Size of the elements to move and change the polygon.
ClickableSize	MeasurementValue	Size of the clickable area around above elements to move and change the polygon. This can simplify touch screen use without changing the appearance.
Offset	Vector2	Specifies the offset with which the shape is to be displayed.
Scale	Number	Specifies the scaling in which the shape is to be displayed. The value 1 corresponds to 100%.

Events

Event	Description
.onShapeChanged	This event is triggered as soon as the attribute ShapeData changes.
.onShapeConfirmed	This event is triggered when a shape is set or an interaction with it is completed.

7.2.5 Help Functions

The following help functions are used as default behavior by certain controls or to ensure interoperability with PLC data types.

BoxColorConverter

The function BoxColorConverter is listed in the category Vision. It is used as the default value for the BoxColorConversion attribute of the Color Control [▶ 2731]. It converts between the RGB and the color format specified by the format parameter.

Parameter	Data type	Description
Color	Vector4	The color value to be converted.
Reverse	Boolean	Direction of the conversion: false: format to RGBa true: RGBa to format
Format	ColorSpace	Format for conversion: Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb
PixelType	ElementType	Scaling of the color values: 8-Bit (USINT), 16-Bit (UINT)

The function returns the converted color as ColorValue.

ConvertColor

The function ConvertColor is listed in the category Vision. It converts a color array between the formats Gray, RGB, RGBa, HSV, HSV FULL, and HLS, HLS FULL, Lab, YUV and YCrCb.

Parameter	Data type	Description
ctx	SelectableRequired<TcHmi.Context, 'success' | 'error'> | null	Is used internally in the HMI to receive asynchronous returns. Can be set to null when called up in JavaScript.
Color	Vector4	The color value to be converted.
SrcFormat	ColorSpace	Format of the incoming color value: Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb
DestFormat	ColorSpace	Format to which is converted: Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb
Precision	Precision	Specifies the number of decimal places to round to. Unrounded is not rounded.
SrcPixelType	ElementType	Scaling of the incoming color values: 8-Bit (USINT), 16-Bit (UINT)
DestPixelType	ElementType	Scaling of the converted color values: 8-Bit (USINT), 16-Bit (UINT)

Errors during conversion result in zero being returned. As the function is asynchronous, it returns the converted color as a future value, i.e. as Promise<ColorValue | null>.

For example, to convert the color red in RGB format to HLS format, the following function call can be used:

```
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
```

In order to be able to process the future `ColorValue`, this can be waited for with `await`. Alternatively, as shown in this sample, the function `then` can be used to pass an anonymous function, e.g. `hls`. This receives the `ColorValue` after the conversion has been completed. The `ColorValue` is then output as `string` in the console.

PixelColorFormatter

The function `PixelColorFormatter` is listed in the category `Formatting`. It is the default value for the `PixelColorFormat` property of the Image Watch Control [▶ 2707]. It converts the `PixelColor` value into a string that can be displayed via HTML. Specifically, the color values of the individual channels are appended to each other and also highlighted in color for certain color formats:



Parameter	Data type	Description
`PixelColor`	`Vector4`	The color value to be converted.
`Format`	`ColorSpace`	Format for conversion: `Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb`
`Precision`	`Precision`	Specifies the number of decimal places to round to. With "Unrounded" there is no rounding.
`PixelType`	`ElementType`	Scaling of the color values: `8-Bit (USINT), 16-Bit (UINT)`

The function returns an HTML coded `String`.

ShapeValueFormatter

The function `ShapeValueFormatter` is listed in the category `Formatting`. It is the default value for the `ShapeValueFormat` property of the Image Watch Control [▶ 2707]. It converts the `ShapeValue` value into a string that can be displayed via HTML. For example, for a rectangle, the area is calculated in pixels and returned as a string:



Parameter	Data type	Description
`ShapeData`	`Any`	The shape data to be converted.
`ShapeType`	ShapeType [▶ 2751]	Type of the shape passed.

The function returns an HTML coded `String`.

ToRotatedRectangle

The `ToRotatedRectangle` function is listed in the Data Conversion category. It converts a `UprightRectangle` to a `RotatedRectangle` with angle `0`.

Parameter	Data type	Description
`rectangle`	TcVnRectangle [▶ 2751]	The rectangle to be converted.

The function returns the converted rectangle as TcVnRotatedRectangle [▶ 2751] and can be applied as follows:

```
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

The `ToUprightRectangle` function is listed in the Data Conversion category. It converts a `RotatedRectangle` to a `UprightRectangle`, discarding the angle.

Parameter	Data type	Description
`rectangle`	TcVnRotatedRectangle [▶ 2751]	The rectangle to be converted.

The function returns the converted rectangle as TcVnRectangle [▶ 2751] and can be applied as follows:

```
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 Data types

ColorChannels

Defines the channels of a color format. Is an array of objects that have the following attributes and each describe a color channel:

Property	Data type	Description
`color`	`SolidColor`	Display color of the channel.
`maxValue`	`Number`	Upper limit of the allowed value range. The lower limit is always 0.
`float`	`Boolean`	Defines whether the channel is described by a float or by an integer.

Used, for example, in Color Control [▶ 2738].

ColorSymbol

Defines a symbol with a ColorValue compatible data type.

ColorValue

Defines a value that can be interpreted as a color. Is either a number describing a grayscale value or a 3- or 4-element array describing a color with a corresponding number of channels.

Valid values are for example:

- `0`
- `128`
- `[0, 0, 255]`
- `[0, 0, 255, 255]`

Orientation

Defines the orientation of a display element. Possible values are "Horizontal" and "Vertical".

TcVnRotatedRectangle

Represents a rectangle, which can be rotated. Is an object with the following properties:

Property	Data type	Description
`aCenter`	`TcVnPoint2_REAL`	Center of the rectangle .
`fAngle`	`DINT`	Angle by which the rectangle is rotated.
`stSize`	`TcVnSize2_REAL`	Size of the rectangle around the center point.

ShapeType

Defines the type of selectable shape in Image Watch Control [▶ 2707]. Is an enum with the following values:

- Point
- Line
- Square
- Rectangle
- Circle
- Ellipse
- Polygon

TcVnRectangle

Represents an axis-aligned rectangle and is an object with the following properties:

Property	Data type	Description
`nHeight`	`DINT`	Height of the rectangle.
`nWidth`	`DINT`	Width of the rectangle.
`nX`	`DINT`	X coordinate of the upper right corner.
`nY`	`DINT`	Y-coordinate of the upper right corner.

UsageMode

Defines the display and editability of display elements. Possible values:

Value	Description
Hidden	The element is not displayed.
ViewOnly	The element is displayed, but is not editable.
Editable	The element is displayed and editable.

Mainly used in Color Control [▶ 2733].

8 Samples and tools

In this chapter you will find samples for the application of many functions and function blocks in the PLC, as well as wizards and camera configuration. Before you start looking at the PLC samples, please have a look at First steps [▶ 27].

The samples are independent of each other. Each sample has a focus under which it is grouped below. Samples that focus on a function or a function block are listed based on the API structure.

You will also find additional support tools:

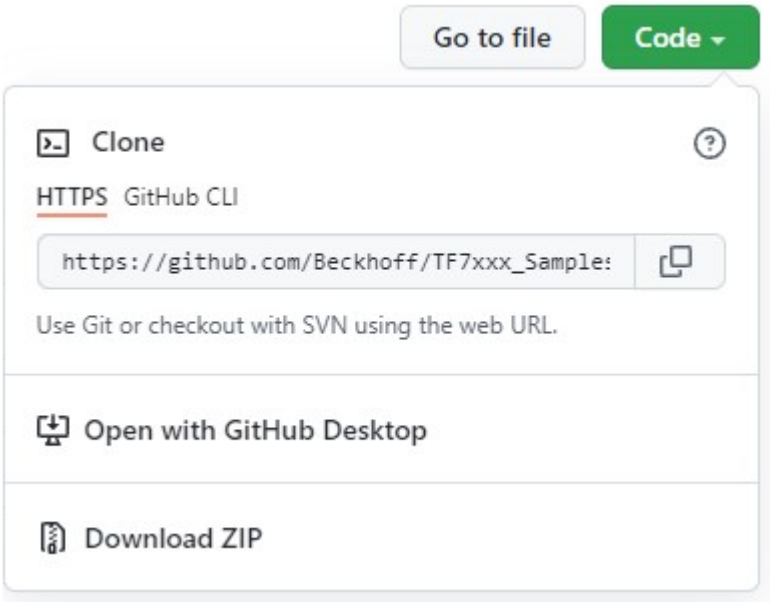
- Neural Network (ONNX) Compatibility Check Tool [▶ 2855]

Download

The samples can be obtained from a repository on GitHub: https://github.com/Beckhoff/TF7xxx_Samples. There you have the option to clone the repository or download a ZIP file containing all samples.

Sample images must be downloaded separately

The GitHub repository contains only the TwinCAT projects with the corresponding code. The image data "TF7xxx_Sample_Images.zip" must be downloaded separately from the Beckhoff website. The link and more information can also be found in the Readme file in the GitHub repository.



Project resources

Each sample is provided in a separate folder, according to the outline order and with the name of the chapter headings. In it there is a TwinCAT project with the corresponding PLC code.

The separate zip folder containing the sample images, for example, can be unzipped to the Images folder within the repository. The folder names as well as the structure in which the sample images are stored is identical to the project structure. Therefore, the assignment of the images is simply done via the path with the same folder name.

Pay attention to short paths to the repository or to the TwinCAT projects, for example

Due to the possible limitation of the path length in the operating system to 255 characters, problems may occur when executing TwinCAT projects if the folder paths are too long. Since the folder structure with the chapter headings is already quite long for the sake of comprehensibility, a short path should be selected for the complete repository or for individual TwinCAT projects, e.g. C:\TF7xxx_Samples\.

Before you execute a project, please adjust the system configuration [▶ 31] according to your target system, with particular attention to configuration of the cores, the router memory and the PLC cycle time. Furthermore, check the version of the TcCOM objects of camera or file source as well as the PLC library as described in the Version overview [▶ 16] and adjust the version if necessary.

In the GitHub repository you can find a PowerShell script (SetTcCOMModuleVersion.ps1) which adjusts the version in all projects of the PLC folder. To do this, run the script, e.g. via the PowerShell console, specifying the desired version as follows:

PS C:\TF7xxx_Samples> .\SetTcCOMModuleVersion.ps1 4.0.4.8

If images or streams are needed, integrate them via the File Source Control [▶ 126] or the Record/Playback [▶ 101] tab of the camera object.

8.1 Function samples

All of the following samples focus on a single function and are sorted in accordance with the TwinCAT Vision API reference. For each sample there are corresponding images that can be loaded to the PLC via the File Source. In addition, the respective sample descriptions show which function parameters have to be changed in order to see certain effects.

The general steps for starting the function samples are:

1. Add the attached sample images in the file source control.
   When using your own images, pay attention to the expected image format, this may vary depending on the example.
2. Activate the configuration and switch TwinCAT and the PLC to Run mode.

⇨ You can now see the images from the PLC via **View > Other Windows > ADS Image Watch**.
- `ipImageInDisp` always show the input image and
- `ipImageResDisp` always show the result image
(Both image names can of course be different for you, we have always named the variables the same in the examples so that they can be easily recognized and you can quickly find your way around in any project)
- if required, other images show results of intermediate steps or further information.
Details can be found in the respective sample description.

In the sample descriptions you will also find the function parameters to be written in logged-in Run mode via

Prepared value, followed by click on **write values**. The effects can then be seen again in the images displayed in Ads Image Watch.

Samples

- Basic Image Operations [▶ 2757]
- Code Reading [▶ 2759]
- Contour Analysis [▶ 2771]
- Image Analysis [▶ 2780]
- Image Color And Contrast Processing [▶ 2800]
- Image Segmentation [▶ 2810]
- Measurement [▶ 2813]

Sample of writing your own functions

If you repeatedly use certain sequences of TwinCAT Vision API functions for your application, it is a good idea to encapsulate them in a function. The following sample explains what you should observe when doing this:

Example: Self-written functions [▶ 2830]

8.1.1 Basic Container Operations

8.1.1.1 Selections of container elements

In this sample, container elements are filtered with the help of a user-defined condition. The function F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 750] is used for this, as the container elements are in turn containers. If the container elements are iterators, the function F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 751] can also be used for this.

Explanation

If the elements of multidimensional containers are to be filtered, this can quickly result in a great deal of programming effort due to the access to the sub-elements. To counteract this, TwinCAT Vision provides a function precisely for this, which is used as follows, for example:

A function block is created that implements the interface `ITcVnCustomElementCondition_ITcVnContainer`. This interface possesses the `Condition` method, which accepts an element of the type `ITcVnContainer`. The filter condition that is to be applied to the element is defined in the implementation of this method. The function block with the filter condition is then transferred to the function `F_VN_CopyContainerElementsConditional_ITcVnContainer` in order to carry out the filtering of all elements of a container. In this sample the elements are filtered according to whether they possess more than five sub-elements.

Application

By way of example, the sub-containers of a container are filtered according to how many elements each of them possesses. Specifically, only those sub-containers that have more than five elements are to be accepted into a new container. The structure of the container can be varied by means of the array `aContainerStructure`. The elements indicate how many sub-elements each sub-container has. If the number of sub-containers itself is likewise varied, the constant `cNumberOfSubContainers` must be matched to it.

Program

MAIN

First of all a constant integer is defined in order to be able to simply vary the sub-containers to be filtered:

```
VAR CONSTANT
    cNumberOfSubContainers  :   INT := 5;
END_VAR
```

Then all necessary variables are declared:

```
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 of the container structure. An array of the length `5` with the initialization values `[2, 7, 3, 5, 12]` means, for example, that a container with 5 sub-containers is to be created in the sample. The first sub-containers is to have 2 elements, the second sub-container 7 elements and so on. The length of the array is defined by the constant `cNumberOfSubContainers`.
ipHelper	Auxiliary interface pointer that is used to fill the main container `ipContainer` with sub-containers.
ipContainerBase	Container in which the sub-containers defined in `aNumberOfElement` are created.
fbCondition	Function block that contains the filter condition.
ipContainerFiltered	Container that, at the end, is to contain only the sub-containers from `ipContainer` that fulfill the filter condition.
nSelectedContainers	Integer that indicates how many sub-containers actually fulfill the filter condition.
hr	Status variable of the type HRESULT [▶ 141].
i	Auxiliary variable

In the program part, a container is created that is filled with the sub-containers defined above. The calling of F_VN_ReserveContainerMemory [▶ 770] is optional and only serves to achieve a better 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
```

The container that has just been created is then filtered. The function block with the self-written condition is specified as the filter condition.

```
hr := F_VN_CopyContainerElementsConditional_ITcVnContainer(
    ipSrcContainer      :=  ipContainerBase,
    ipDestContainer     :=  ipContainerSelection,
    ipConditionFB       :=  fbCondition,
    hr);
```

In order to comprehend the functionality of the function, the elements that fulfill the described condition are counted. The variable `nSelectedContainers` now changes depending on the sub-containers and the filter condition.

```
hr := F_VN_GetNumberOfElements(ipContainerSelection, nSelectedContainers, hr);
```

In this case the number indicates how many sub-containers have more than five sub-elements.

Function block with filter condition

The filter condition is transferred as a function block with a `Condition` method. So that a self-written function block can be transferred to the function, it must implement the interface ITcVnCustomElementCondition_ITcVnContainer [▶ 248]:

```
FUNCTION_BLOCK FB_ConditionMoreThanFive IMPLEMENTS ITcVnCustomElementCondition_ITcVnContainer
```

The `Condition` method defines the filter condition. It is called individually for each sub-container. Depending on whether the return value is `TRUE` or `FALSE`, the respective sub-container is accepted into the new container or not.

```
METHOD Condition : BOOL
```

With each call of the method, the current sub-container is transferred as the input parameter:

```
VAR_INPUT
    ipElement           :   ITcVnContainer;
END_VAR
```

In this case an element is to be accepted if it has more than five sub-elements. For this purpose the number of sub-elements is calculated and, depending on the result, the value `TRUE` or `FALSE` is returned.

```
hr := F_VN_GetNumberOfElements(ipElement, nNumberOfElements, hr);
IF FAILED(hr) OR nNumberOfElements <= 5 THEN
    Condition := FALSE;
```

```
ELSE
    Condition := TRUE;
END_IF
```

The required variables are declared as follows:

```
VAR
    nNumberOfElements   :   ULINT;
    hr                  :   HRESULT;
END_VAR
```

Both the function block itself and the `Condition` method must contain the compiler attribute `{attribute 'c++_compatible'}`.

8.1.2 Basic Image Operations

8.1.2.1 Fusing several images

In this sample, five images are joined using the F_VN_FuseImagesArray [▶ 794] function. This functionality is particularly useful in conjunction with line scan cameras and seamless recording of continuous material flow.

Explanation

The F_VN_FuseImagesArray [▶ 794] function accepts an array of (up to ten) images and joins them in a new image. In the following sample program, initially 5 differently colored images with a height of 10px are created. The number can be changed with the variable *nImages* and the height with *nHeight*. Be sure to vary this number in the range 1-10 only.

The size of the result image is determined by specifying the image lines with the parameters *nFirstLine* and *nNumLines*. This makes it possible to crop the first and last image.

The function F_VN_FuseImages [▶ 793] can also be used if only two images are to be joined.

Program

The following variables are declared:

```
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	Status variable of the type HRESULT [▶ 141].
aImages	Array with several images that are to be joined together.
ipImageFused	Result image in which several images from aImages are joined together.
ipImageFusedDisp	Displayable result image.
nImages	Number of images to be joined.
aColor	Ten different colors in order to be able to tell the images in aImages apart.
i	Auxiliary variable.

In the program the images in the array `aImages` are first of all initialized and colored. In order to be able to visually tell the images apart, ten different colors from `aColor` are used.

```
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
```

After that a defined number `nImages` of images from `aImages` are joined vertically in the image `ipImageFused`:

```
hr := F_VN_FuseImagesArray(aImages, nImages, ipImageFused, 0, nImages * nHeight, hr);
```

Finally, the image is displayed:

```
hr := F_VN_TransformIntoDisplayableImage(ipImageFused, ipImageFusedDisp, hr);
```

Result

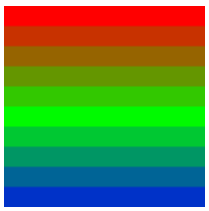
nImages = 2:



nImages = 5:



nImages = 10:



Similar samples

Copy image areas [▶ 2758]

8.1.2.2 Copy image areas

In this sample, two images are copied to another image using different methods:

- Using the function F_VN_CopyImageRegionToRegion [▶ 780],
- Selecting a ROI and copying an image, and
- Selecting a ROI and adding two images using the function F_VN_AddImages [▶ 507].

Explanation

The methods shown below can be used to swap rectangular image areas between images. If required, this can also be done in a partially transparent manner with the aid of arithmetic image operations.

The function F_VN_CopyImageRegionToRegion [▶ 780] takes two images and two image regions and copies the contents of one image region to the other. Both image regions have the same size and must be fully present in the image. The function is used in the sample to copy the red image to the mixed image. However, to copy the blue image, a programmed version with the same functionality is used. The advantage of this version is that instead of copying, it is also possible to mix the blue and the mixed image.

In the sample, the position of the two copied images can be changed using the variables `aPositionRed` and `aPositionBlue`. The parameter `bCopyBlue` determines whether the blue image is copied or mixed.

Variables

```
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;
```

Program

```
// 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);

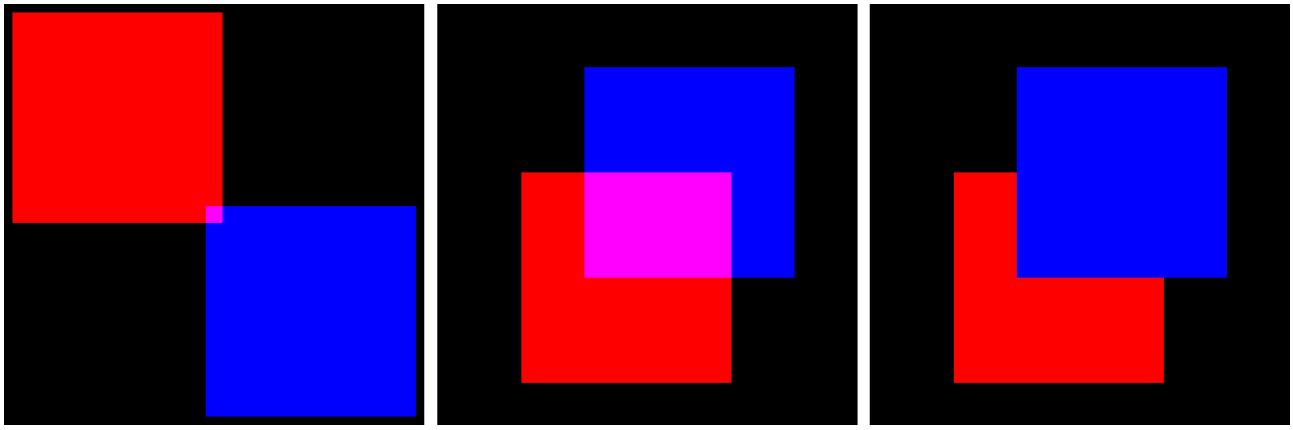
// Copy red image to selected region in merge-image
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage  :=  ipImageRed,
    nXSrc       :=  0,      nYSrc   :=  0,
    nWidth      :=  500,    nHeight :=  500,
    ipDestImage :=  ipImageMerge,
    nXDest      :=  aPositionRed[0],
    nYDest      :=  aPositionRed[1],
    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);
```

Result

The following are samples of the program results. In the images on the left and in the center, the blue square was added, in the image on the right it was copied. In addition, the target positions of the squares are different.



Similar samples

Fusing several images [▶ 2757]

8.1.3 Code Reading

8.1.3.1 EAN-13 Barcode Reading

This sample illustrates the following

- One EAN13 code is read per image using the function F_VN_ReadBarcodeExp [▶ 841]
- The execution time of the function is monitored via a watchdog [▶ 146].

Explanation

- The functions F_VN_ReadBarcode [▶ 839] / F_VN_ReadBarcodeExp [▶ 841] can search and read barcodes of different types in the image. To speed up the processing time, the barcode type (`eBarcodeType`) can be specified. In this example, these are `TCVN_BT_EAN13`. In addition, the processing time can be monitored via a watchdog. The sample Blob Detection with watchdog monitoring [▶ 2780] provides more comprehensive information on the watchdog.
- Compared to F_VN_ReadBarcode [▶ 841], the function F_VN_ReadBarcodeExp [▶ 839], which is used in this example, has the following features:
  - in addition to the barcode type, the search direction (`eSearchDirection`) can be specified:
    - TCVN_BSD_ANY first searches in horizontal direction and then in vertical direction
    - TCVN_BSD_HORIZONTAL only searches in horizontal direction
    - TCVN_BSD_VERTICAL only searches in vertical direction
  - the code section in which the readout took place can be returned.

Variables

```
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);
```

Results

If TCVN_BSD_ANY is selected as the search direction, both horizontal and vertical EAN13 codes will be detected. When comparing the watchdog time, however, it becomes apparent that horizontal EAN13 codes (first image) are recognized much faster than vertical EAN13 codes (second image), since the code search initially takes place in horizontal direction.





If the search direction is set to TCVN_BSD_VERTICAL, the vertical EAN13 code is recognized much faster (third image). However, in this case there is no search for horizontal codes.



Similar samples

- Data Matrix Code Reading [▶ 2763]
- Pharma-Code Reading [▶ 2764]
- QR-Code Reading [▶ 2766]

- Result evaluation during Code Reading [▶ 2768]

8.1.3.2 Data Matrix Code Reading

This sample illustrates the following

- For each image a Data Matrix Code is read using the function F_VN_ReadDataMatrixCodeExp [▶ 847]
- The execution time of the function is monitored via a watchdog [▶ 146].

Explanation

The function F_VN_ReadDataMatrixCodeExp [▶ 847] searches for Data Matrix Codes in the specified image and reads them. In contrast to the F_VN_ReadDataMatrixCode [▶ 845] function, the number of codes to be searched for each image can be specified. In this sample, it is one code per image. In addition, the contour of the code that is found is returned. `TCVN_CSS_ONLY_NOT_FLIPPED` is selected as the search strategy, since the image is not mirrored.

If there are areas in the image that look similar to codes, or if there is no code at all in the image, this can have a negative effect on the execution time. To prevent cycle time overruns, in this sample the function F_VN_ReadDataMatrixCodeExp [▶ 847] is monitored with the aid of a watchdog [▶ 146] (see the sample on Blob Detection with watchdog monitoring [▶ 2780]).

Variables

```
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);

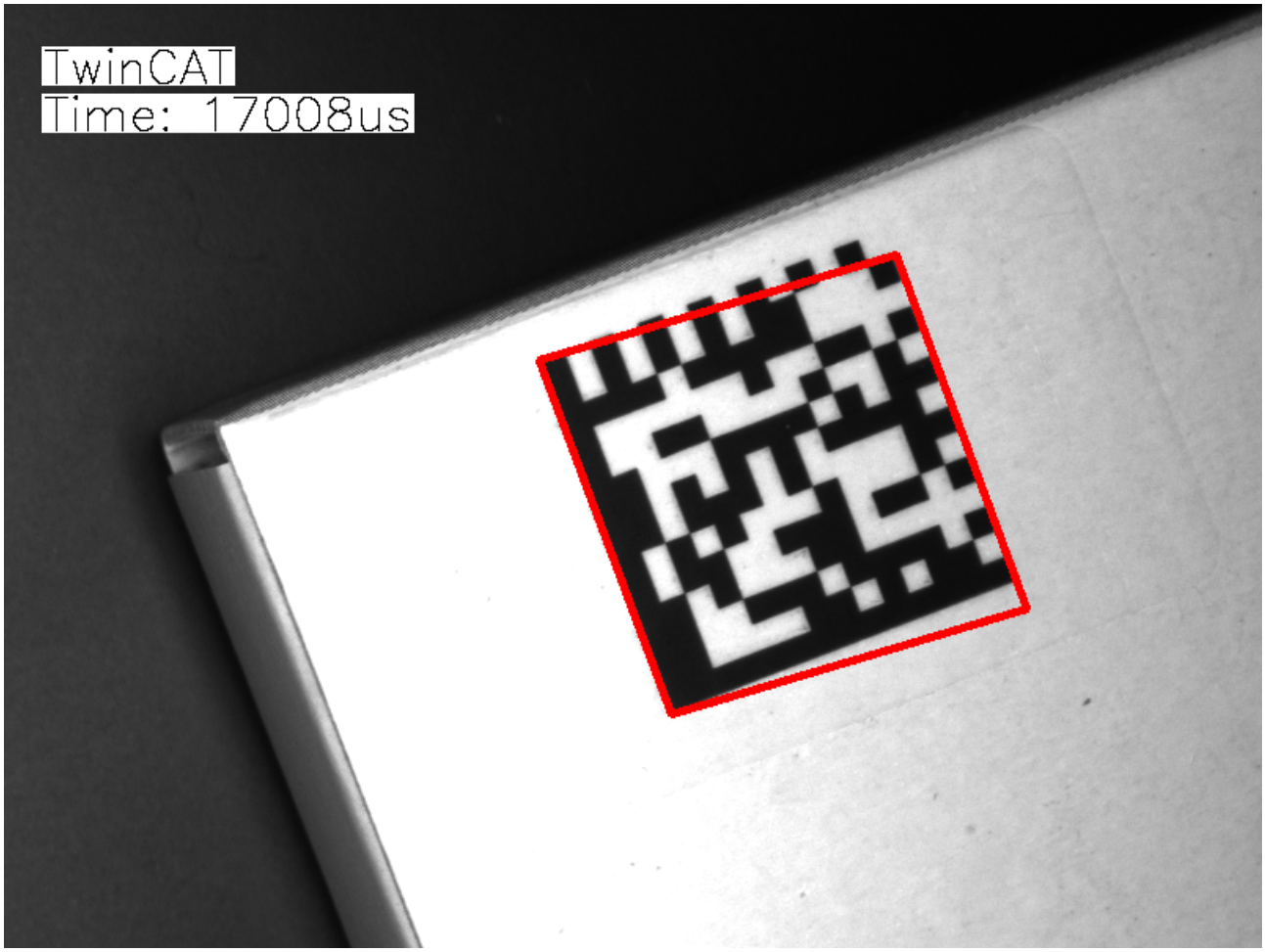
// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // 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);
```

Result



Similar samples

- EAN-13 Barcode Reading [▶ 2759]
- Pharma-Code Reading [▶ 2764]
- QR-Code Reading [▶ 2766]
- Result evaluation during Code Reading [▶ 2768]

8.1.3.3 Pharma-Code Reading

This sample illustrates the following

- One pharma code is read per image using the function F_VN_ReadPharmaCodeExp [▶ 852]
- The execution time of the function is monitored via a watchdog [▶ 146].

Explanation

- The function F_VN_ReadPharmaCodeExp [▶ 852] searches for pharma codes in the specified image and reads them. In contrast to the function F_VN_ReadPharmaCode [▶ 850], the code position is also returned.
- Code-like structures have a negative effect on the execution time. To prevent cycle time overruns, in the sample the function F_VN_ReadPharmaCodeExp [▶ 852] is monitored via a watchdog [▶ 146]. The sample Blob Detection with watchdog monitoring [▶ 2780] explains how to use a watchdog [▶ 146].

Variables

```
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);
```

Result



Similar samples

- Data Matrix Code Reading [▶ 2763]
- EAN-13 Barcode Reading [▶ 2759]
- QR-Code Reading [▶ 2766]
- Result evaluation during Code Reading [▶ 2768]

8.1.3.4 QR-Code Reading

This sample illustrates the following

- One QR code is read per image using the function F_VN_ReadBarcodeExp [▶ 858]
- The execution time of the function is monitored via a watchdog [▶ 146].

Explanation

The function F_VN_ReadPharmaCodeExp [▶ 858] searches for QR codes in the specified image and reads them. In contrast to the function F_VN_ReadQRCode [▶ 855], the contour of the code that is found is also returned.

If there are areas in the image that look similar to codes, or if there is no code at all in the image, this can have a negative effect on the execution time. To prevent cycle time overruns, in this sample the function F_VN_ReadQRCodeExp [▶ 858] is monitored with the aid of a watchdog [▶ 146] (see the sample on Blob Detection with watchdog monitoring [▶ 2780]).

Variables

```
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);
```

Result



Similar samples

- Data Matrix Code Reading [▶ 2763]
- EAN-13 Barcode Reading [▶ 2759]
- Pharma-Code Reading [▶ 2764]
- Result evaluation during Code Reading [▶ 2768]

8.1.3.5 Result evaluation during Code Reading

This sample illustrates the following

- One EAN13 code is read per image using the function F_VN_ReadBarcode [▶ 841]
- The execution time of the function is monitored via a watchdog [▶ 146].
- Handling of different return codes of the function.

Explanation

- The function F_VN_ReadBarcode [▶ 839] can search and read barcodes of different types in the image. Details of the use of the function can be found in the sample EAN-13 Barcode Reading [▶ 2759].
- The handling of the return code of the function is explained using a corresponding query. A distinction is made between the following cases:
  - Successful execution of the function, code was found and decoded.
  - Successful execution of the function, code is not present in the image or cannot be found / decoded.
  - The function itself has no error and was aborted by the watchdog.
  - Function has other error.

You can find a detailed explanation of the return codes for Code Reading functions in the overview chapter Code Reading [▶ 837].

Variables

```
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);
```

NOTICE

Reserve hr

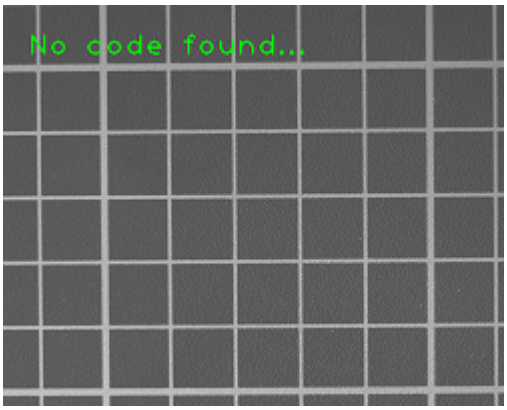
Make sure not to use the variable hr between the call of the Code Reading function and the evaluation as a return value of another function.

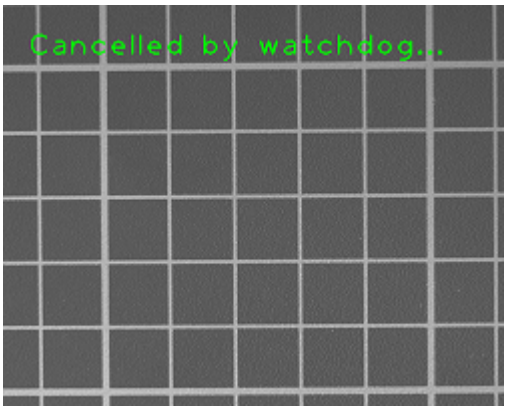
Results

If a code is found as expected, is `hr = S_OK` and the result can be output.



If the expected code cannot be found, the return code is `S_FALSE`. If execution is aborted by a watchdog [▶ 146], the code is `16#256`. A corresponding message can be output.





If the function is not executed correctly, the error code can be output. This does not occur in the sample and is only caused by incorrect use of Code Reading functions.

Similar samples

- Data Matrix Code Reading [▶ 2763]
- EAN-13 Barcode Reading [▶ 2759]
- Pharma-Code Reading [▶ 2764]
- QR-Code Reading [▶ 2766]

8.1.4 Contour analysis

8.1.4.1 Match Contours 1vsN (manual shapes)

In this sample, contours are compared with reference contours using the function F_VN_MatchContours1vsN [▶ 984]. Where

- the reference contours are idealized, manually created polygons
- and the other contours are extracted from an image.

Explanation

The function F_VN_MatchContours [▶ 983] compares two contours used in the sample based on the Hu moments. The function F_VN_MatchContours1vsN [▶ 984] represents an extension that can be used to compare a contour simultaneously with several other contours. In this sample, this is used to classify workpieces according to predefined reference shapes.

The reference shapes are idealized, manually created polygons of an isosceles triangle, a rectangle/square and a trapezoid. Since only the corner points were specified for the polygons, the reference shapes differ greatly from the test contours in the number of descriptive contour points. Furthermore, they differ in position, orientation, size and the degree of interference within the contour.

The function F_VN_DetectBlobs [▶ 1178] is used to search for contours with a certain minimum size in order to extract the shape of the components from the image. The contours that are found are then used for the described matching.

Input parameters

In addition to the contours to be compared, only a parameter hast to be passed to the function that describes the calculation method used to determine the dissimilarity from the Hu moments (**eComparisonMethod** of type ETcVnContoursMatchComparisonMethod [▶ 193]). The method `TCVN_CMCM_CONTOURS_MATCH_I1` calculates the sum of the differences in the reciprocal values of the individual characteristics, while the method `TCVN_CMCM_CONTOURS_MATCH_I2` calculates the sum of the differences in the pure characteristics. Unlike the first two methods, the third variant `TCVN_CMCM_CONTOURS_MATCH_I3` only calculates the maximum difference between the individual characteristics. Which of the three methods is best suited for comparing two contours depends on the use case.

Variables

Data type only in this sample

The CustomTcVnArray4_Point2_DINT data type is not included in TwinCAT Vision, but was created explicitly for this sample. The implementation can be found in the download of this sample.

```
// 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));

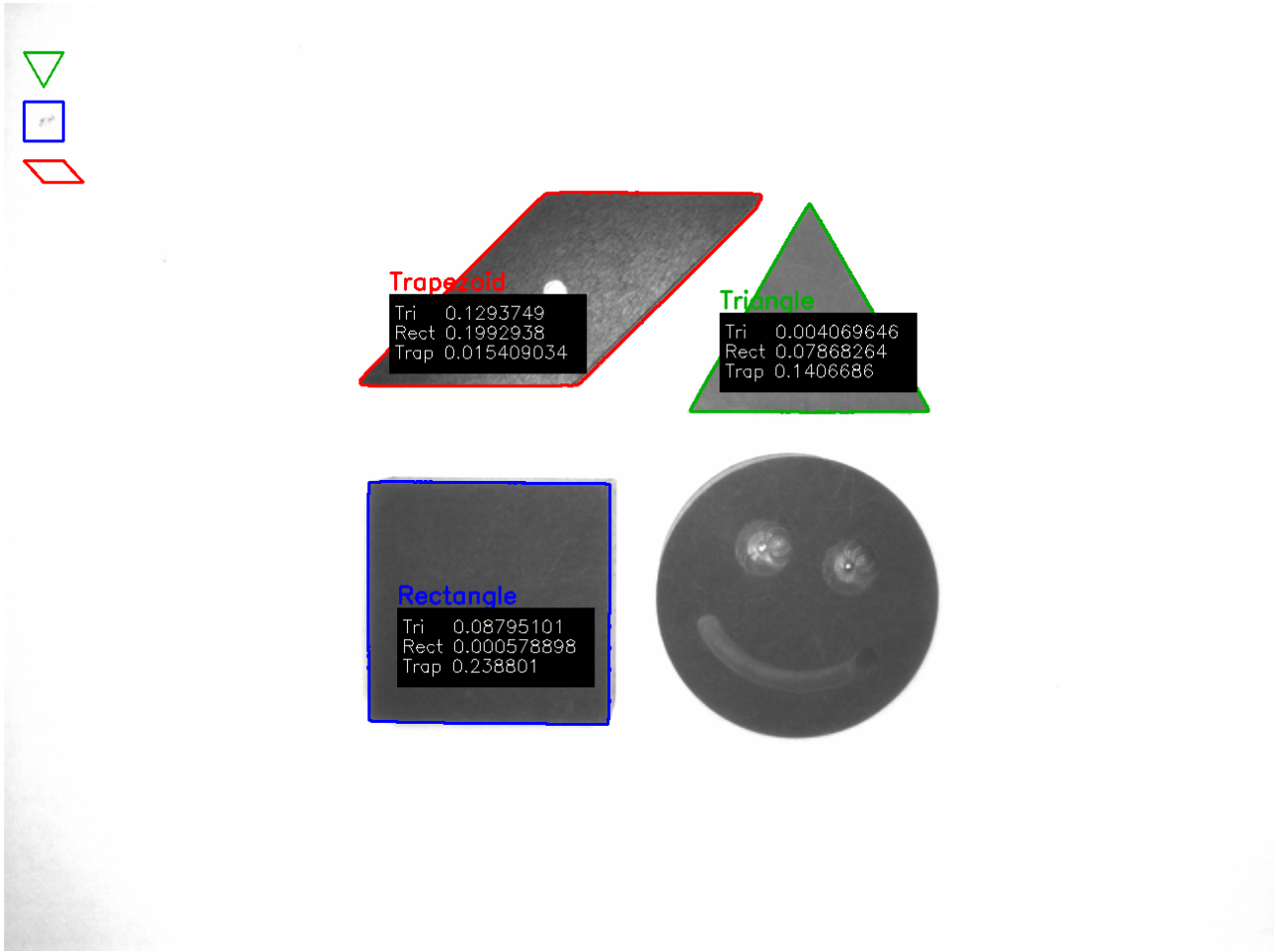
        // Draw matching result
        hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColors[aMatchIndexes[0]], 5, hr);
        hr := F_VN_PutTextExp(aTexts[aMatchIndexes[0]], ipImageRes, nTopLeftX, nTopLeftY,
TCVN_FT_HERSHEY_SIMPLEX, 0.8, aColors[aMatchIndexes[0]], 2, TCVN_LT_8_CONNECTED, FALSE, hr);
        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

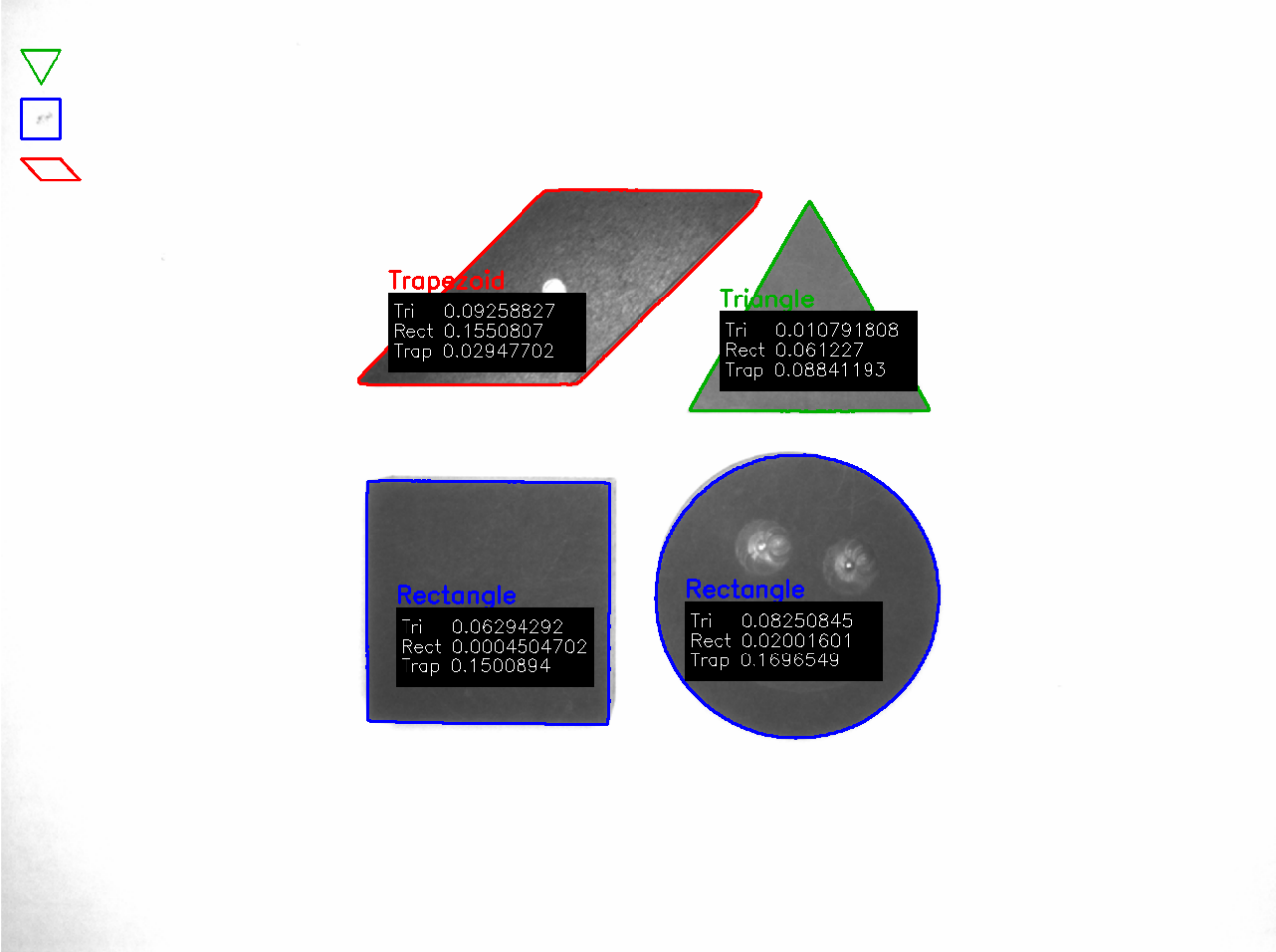
// Draw reference shapes
FOR i:=TRIANGLE TO TRAPEZOID DO
    hr := F_VN_DrawContoursExp(aShapes[i], -1, ipImageRes, aColors[i], 2, TCVN_LT_8_CONNECTED, 0, 0,
aOffsets[i], hr);
END_FOR
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
```

Results

The result image shows the reference shapes (left border) and the found contours whose dissimilarity relating to all reference shapes does not exceed a maximum value. Furthermore, a visualization indicates which reference shape the contour resembles most closely. A corresponding description text (including dissimilarities for each reference shape) is superimposed over the respective contour. For the parameters used in this sample, the result for the MatchContours1.bmp image looks as follows:



In each case the lowest value indicates the correct reference contour. If the parameter eComparisonMethod is changed from `TCVN_CMCM_CONTOURS_MATCH_I3` to `TCVN_CMCM_CONTOURS_MATCH_I2`, the results change considerably. This also means that the parameter for the maximum accepted dissimilarity must be adjusted:



The new values allow a much less clear assignment to corresponding reference shapes, and the round component is incorrectly assigned to the rectangle reference shape. This calculation method is therefore obviously less suitable for this sample.

Similar samples

- Match Contours (extracted shapes) [▶ 2774]

8.1.4.2 Match Contours (extracted shapes)

In this sample, two contours are compared using the function F_VN_MatchContours [▶ 983].

Explanation

The function F_VN_MatchContours [▶ 983] compares two contours used in the sample based on the Hu moments. In this sample, the function is used to recognize a given component in a new situation. For this purpose, a component is selected in a reference image, its contour is saved and then compared with the contours from another image. In this way, the components that fall below a certain dissimilarity compared to the reference component are marked.

Input parameters

In addition to the contours to be compared, only a parameter hast to be passed to the function that describes the calculation method used to determine the dissimilarity from the Hu moments (**eComparisonMethod** of type ETcVnContoursMatchComparisonMethod [▶ 193]). The method TCVN_CMCM_CONTOURS_MATCH_I1 calculates the sum of the differences in the reciprocal values of the

individual characteristics, while the method TCVN_CMCM_CONTOURS_MATCH_I2 calculates the sum of the differences in the pure characteristics. Unlike the first two methods, the third variant TCVN_CMCM_CONTOURS_MATCH_I3 only calculates the maximum difference between the individual characteristics. Which of the three methods is best suited for comparing two contours depends on the use case.

Variables

```
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

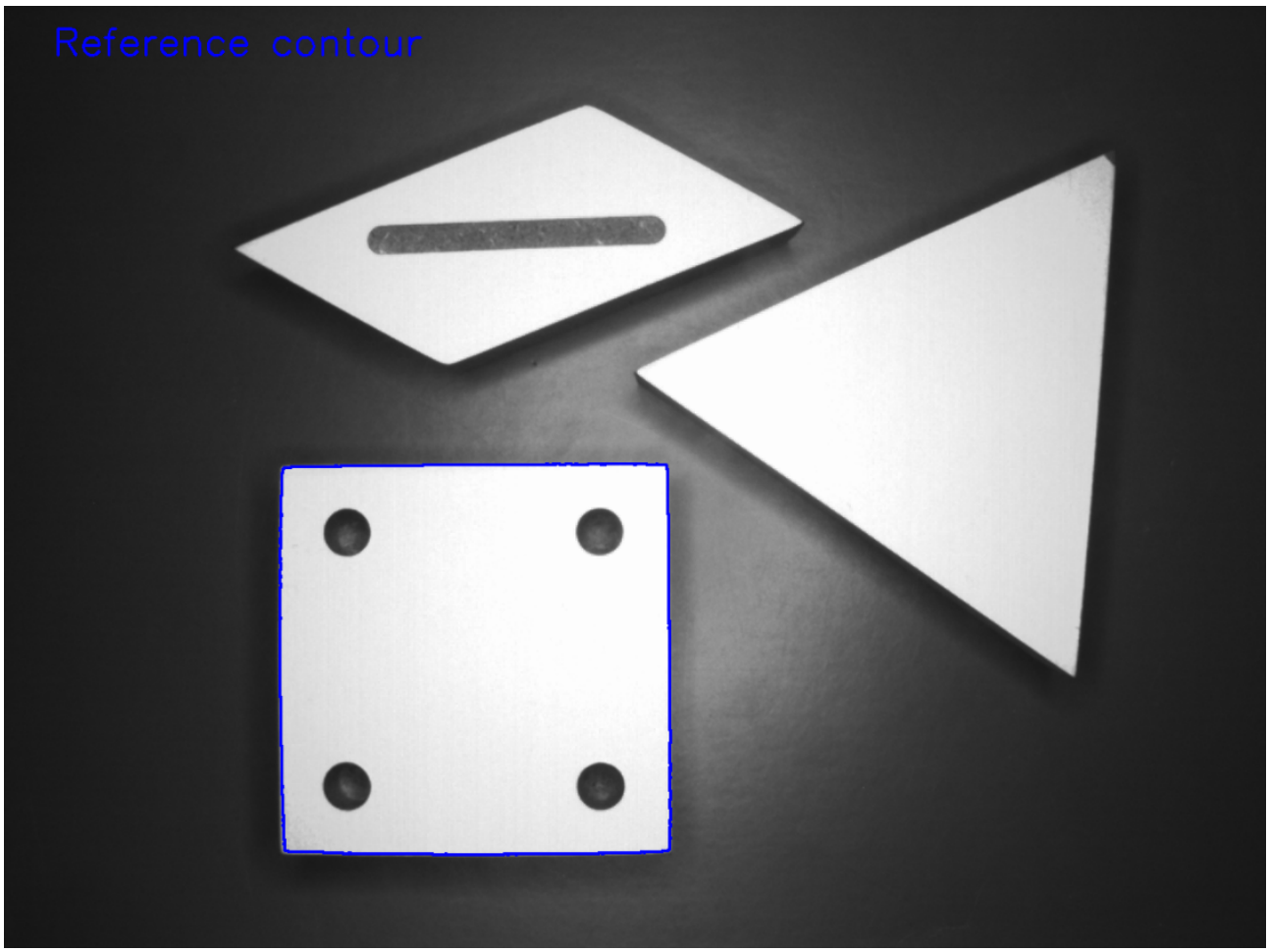
    // Draw text
    hr := F_VN_PutTextExp(sTextCheck, ipImageRes, 50, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen,
2, TCVN_LT_8_CONNECTED, FALSE, hr);
    sText := CONCAT('Best match: ', REAL_TO_STRING(LREAL_TO_REAL(fBestDissimilarity)));
    hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);

END_IF

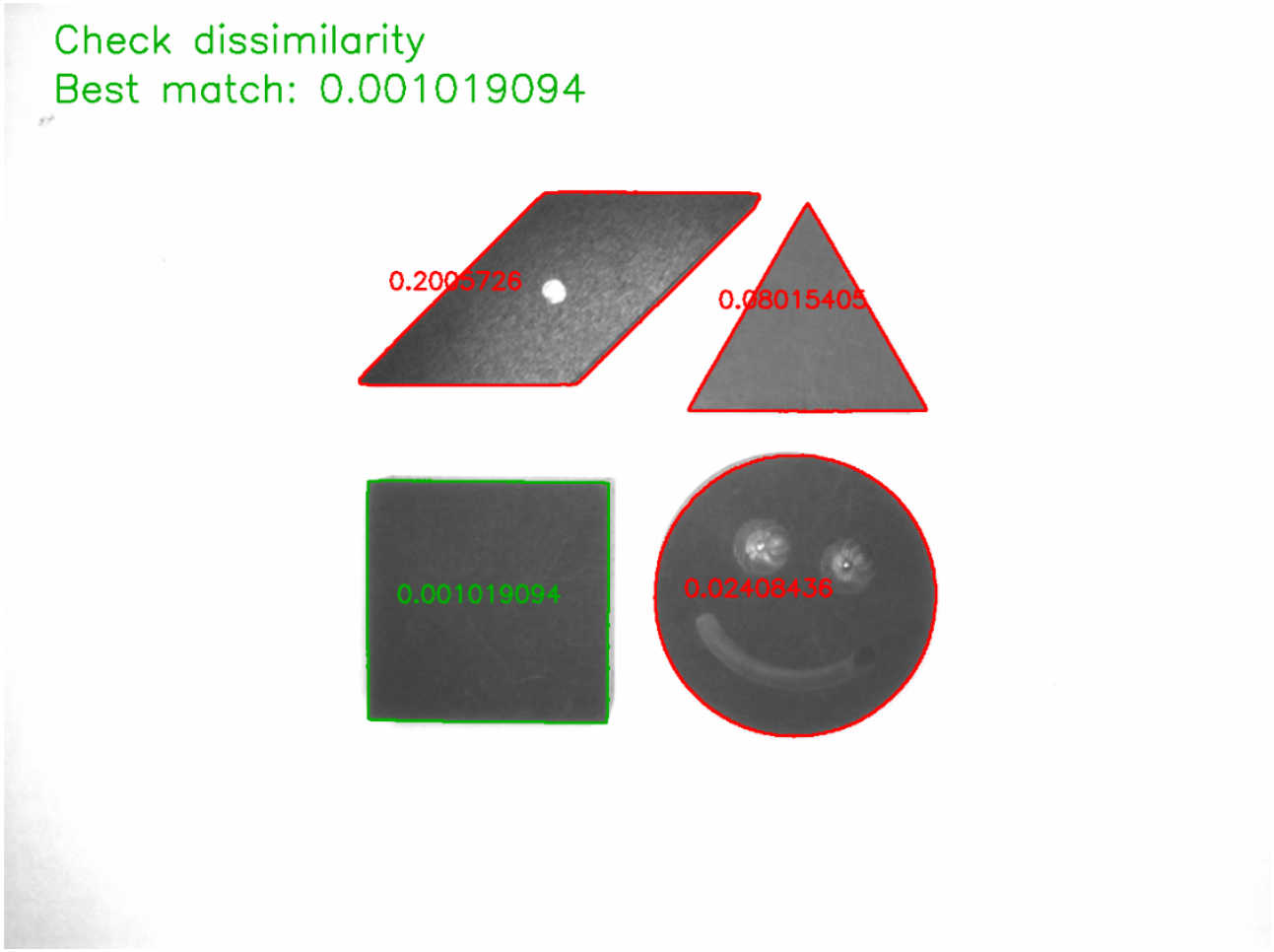
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
```

Results

The currently selected reference contour is marked in blue on the reference image:



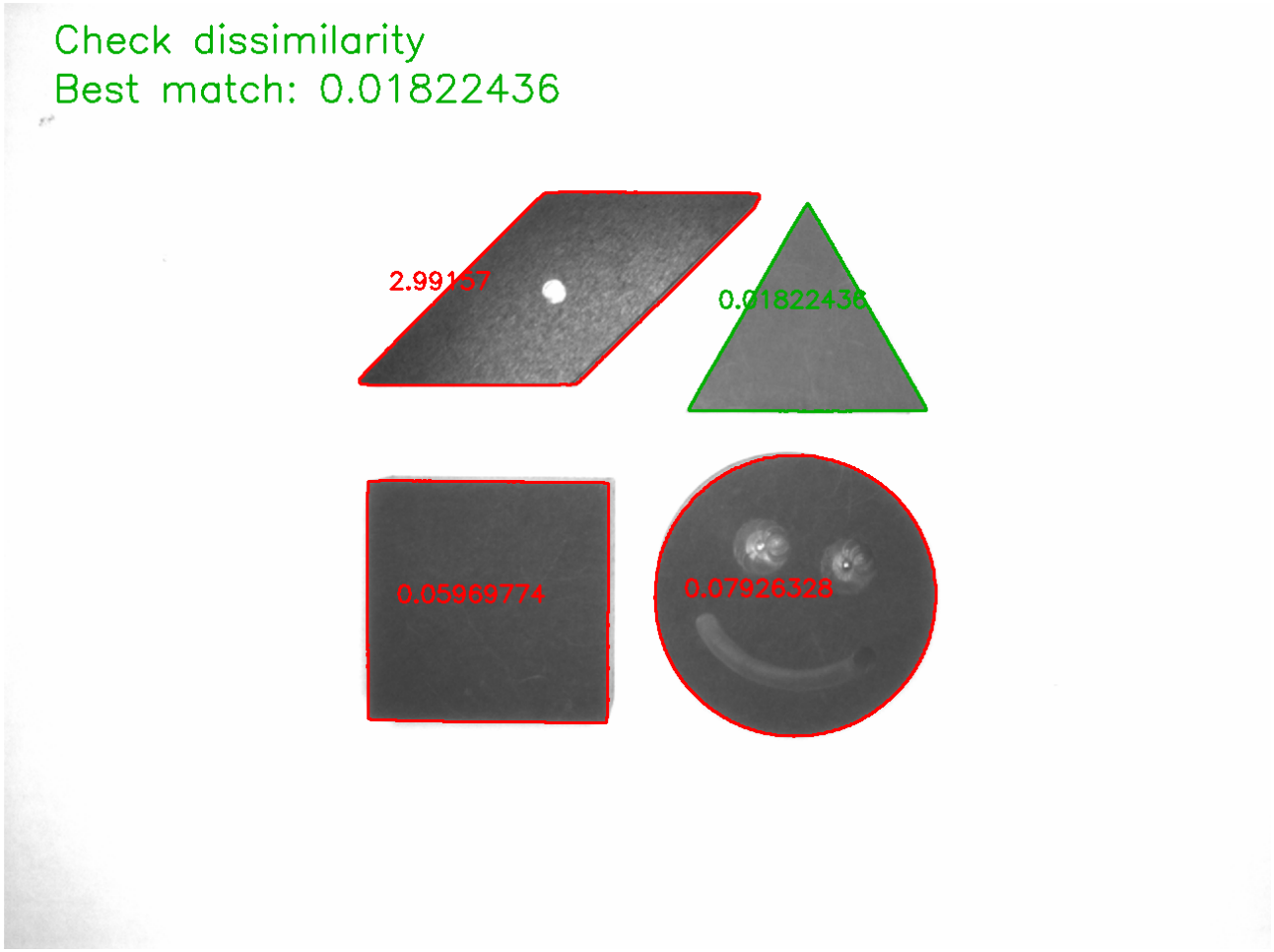
You can see that the contour is slightly curved by the lens distortion and has some minor flaws. In the test pattern, contours whose dissimilarity relative to the reference contour falls below the set threshold value are displayed in green, the other contours are shown in red. In addition, the calculated dissimilarity is superimposed over all contours. The following illustration shows the result for the *MatchContours1.bmp* input imaged when the rectangle is selected as a reference component:



It can be seen that the rectangular component has by far the smallest dissimilarity compared to the reference contour. If the parameter eComparisonMethod is changed from TCVN_CMCM_CONTOURS_MATCH_I3 to TCVN_CMCM_CONTOURS_MATCH_I2, the calculated dissimilarities change accordingly:



In this case, the result is even clearer, but this does not apply to all shapes. If the triangle is selected as the reference component, the result with this setting is significantly less clear than with the original setting:



This result demonstrates that different calculation methods can provide the best results, depending on the application.

Similar sample

- Match Contours 1vsN (manual shapes) [▶ 2771]

8.1.5 Image analysis

8.1.5.1 Object Detection

8.1.5.1.1 Blob Detection with watchdog monitoring

This sample illustrates the following

- The function F_VN_DetectBlobs [▶ 1178] is used to find round structures in images.
- The execution time of the function is monitored via a watchdog.

Explanation

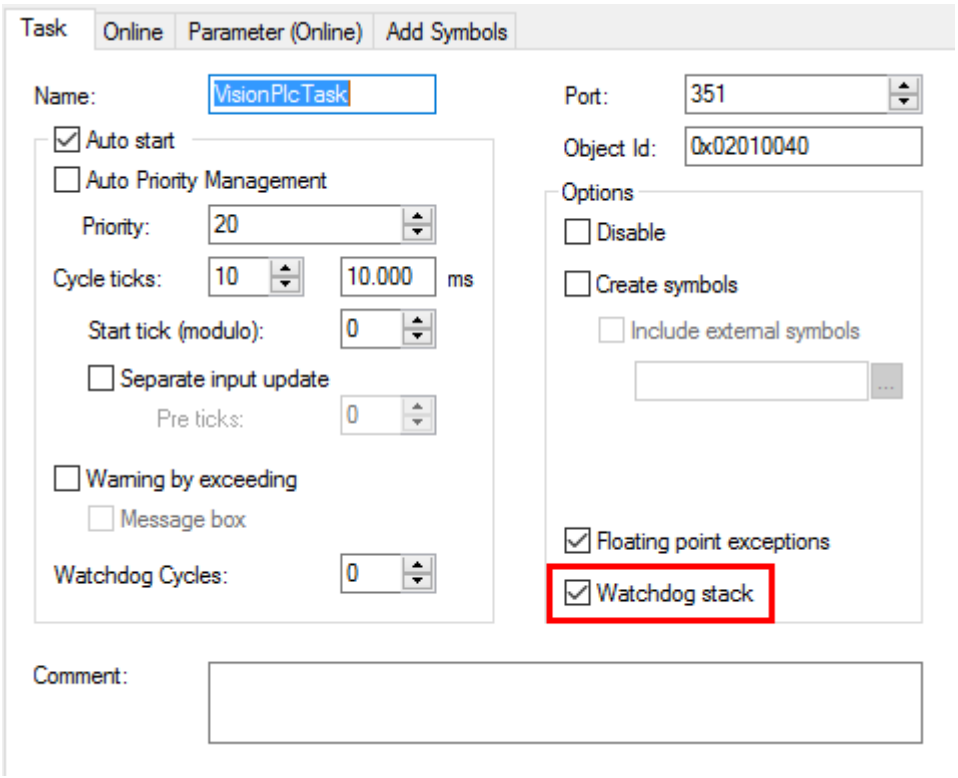
The F_VN_DetectBlobs [▶ 1178] function facilitates the search for similar structures in the image. It combines setting threshold values with searching and filtering for contours in a single call. In this sample, the function is parameterized to return all round structures in the example images.

Due to external influences, the actual image content can deviate greatly from the expected image content. With image processing algorithms whose execution time depends on the image content, this can lead to cycle time overruns and thus to undesired side effects. To prevent this there is a watchdog. The watchdog can abort functions marked with the sentence: *"Can return partial results when canceled by Watchdog"* in the API after a specified period of time. The results obtained up to this point can be used further.

F_VN_DetectBlobs is one the functions that can be monitored in this way. By changing the permissible execution time `tStop` in the watchdog start function F_VN_StartRelWatchdog [▶ 991], a function termination followed by application of partial results can be observed and compared to full execution of F_VN_DetectBlobs. The watchdog can be stopped via F_VN_StopWatchdog [▶ 993]. A positive return value in `tRest` specifies the remaining time, while a negative return value in `tRest` specifies the additional time required for termination.

Task configuration

Watchdog monitoring [▶ 146] must be enabled in the executing PLC task. Otherwise the watchdog function calls in the PLC are ignored and the start watchdog function returns the return value 16#71A: "No Interface".



Variables

```
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);

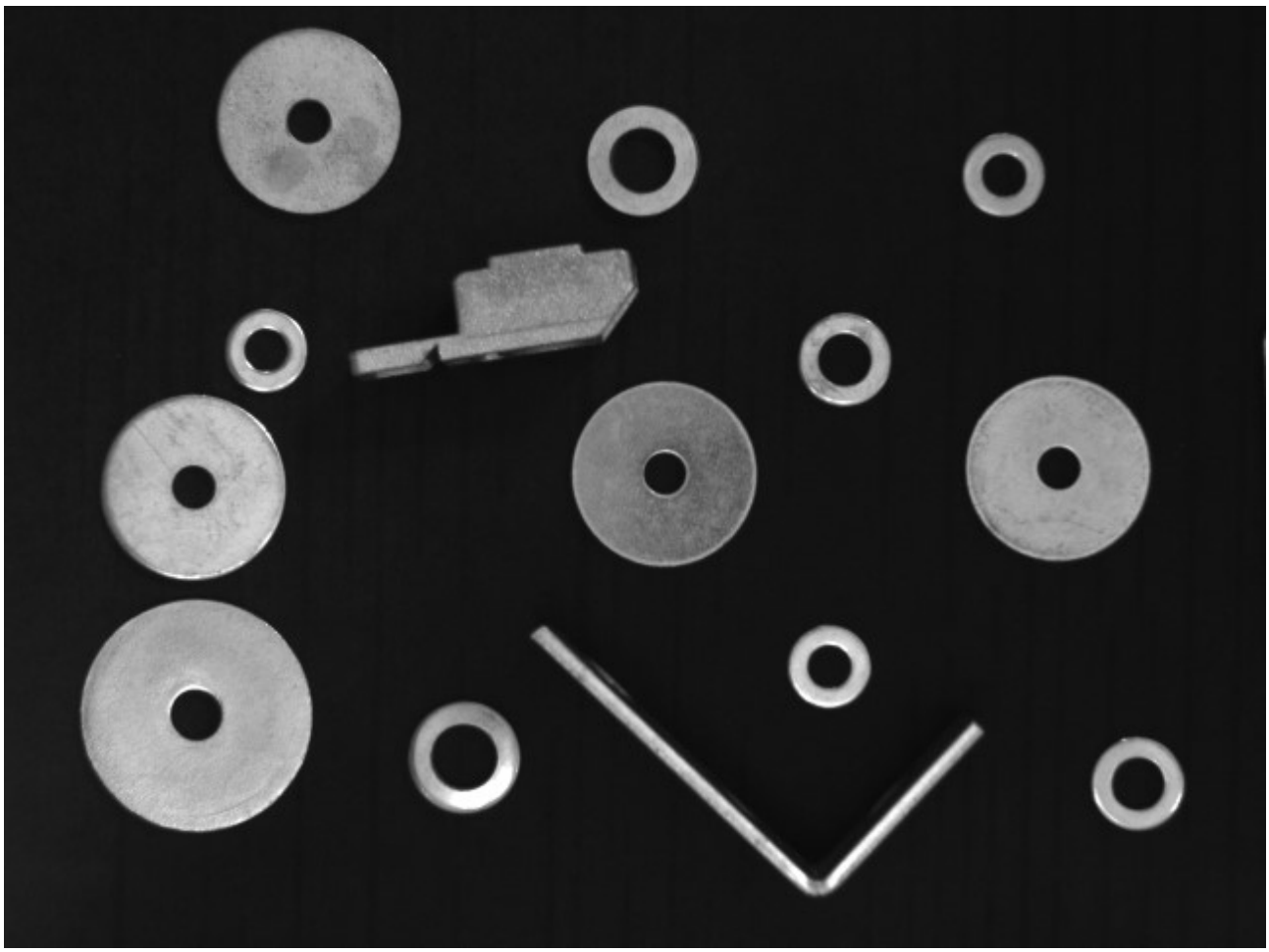
sText := CONCAT(CONCAT('Processed ', UDINT_TO_STRING(nFraction)), '%');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);

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);

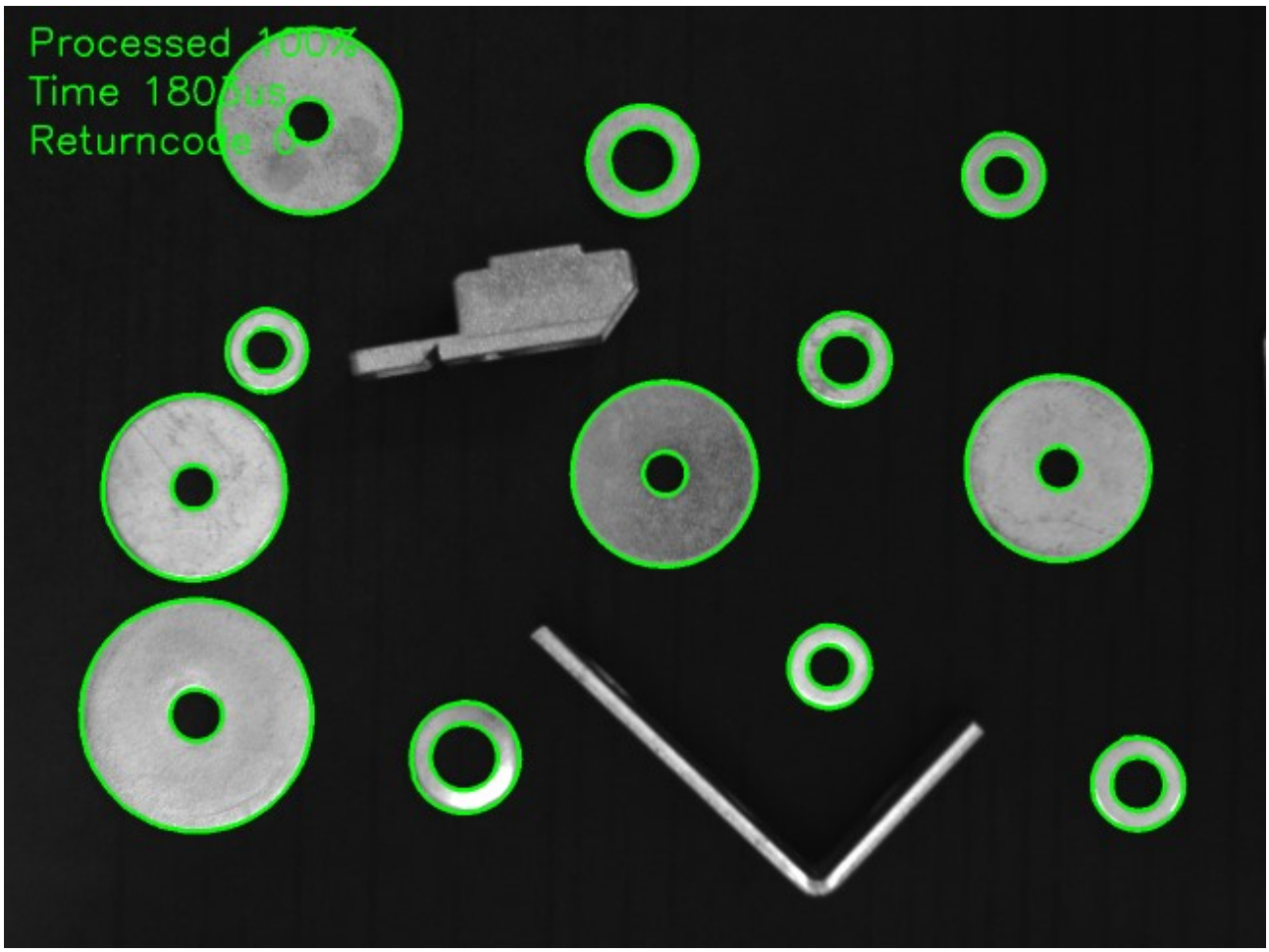
sText := CONCAT('Returncode ', DINT_TO_STRING(hrFunc));
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 150, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
```

Results

The input image

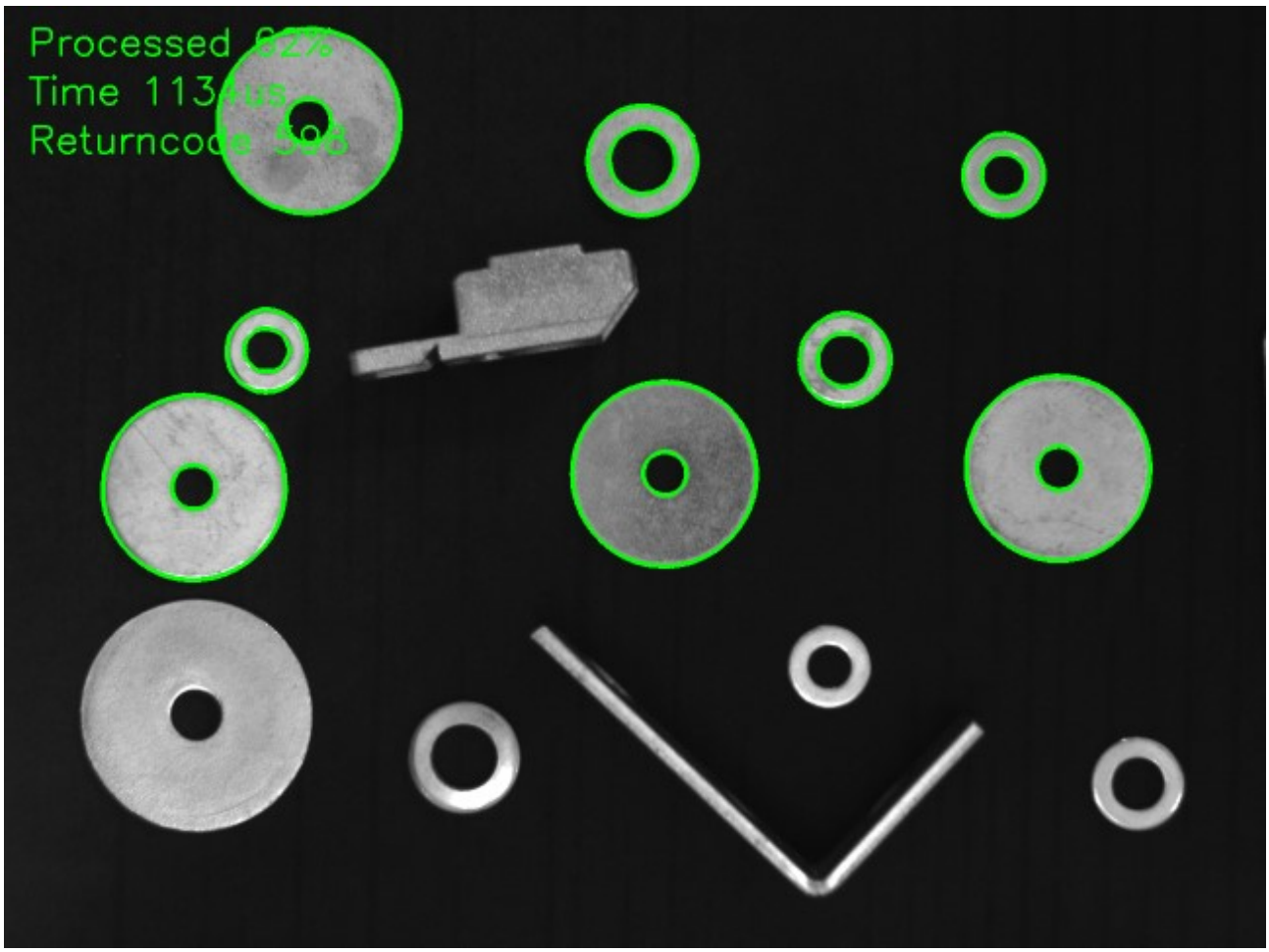


All circles were found in the image, with the parameterization of the sample code.



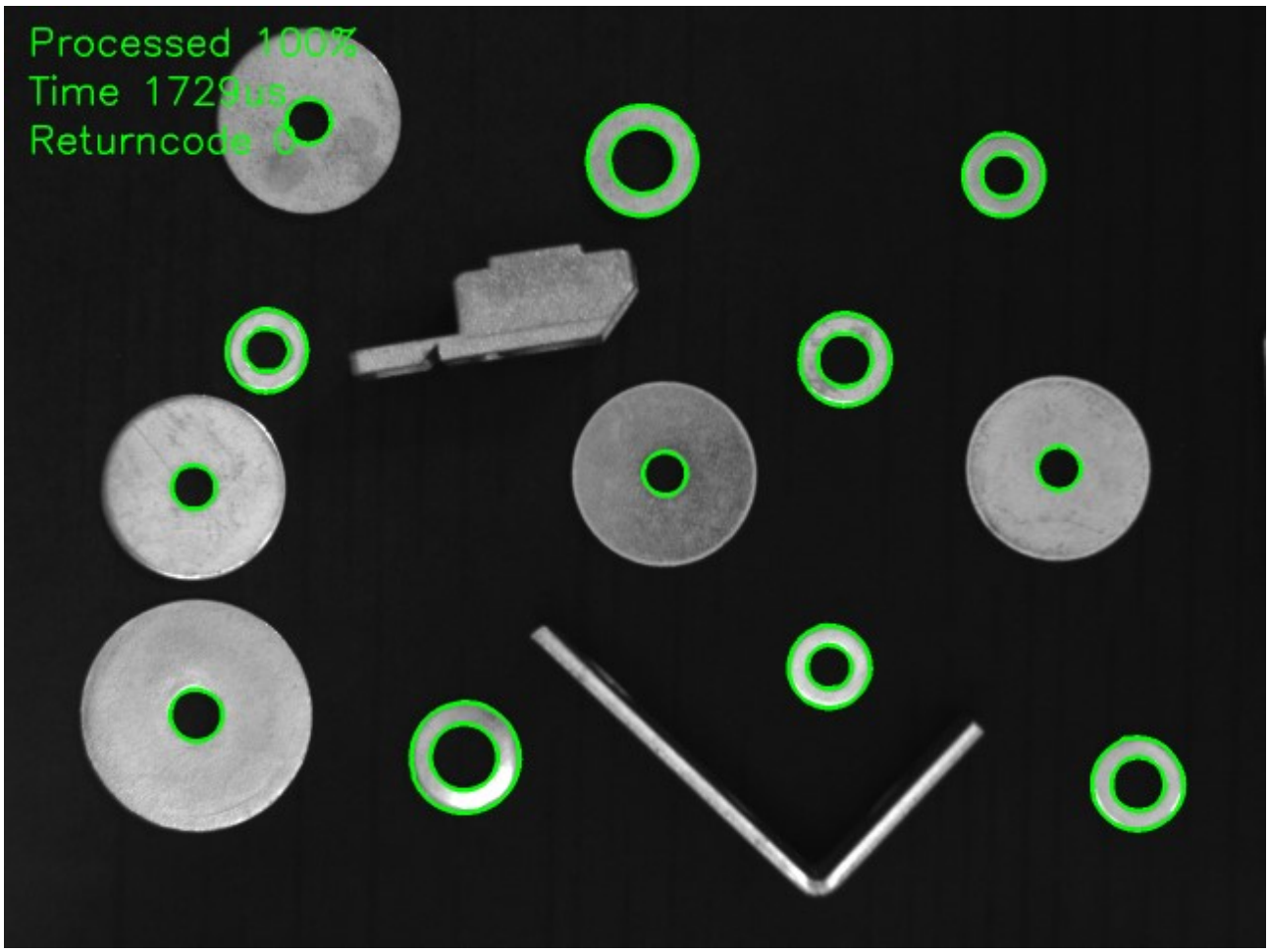
Some of the circles were found in the image, by triggering the watchdog after 1 ms. Since the abort process takes some time, the actual time required is greater than 1 ms.

```
hrWD := F_VN_StartRelWatchdog(tStop:=1000, WATCHDOG_ACCUMULATION_TYPE_MEAN, S_OK);
```



Only the small circles were found in the image by reducing the maximum enclosing area in pixels.

```
stBlobParams.fMinArea := 100;
stBlobParams.fMaxArea := 10000;
```



Similar sample

- Find contour instead of Blob Detection [▶ 2786]

8.1.5.1.2 Find contour instead of Blob Detection

This sample shows an alternative solution to the Blob Detection sample [▶ 2780] using F_VN_FindContoursExp [▶ 1190].

Explanation

The function F_VN_DetectBlobs [▶ 1178] consolidates many individual image processing steps in one function call. However, the internal processing steps can also be carried out by individual function calls. This can make it easier to set parameters because one can see the intermediate results. Therefore, in this sample the same images are used as in the Blob Detection sample [▶ 2780], and the same round structures are searched for.

Variables

```
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;

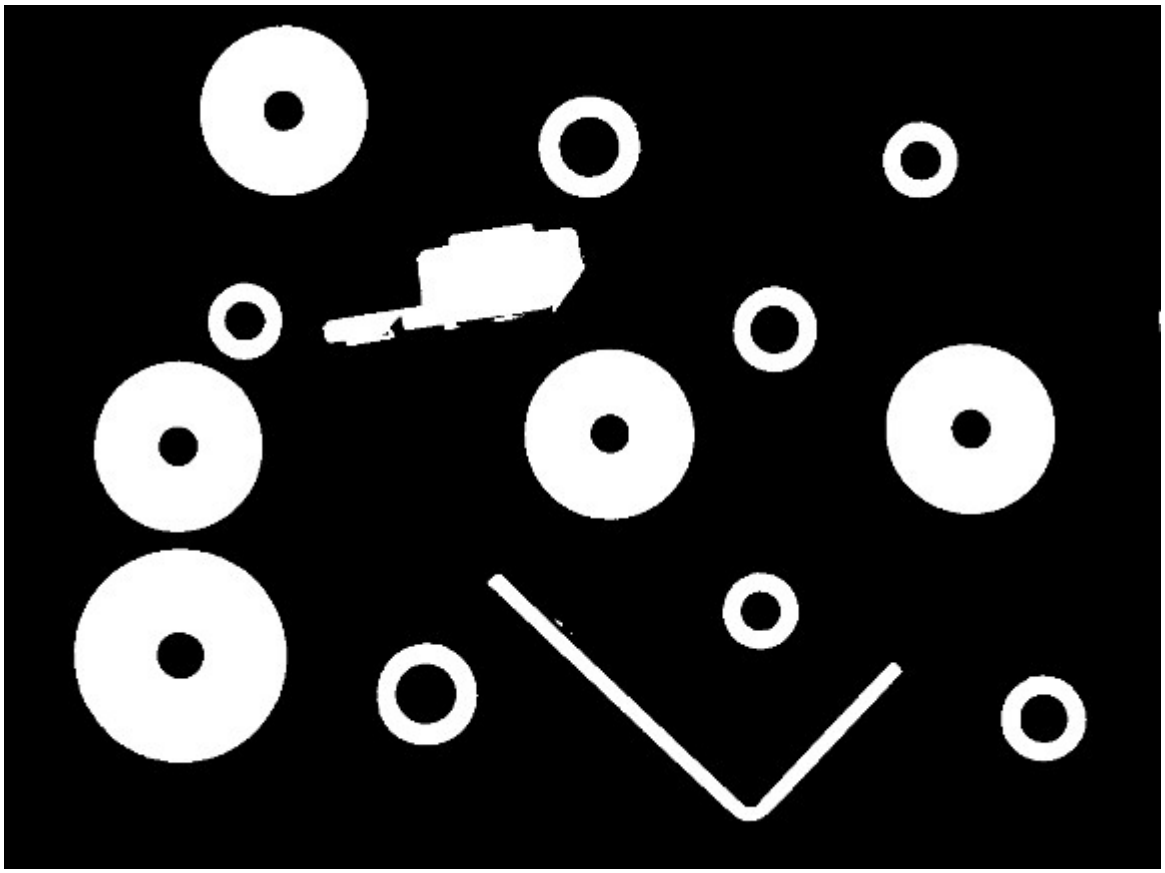
// Image Infos
stPixelFormat           : TcVnPixelFormat;

// COLORS
aColorGreen             : TcVnVector4_LREAL := [0, 255, 0];
```

Code

Preprocessing for image segmentation

```
// Image Segementation
hr := F_VN_Threshold(ipImageIn, ipImageWork, fThreshold, 255, TCVN_TT_Binary, hr);
```



Contour search

```
// Find Contours
hr := F_VN_FindContoursExp(
ipSrcImage              := ipImageWork,
ipContours              := ipContourList,
eRetrievalMode          := ETcVnContourRetrievalmode.TCVN_CRM_LIST,
eApproximationMethod    := ETcVnContourApproximationMethod.TCVN_CAM_SIMPLE,
aOffset                 := aOffset,
hrPrev                  := hr);
```

Filtering of contours according to the surrounding area in pixels and circularity

```
// 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

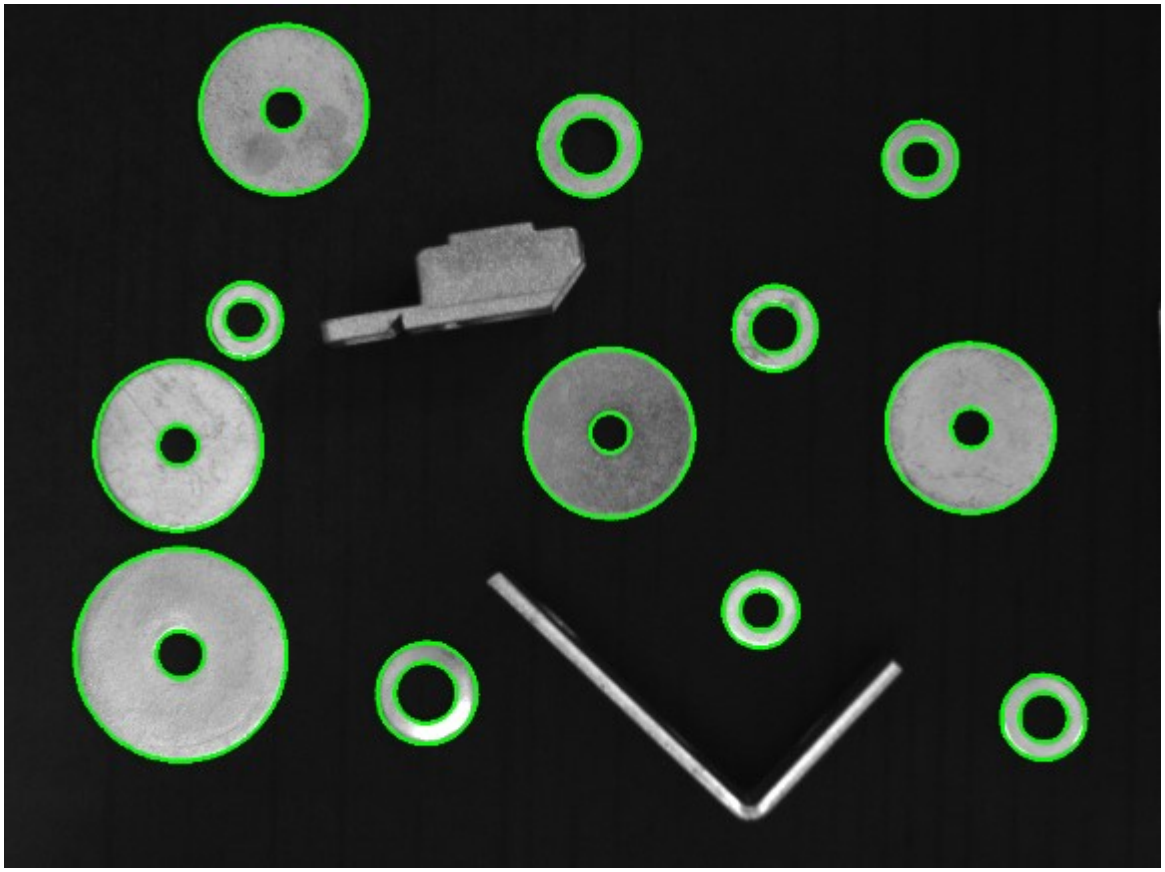
// Filter by Circularity
hr := F_VN_ContourCircularity(ipContour, fCircularity, hr);

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
```

Result output

```
// 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);
```



Similar sample

- Blob Detection with watchdog monitoring [▶ 2780]

8.1.5.1.3 Find Contour - Approximation Method

This sample deals with the possible approximation modes (**eApproximationMethod** of type ETcVnContourApproximationMethod [▶ 188]) for contour search.

Explanation

The functions F_VN_FindContours [▶ 1187], F_VN_FindContoursExp [▶ 1190] and F_VN_FindContourByHierarchy [▶ 1183] are used to detect contours in the image. The contours are described by points. Depending on the selected approximation mode (**eApproximationMethod** of type ETcVnContourApproximationMethod [▶ 188]) all points of each contour or a simplified point set are returned:

- TCVN_CAM_NONE
  All points of the contour are returned
- TCVN_CAM_SIMPLE
  Loss-free compression is applied
- TCVN_CAM_TC89_L1 oder TCVN_CAM_TC89_KCOS
  Both functions apply one of the variants of the Teh-Chin chain approximation algorithm

Variables

```
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);

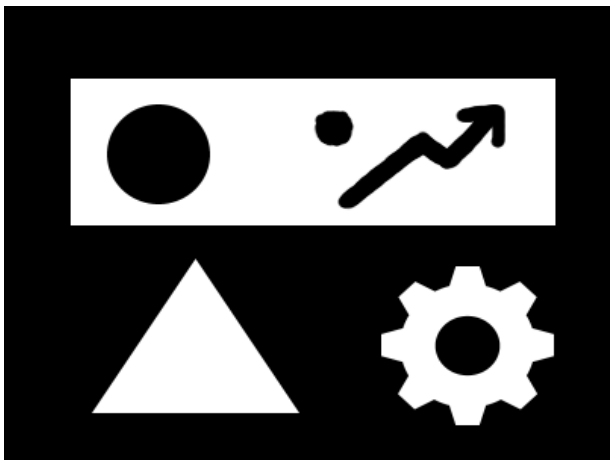
// Find Contours and their Hierarchy
// ---------------------------------
hr := F_VN_FindContoursExp(
                  ipSrcImage             :=   ipImageIn,
                  ipContours             :=   ipContourList,
                  eRetrievalMode         :=   eRetrievalMode,
                  eApproximationMethod   :=   eApproximationMethod,
                  aOffset                :=   aOffset,
                  hrPrev                 :=   hr);

// Draw the points of the contour
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);

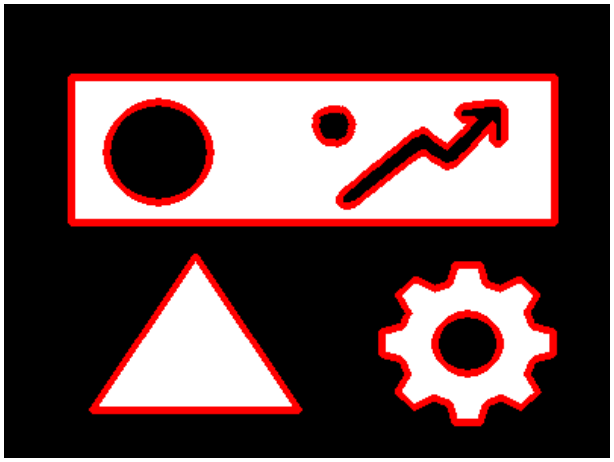
WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
   hr := F_VN_GetContainer(ipIterator, ipContour, hr);
   hr := F_VN_IncrementIterator(ipIterator, hr);
   hr := F_VN_DrawPoints(ipContour, ipImageRes, ETcVnDrawShape.TCVN_DS_CIRCLE, aColorRed, hr);
END_WHILE
```

Results

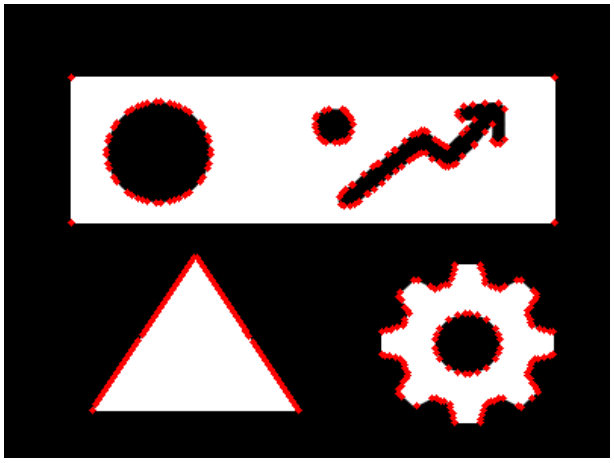
The input image



The result image with all points drawn according to ETcVnContourApproximationMethod [▶ 188].TCVN_CAM_NONE



The result image with all points drawn according to ETcVnContourApproximationMethod [▶ 188].TCVN_CAM_TC89_L1



Similar samples

- Find Contour - Hierarchy & Retrieval Mode [▶ 2791]
- Find contour instead of Blob Detection [▶ 2786]

8.1.5.1.4 Find Contour - Hierarchy & Retrieval Mode

This sample shows the retrieval mode (ETcVnContourRetrievalMode [▶ 191]) for the contour search (F_VN_FindContoursExp [▶ 1190] and F_VN_FindContourHierarchyExp [▶ 1183]) and how it affects the contour result and the returned hierarchy.

Explanation

The contour retrieval mode (**eRetrievalMode** of type ETcVnContourRetrievalMode [▶ 191]) can be used to specify how the contour hierarchy is taken into account during the contour search. This sample visualizes the results according to the selected mode.

Detailed explanations of the hierarchy structure can be found in F_VN_FindContourHierarchyExp [▶ 1183].

Variables

```
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);

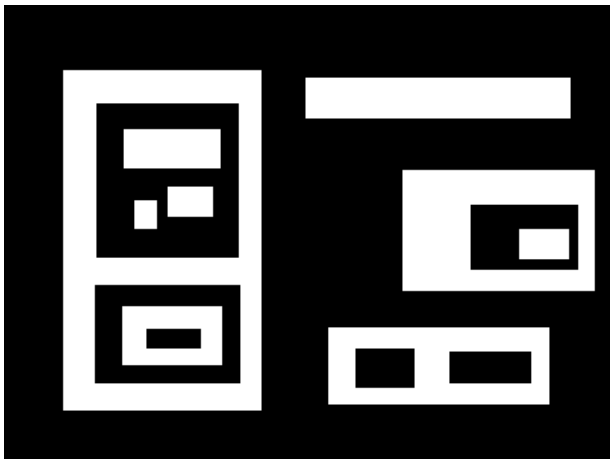
      // Count Parents and set aColor depending on parent number
      nParents := 0;
      aHierarchyTree := aHierarchy;
      WHILE SUCCEEDED(hr) AND_THEN aHierarchyTree[3] >= 0 DO
         nParents := nParents + 1;
         hr := F_VN_GetAt_TcVnVector4_DINT(ipHierarchyList, aHierarchyTree, aHierarchyTree[3], 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);

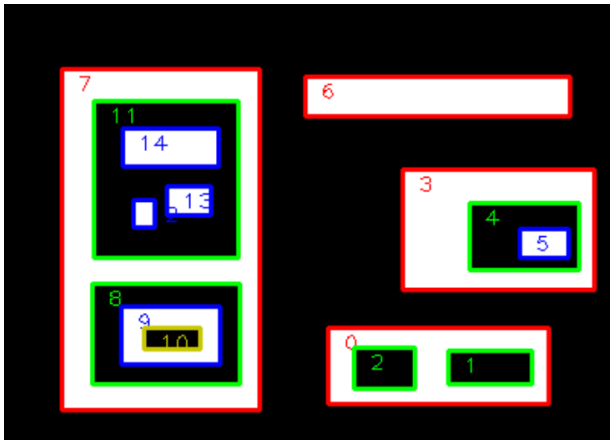
      // Write Hierarchy
      hr := F_VN_PutText(sText, ipImageHierarchy, 10, TO_UDINT(10 + i*10), TCVN_FT_HERSHEY_PLAIN,
0.5, aColorList[(nParents) MOD 4], hr);
      FOR j := 0 TO 3 BY 1 DO
         sText := TO_STRING(aHierarchy[j]);
         hr := F_VN_PutText(sText, ipImageHierarchy, TO_UDINT(10 + (j+1)*22), TO_UDINT(10 + i*10),
TCVN_FT_HERSHEY_PLAIN, 0.5, aColorBlack, hr);
      END_FOR
   END_FOR
END_IF
```

Results

The input image



The contours found with TCVN_CRM_TREE. The colors render contours at the same level



The hierarchy for the above result image. Each row refers to a contour. The columns have the following meaning:

- Contour index ("colored column“)
- Index of the next contour on the same hierarchical level

- Index of the previous contour on the same hierarchical level
- Index of the first child contour
- Index of parent contour

The number -1 indicates that no corresponding contour exists.



Similar samples

- Find Contour - Approximation Method [▶ 2788]
- Find contour instead of Blob Detection [▶ 2786]

8.1.5.1.5 Template Matching with evaluation

In this sample you will perform Template Matching in order to find a component on a printed circuit board. For this purpose, use:

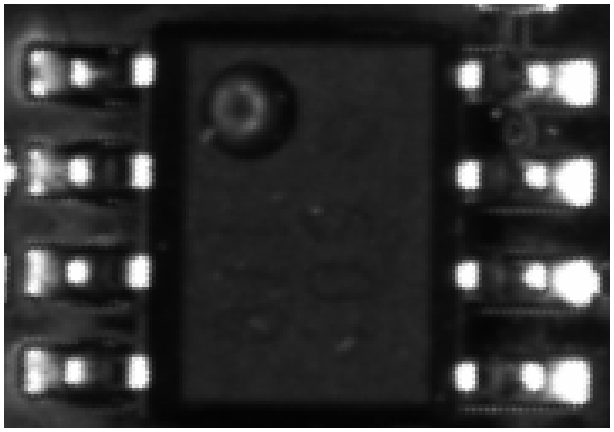
- F_VN_MatchTemplateAndEvaluate [▶ 1200]
- F_VN_DrawRectangle [▶ 1037]

Application

Remember to add both the input image and the template image to the respective File Source. Via the threshold value `fMatchThreshold` you can set a limit as to how good a match must be in order to be accepted.

Program

Important in this sample is that you require a template image in addition to the normal input image in order to search for occurrences of the template in the input image.



If both images are available, perform the template matching automatically with the function F_VN_MatchTemplateAndEvaluate [▶ 1200].

```
hr := F_VN_MatchTemplateAndEvaluate(ipImageIn, ipImageTemplate, ipMatches, fMatchThreshold, hr);
```

The matches found whose degree of correlation is higher than the specified threshold value `fMatchThreshold` are now located as points in the container `ipMatches`. Each point specifies the top left corner of each occurrence of `ipImageTemplate` in `ipImageIn`.

In principle you are now done with the Template Matching. By way of example, access the individual matches in the following in order to paint them into the image. For this purpose, use the construct for accessing container elements as explained in the section Container [▶ 152].

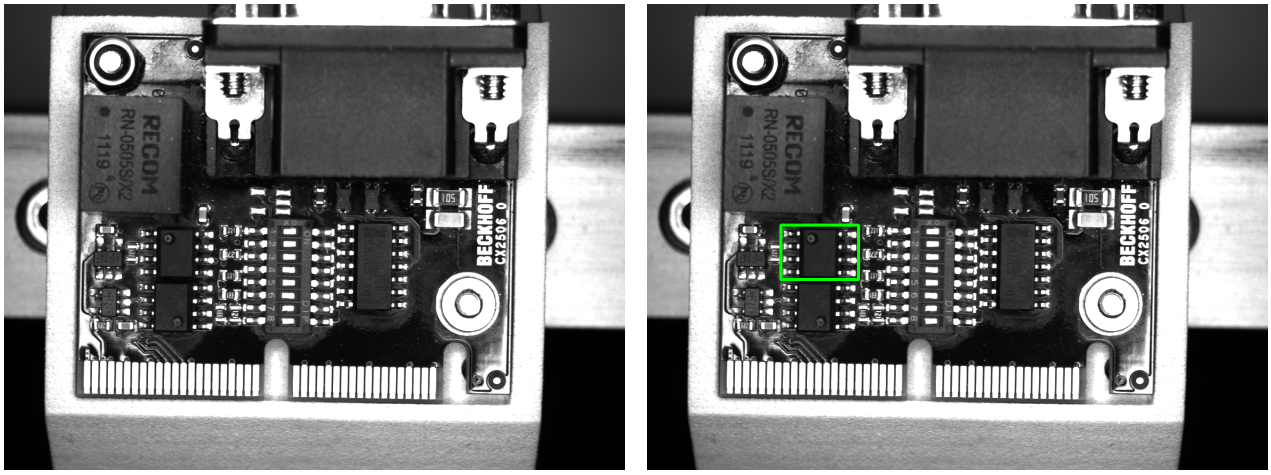
```
hr := F_VN_GetImageHeight(ipImageTemplate, nHeight, hr);
hr := F_VN_GetImageWidth(ipImageTemplate, nWidth, hr);
hr := F_VN_GetAt_TcVnPoint2_DINT(ipMatches, aPosition, 0, hr);
hr := F_VN_GetForwardIterator(ipMatches, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    hr := ipIterator.TcQueryInterface(IID_ITcVnAccess_TcVnPoint2_DINT, 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
```

NOTICE

Long execution duration

Depending on the size of the input and template image, the performance of the Template Matching can take a very long time in comparison with other functions. Make sure that you set the cycle time accordingly and use a watchdog if necessary!

Original image **Result image**



8.1.5.2 Average intensity in shapes of any kind

In this sample the average intensity of an image is calculated within a manually defined area of an image. For this purpose a circular image mask [▶ 157] is used.

The following functions are mainly used:

- F_VN_ImageAverageExp [▶ 1218]
- F_VN_DrawCircle [▶ 1000]

Explanation

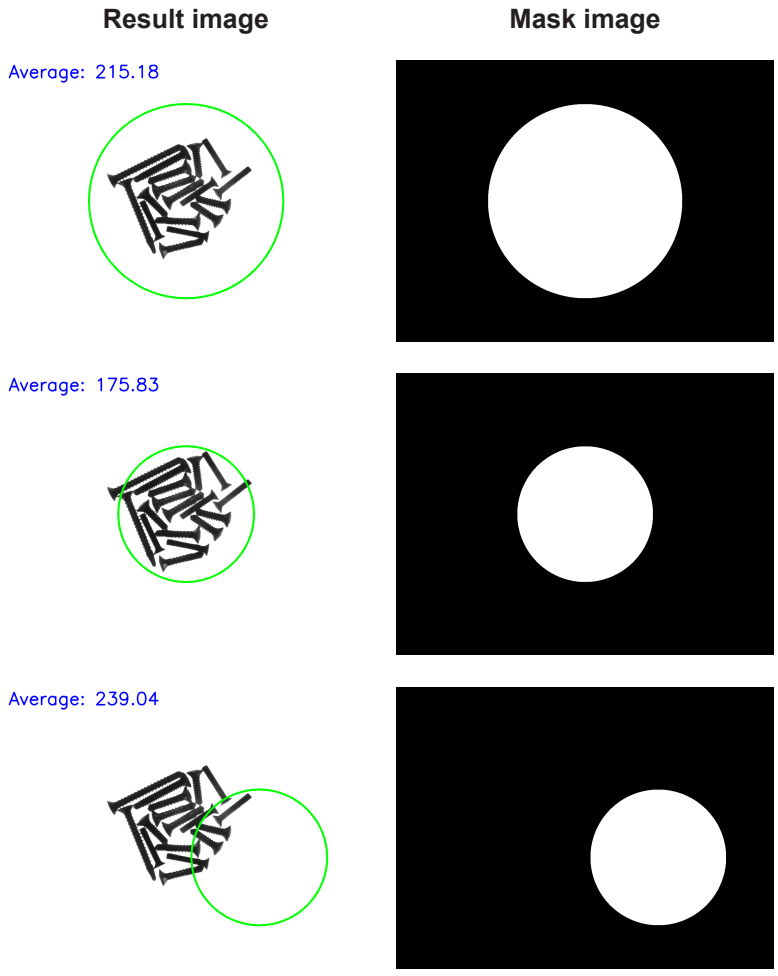
The sample Statistical image features [▶ 2797] shows how statistical image features (such as average, maximum and minimum) can be calculated. Frequently, however, these features are not calculated for the entire image, but only for certain image areas. The setting of an ROI [▶ 158] is helpful if these areas are rectangular. You have to work with image masks [▶ 157] in order to restrict the calculation to areas of any shape. These are essentially images whose intensity values describe which pixels in the original image are to be considered. A mask can thus assume any shape; this sample shows a circular mask.

For this sample a quasi-binary image with a collection of screws will be observed. One conceivable use case would be, for example, that the number of screws is to be approximately determined on the basis of the average intensity value.



Application

By varying the parameters `nCircleShrink` and `aCenterOffset`, you can change the size and position of the circular mask. According to this mask, the average intensity value changes: if the mask becomes smaller and shows less of the white background, the average value decreases. If the mask shifts and shows more of the white background, the average value increases.



Other images can be loaded into this sample if necessary. Note that the mask parameters may then have to be adjusted.

Code

The sample image is loaded via a FileSource state machine as `ipImageWork` and processed with the following code.

First of all the properties of the mask are defined. As the mask is circular, it is described by the center point `aCenter` and the radius `nRadius`. The image center point corrected by the user-adjustable offset `aCenterOffset` is used as the center point of the circle. The radius is calculated as the distance from the center point to the nearest border of the image, minus the value `nCircleShrink`.

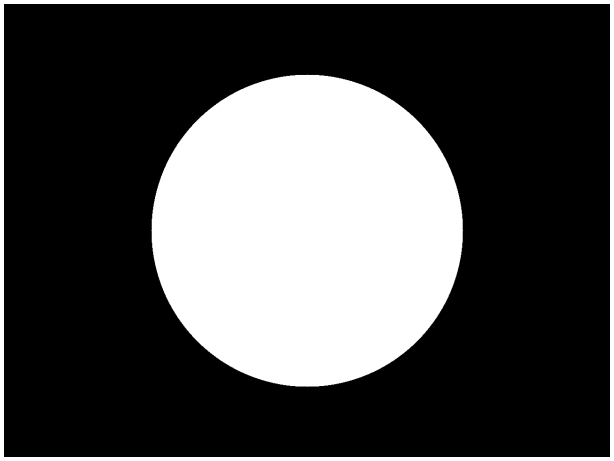
```
hr := F_VN_GetImageWidth(ipImageWork, nWidth, hr);
hr := F_VN_GetImageHeight(ipImageWork, nHeight, hr);
aCenter[0] := (nWidth / 2) + aCenterOffset[0];
aCenter[1] := (nHeight / 2) + aCenterOffset[1];
nRadius := (MIN(nWidth, nHeight) / 2) - nCircleShrink;
```

A new image with the same size as the input image `ipImageWork` is created as the mask and painted with a white circle on a black background. The circle contains the calculated properties. The value `nThickness` is selected as the width `-1` of the border of the circle so that the circle is filled out.

```
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
);
```

The result is that a mask image `ipImageMask` is now available with the desired circular shape:



The average intensity value of the input image `ipImageWork` in the defined circular area is now calculated with the help of this mask `ipImageMask`. The function F_VN_ImageAverageExp [▶ 1218] returns the array `aAverage` of the type `TcVnVector4_LREAL` in order to cover the maximum number of image channels. As the input image in this sample only has one channel, the first array element is the desired average value and is saved separately.

```
hr := F_VN_ImageAverageExp(
    ipSrcImage  :=  ipImageWork,
    aAverage    :=  aAverage,
    ipMask      :=  ipImageMask,
    hrPrev      :=  hr
);
fAverageInMask := aAverage[0];
```

Finally, the mask circle and the calculated average value are drawn in a result image.

```
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);
```

The PLC library Tc2_Utilities is required for the function `LREAL_TO_FMTSTR`.

8.1.5.3 Statistical image features

In this sample, statistical features (min, max, median, average value, standard deviation) of images are calculated. The following functions are used for this:

- F_VN_MinPixelValue [▶ 1229]
- F_VN_MaxPixelValue [▶ 1227]
- F_VN_ImageMedian [▶ 1224]
- F_VN_ImageAverageStdDev [▶ 1219]

Application

This project contains a File Source into which any RGB images can be loaded. These are converted in the program into grayscale images and analyzed with regard to various statistics. The result values are drawn in the image and can be viewed in the ADS Image Watch [▶ 131] under `ipImageRes`.

Program

```
// 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);
```

The PLC library Tc2_Utilities is required for the function `LREAL_TO_FMTSTR`.

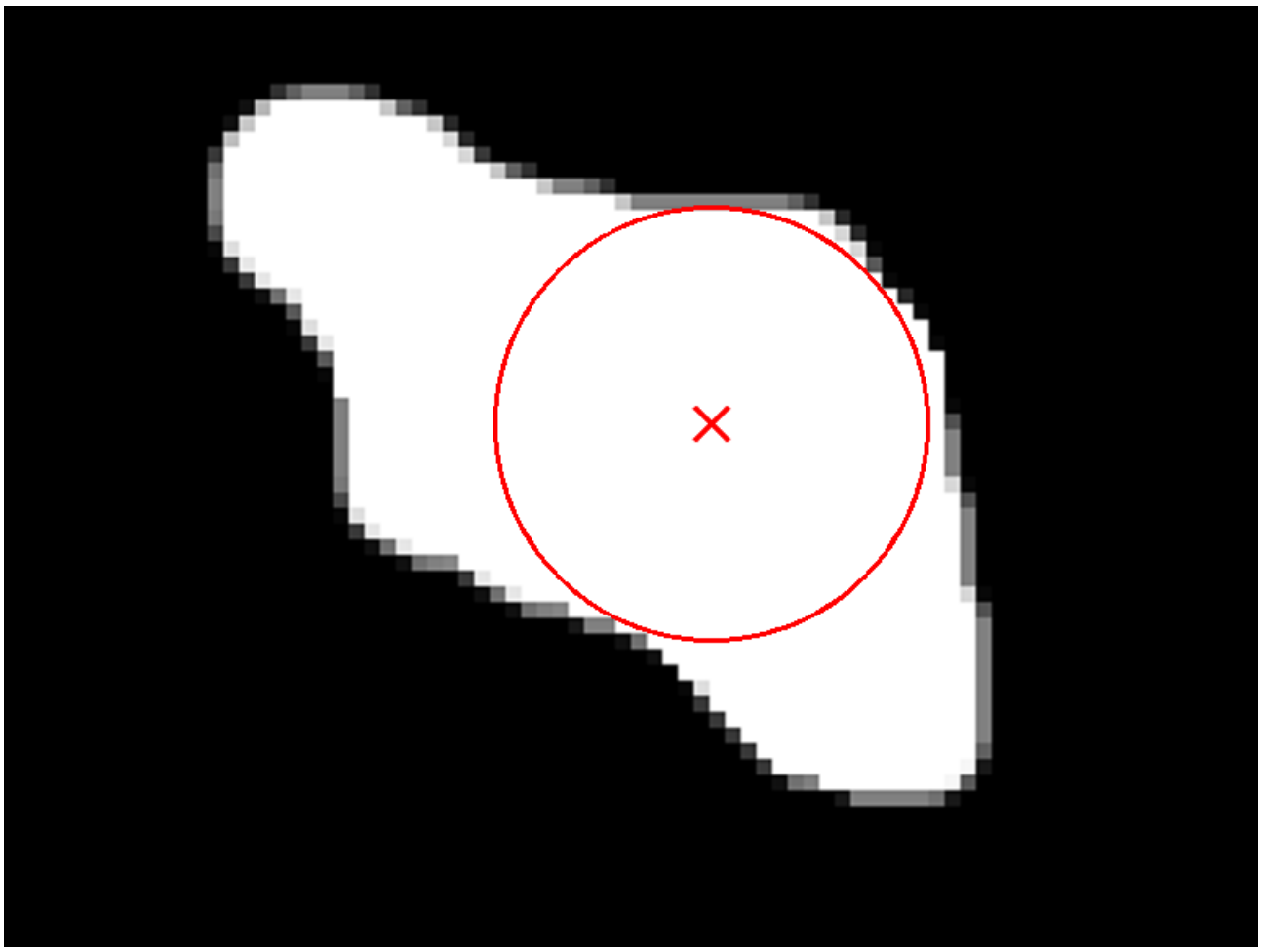
8.1.5.4 Inner circle by means of distance transformation

In this sample you will find the inner circle of a misshapen object by means of distance transformation. For this purpose you essentially use the following functions:

- F_VN_DistanceTransformation [▶ 1213]
- F_VN_DrawCircle [▶ 1000]

Explanation

External contour approximations such as an enclosing rectangle or the convex envelope can be calculated via appropriate API functions. Conversely, an inner circle with a maximum size inside a contour can be found by means of a distance transformation.



For this purpose, the point with the greatest Euclidean distance to the object edges is determined with a distance transformation. This point is then the center point of the inner circle of the object.

If you use this sample with your own images, you should adjust the threshold value and the binarization via the variables `bInvert` and `fThreshold`. By means of the parameters `eDistanceType` and `eMaskSize` you can purposefully change the parameters of the function F_VN_DistanceTransformation [▶ 1213].

Program

The first step is to binarize the input image. This achieves a clear distinction between object and background. The binarization is important for the distance transformation, as the distance transformation only reacts to pixels with the value 0. If the object is black on a white background, it must be inverted by means of the threshold type `BINARY_INV`.

```
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);
```

Through the distance transformation you obtain an image in which each pixel value indicates the distance to the next pixel with a value of 0. The pixel with the highest value is thus the pixel with the greatest distance. If you use the L2 norm (Euclidean distance) as the distance type `eDistanceType`, you could interpret this pixel as the center point of a circle that fills out the object as much as possible. Therefore, determine the pixel with the maximum value using the function F_VN_MaxPixelValue [▶ 1227] and use its position as the center point of the circle and its value as the radius of the circle.

```
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;
```

The mask size `eMaskSize` determines the accuracy of the distance calculation and has little influence on the result.

Finally, paint the circle in the result image on the basis of the determined center point and radius in order to be able to validate the execution.

```
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 with RGB reference color

In this sample, the function F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1260] calculates the similarity of each pixel to 3 reference colors and returns them as grayscale images. The objects in the reference colors are then detected on the basis of these grayscale images.

Variables

```
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);

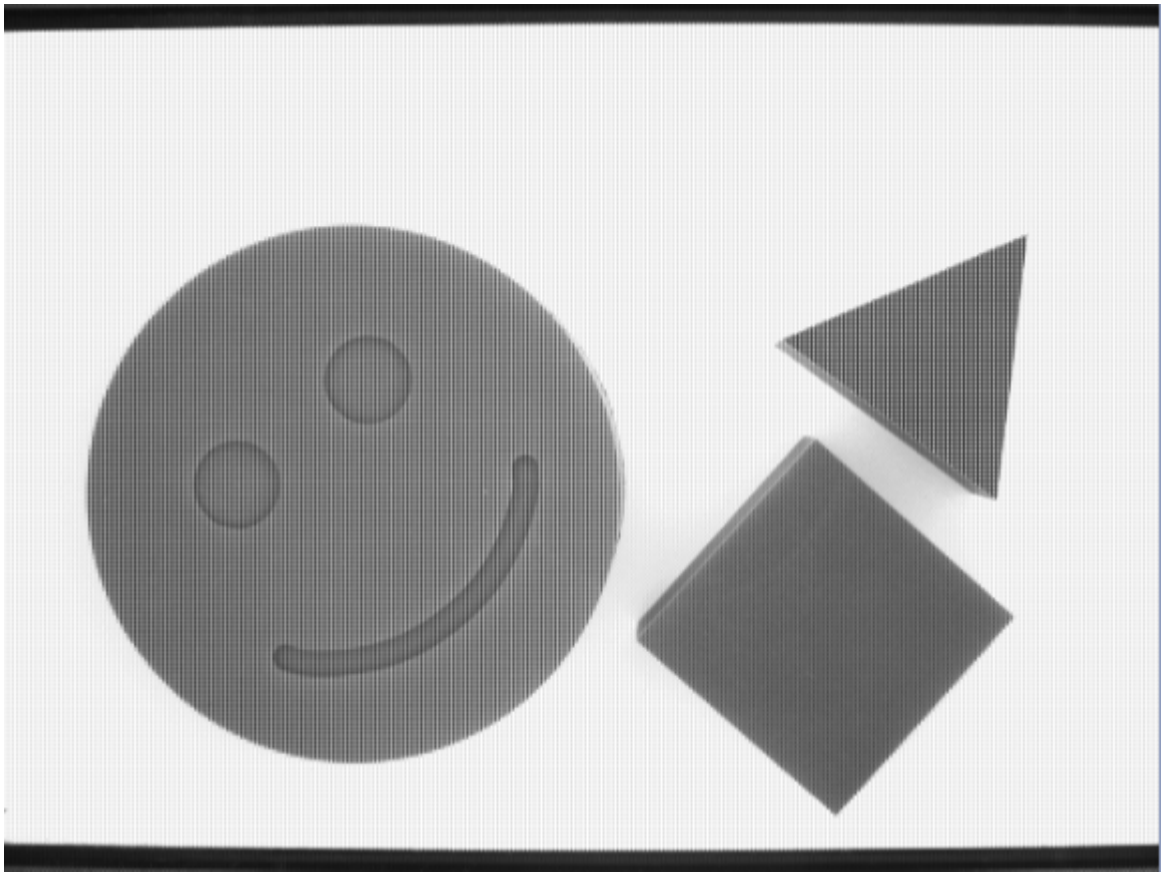
   // Filter the objects by size and draw the contours
   WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
      hr := F_VN_GetContainer(ipIterator, ipContour, hr);
      hr := F_VN_IncrementIterator(ipIterator, hr);

      // 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
```

Result

The input image in Bayer RG format



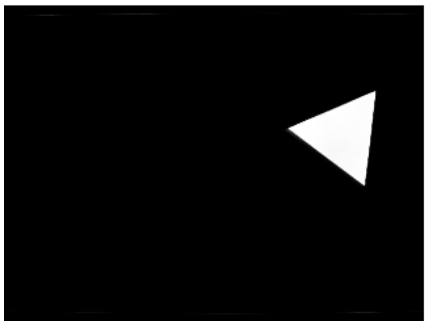
After converting the input image to an RGB image with the function F_VN_ConvertColorSpace [▶ 1241], the F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1260] can be used to determine the color similarity of each pixel compared to the reference color. The following applies in the returned result image:

- The pixel value 255 (white) corresponds to a 100% match with the reference color

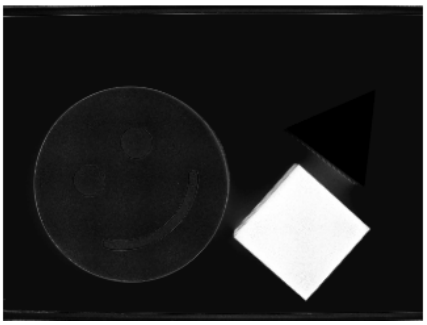
- The pixel value 0 (black) corresponds to a 0% match with the reference color
- and the pixel values between 0 and 255 a partial correlation to the reference color.

```
// 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)



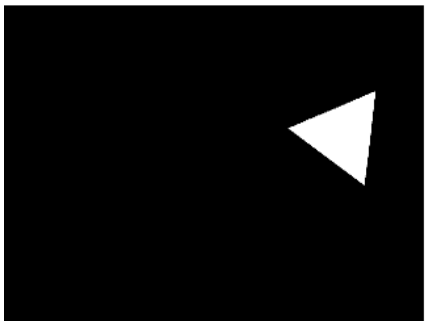
aColorRef := [40,140,190]; (BLUE)



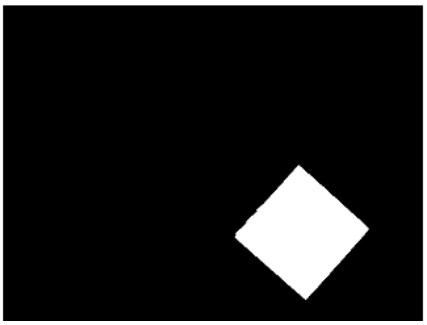
A threshold value (F_VN_Threshold [▶ 1334]) is used to define how close the match to the reference color should be in order to be assigned to it.

```
hr := F_VN_Threshold(ipImageWorkCol, ipImageThres, 200, 255, TCVN_TT_Binary, hr);
```

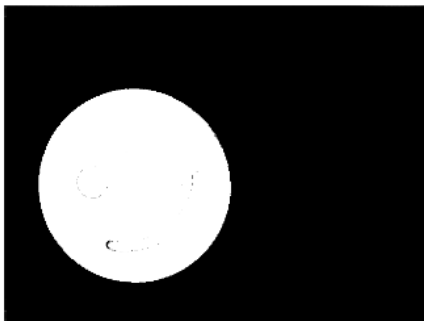
aColorRef := [255,75,60]; (RED)



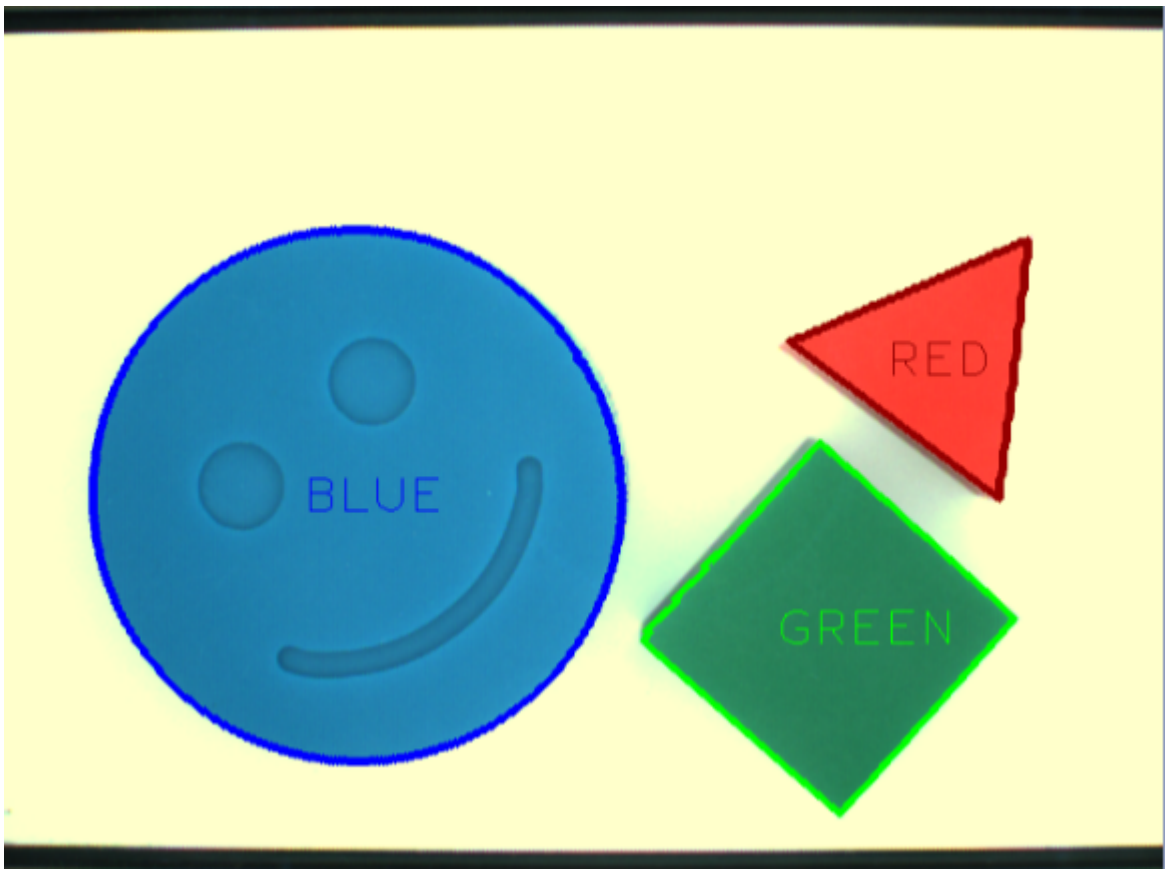
aColorRef := [40,140,95]; (GREEN)



aColorRef := [40,140,190]; (BLUE)



The result image, after subsequent detection.



Similar sample

- Check Color Range with RGB range [▶ 2810]

8.1.6.2 Display with color tables

In this sample you apply color tables to the grayscale images. You essentially use the following functions:

- F_VN_GenerateColorMap [▶ 1241]
- F_VN_CreateContainerFromArray [▶ 754]
- F_VN_GenerateCustomColorMap [▶ 1243]
- F_VN_ApplyColorMap [▶ 1236]

Explanation

There are several reasons why you would want to apply a color table to a monochrome image. These include, for example, a more pleasant appearance and a display of the value range that is clearer to the human eye. This is above all relevant for 16-bit images, as the value spectrum is significantly more recognizable if it is distributed over three color channels.

Depending on the use case, you would like to use various color tables. TwinCAT Vision provides some predefined color gradients via the enum ETcVnColorMap [▶ 170]. You can also define your own color gradients and create color tables with them.

In this sample you apply various color tables to the following grayscale gradient (with a value range of 16 bits).



Fig. 26: Grayscale gradient

Of course, you can also use your own images. Note that you can only use 1-channel images, or you will have to perform an appropriate conversion beforehand using F_VN_ConvertColorSpace [▶ 1241].

Program

Before a color table can be applied to an image, it must first be created. To this end, it must be known whether the color table is to be for 8-bit or 16-bit images, as depending on that it will require a different number of entries (see Color tables). Accordingly, you must set the parameter `eColorMapSize` of the type ETcVnColorMapSize [▶ 175] to `256` in the case of 8-bit and `65536` in the case of 16-bit to match the bit depth of the input image `ipImageIn`. You achieve this by querying the element size of the pixel format by means of F_VN_GetImageInfo [▶ 797]:

```
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
```

A color table is generated on the basis of a color gradient. You can either manually define a color gradient or use one from a template. This sample deals with the distinction on the basis of the parameter `bUseCustomColors`.

```
IF bUseCustomColors THEN
    // create color map from custom-defined colors
    <...>
ELSE
    // create color map from preset
    <...>
END_IF
```

You define a color gradient manually by creating a container with several entries of the type TcVnVector3_REAL. Each entry represents a color. The colors are interpreted as being evenly distributed over the definition area when creating the color table; all other values are linearly interpolated for each of the three channels. A color gradient from the array `aColors` with the three colors red, green and blue is used here. You then create the color table with the function 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
);
```

In order to use a predefined color gradient, use the function F_VN_GenerateColorMap [▶ 1241] and set the parameter `eColorMap` to the desired color gradient from the enum ETcVnColorMap [▶ 170]. In this case this is initially set to `JET`.

```
VAR
    eColorMap: ETcVnColorMap := TCVN_CM_JET;
END_VAR

hr := F_VN_GenerateColorMap(
    ipColorMap      :=  ipColorMap,
    eColorMap       :=  eColorMap,
    eColorMapSize   :=  eColorMapSize,
    hrPrev          :=  hr
);
```

The color table created is a container [▶ 152] that stores a color to be issued for every possible value in the input image. Therefore you can also create a color table yourself by defining all 256 or 65536 entries and then directly creating a container out of them. In most cases, however, the use of the API functions is simpler.

Finally, apply the color table created to the input image to obtain the desired result. The result image is always a 3-channel 8-bit image.

```
hr := F_VN_ApplyColorMap(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImage,
    ipColorMap      :=  ipColorMap,
    hrPrev          :=  hr
);
```

The application of a predefined JET color table to the input image shown above produces the following result:

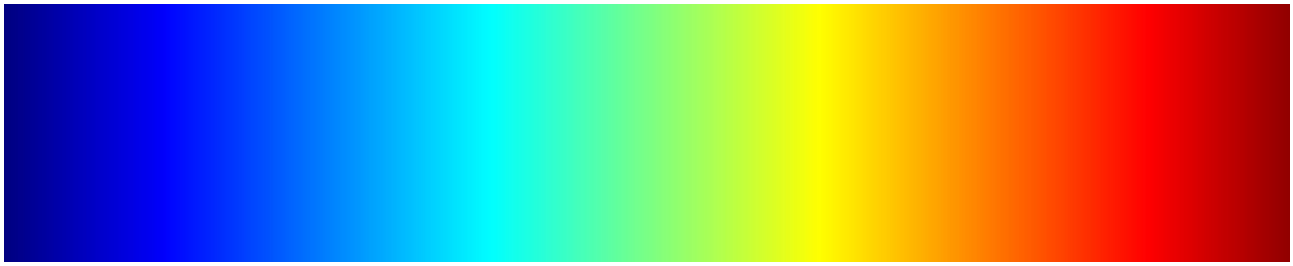


Fig. 27: Color gradient as result of the application of the JET color table

8.1.7 Image filtering

8.1.7.1 Blur filter

In this sample the following blur filters are compared:

- Bilateral filter [▶ 1276]
- Gaussian filter [▶ 1295]
- Median filter [▶ 1306]
- Mean filter implemented as custom filter [▶ 1287]

Explanation

In some cases an image is to be smoothed or freed from noise. For tasks such as Code Reading it is necessary at the same time to obtain a good contrast and good edges on the image so that the downstream algorithms can successfully work on it.

The **bilateral filter** is particularly suitable for the combination of image smoothing with the simultaneous retention of edges. For each pixel intensity it calculates a weighted average value of the surrounding pixels. The weighting depends not only on how far away the pixel is, but also on the difference in intensity. As a result, sharp edges are retained.

The **Gaussian filter** smooths images with the help of a two-dimensional discrete approximation to the Gaussian bell function. This allows the image noise to be reduced. Smaller structures are lost, but coarser structures on the other hand are retained. Edges are therefore less blurred than with the two following filters, but larger disturbances are not removed so well.

The **median filter** sorts all pixel values of the specified mask in ascending order and then selects the median value, to which the corresponding pixel is then set. This filter is well suited for suppressing structures (disturbances) that are smaller than the applied filter mask and is very robust towards outliers. As a result, the small salt-and-pepper disturbances, for example, are removed best, but the disadvantage of this is that the other detailed information is also lost.

The **custom filter** function was used in order to form a mean value. Here, the value of each pixel is replaced by the mean value of the kernel pixel, which results in blurring. Disturbances (noise) are suppressed as a result, but details and edges gradually become blurred as the number of kernels increases. As the custom filter function offers many different options for applying your own filters, the `F_VN_InitMatrixStruct` function is additionally required for the standardization of the filter kernel.

Variables

```
// 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);

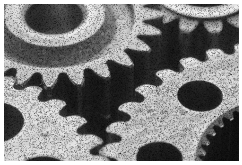
// Mean-Filter with Custom Filter Function
hr := F_VN_InitMatrixStruct(
        pSrcBuffer      := ADR(aMatrixArray7x7),
        stDestMatrix    := stKernelMatrix,
        nRows           := 7,
        nCols           := 7,
        eElementType    := TCVN_ET_REAL,
        hrPrev          := hr);

hr := F_VN_CustomFilter(
        ipSrcImage      := ipImageIn,
        ipDestImage     := ipImageCustom,
        eDestDepth      := TCVN_ET_USINT,
        stKernel        := stKernelMatrix,
        hrPrev          := hr);
```

Result

The unprocessed original image (1st row) already exhibits fine structures on the surface of the gear wheels in the detail. In order to illustrate the effects of the filters, additional disturbances were added to the original image by Gauss noise (2nd row) and salt-and-pepper noise (3rd row).

So that the effects of the different filters are as comparable as possible, a kernel size of 7 x 7 is used with all of them. Using other parameter values or kernel sizes, as well as further setting options for the EXP variants of the functions, other effects can be achieved in detail. As the filter effect very much depends on the content of the input image, it is recommended to test several filters with different parameter settings in order to compare them directly and find the best filter for the use case. The table below contains an overview of the result images:

Original images	Bilateral filter	Gaussian filter	Median filter	Mean filter
				
				
				

NOTICE

Long or fluctuating execution times

The duration of the execution of a filter essentially depends on the type of filter as well as the image and kernel size. Therefore, please note that a filter operation can take considerably longer than other functions, so you should adapt the cycle time of your task accordingly.

If the filter parameters are changed during the runtime, in particular the kernel size, it is recommended to use watchdogs [▶ 146]. In the case of the median filter, the runtime can always fluctuate somewhat due to the principle.

Furthermore, job tasks [▶ 64] can lead to a reduction in the total computing time, which should be tested according to the system.

8.1.7.2 Edge detection filters

In this sample the following edge detection filters are compared:

- Sobel filter [▶ 1321]
- Scharr filter [▶ 1313]
- Laplace filter [▶ 1299]

Explanation

The Sobel filter is a simple linear filter for marking gradients in an image. The Scharr filter is very similar to this; only the characteristic of the filter kernel is somewhat different. The Scharr filter has a better isotropy, i.e. the detection of edges is independent of the direction.

The Laplace filter detects edges based on the zero points of the discrete 2nd derivation of the image. This variant is susceptible to noise by comparison.

Variables

```
// 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

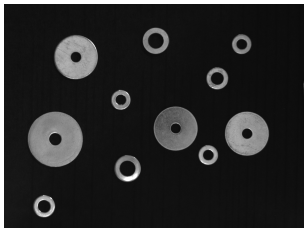
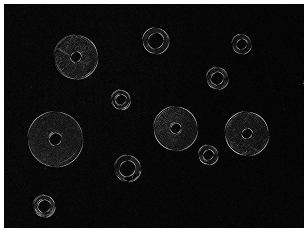
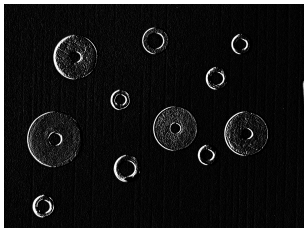
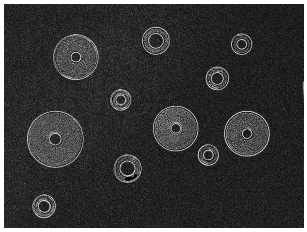
```
// Execute the Sobel Filter
hr := F_VN_SobelFilterExp(
          ipSrcImage    := ipImageIn
          ipDestImage   := ipImageSobel,
          eDestDepth    := eSobel_DestDepth,
          nXOrder       := aSobel_DerivOrder[0],
          nYOrder       := aSobel_DerivOrder[1],
          nKernelSize   := 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);
```

Results

The sample project contains the functions calls of the three filters. The parameter configuration and its effect on a sample image can be tested in this sample.

Original image	Sobel filter	Scharr filter	Laplace filter
			

8.1.7.3 Morphological operators

In this sample you apply morphological operators to an image. For this you use:

- F_VN_CreateStructuringElement [▶ 1285]
- F_VN_MorphologicalOperator [▶ 1309]

Explanation

Objects in an image can be changed according to certain rules with a morphological operator. Morphological operators are mostly applied to binary images; however, application to grayscale images is generally also possible. Through combined operators such as `OPENING` and `CLOSING` it is possible, for example, to supplement missing parts in objects and small outliers can be removed.

Variables

```
ipElement   :  ITcVnImage;
eSEShape    :  ETcVnStructuringElementShape := TCVN_SES_RECTANGLE;
aSESize     :  ARRAY [0..1] OF UDINT := [29, 29];
eOperator   :  ETcVnMorphologicalOperator := TCVN_MO_OPENING;
```

Program

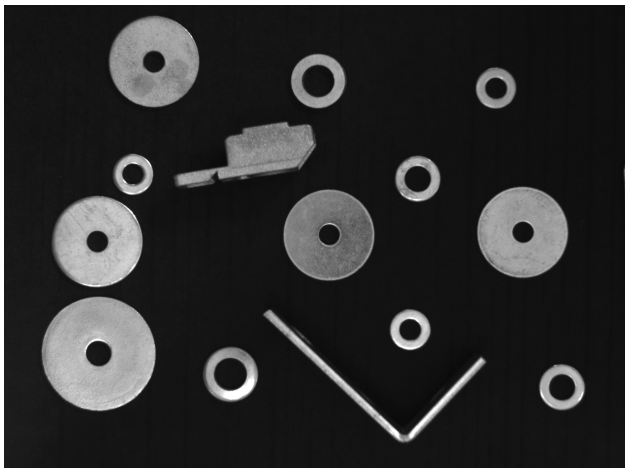
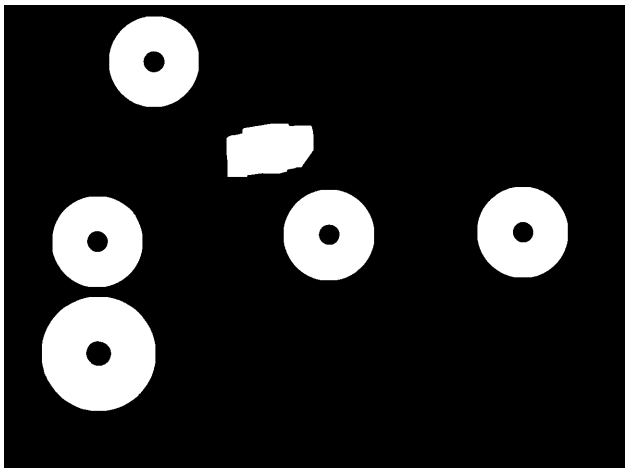
A morphological operator requires a structure element in order to define its area of influence for each pixel. For this purpose you can select various shapes from the enum ETcVnStructuringElementShape [▶ 217] and in different sizes.

```
hr := F_VN_CreateStructuringElement(
    ipStructuringElement    :=  ipElement,
    eShape                  :=  eSEShape,
    nWidth                  :=  aSESize[0],
    nHeight                 :=  aSESize[1],
    hrPrev                  :=  hr);
```

You then define a morphological operation by means of the enum ETcVnMorphologicalOperator [▶ 208] and apply it with the structure element created to the input image.

```
hr := F_VN_MorphologicalOperator(
    ipSrcImage              :=  ipImageIn,
    ipDestImage             :=  ipImageRes,
    eOperator               :=  eOperator,
    ipStructuringElement    :=  ipElement,
    hrPrev                  :=  hr);
```

A morphological opening with a rectangular structure element of the size [31; 31] and previous threshold value, for example, produces the following result. It removes smaller or narrower objects from the image.

Input image	Result image
	

NOTICE

Long execution duration

Depending on the size of the input image and of the structure element, the execution of the morphological operator can take a very long time in comparison with other functions. Make sure that you set the cycle time accordingly and use a watchdog if necessary!

8.1.8 Image segmentation

- Check Color Range with RGB range [▶ 2810]

8.1.8.1 Check Color Range with RGB range

In this sample, the function F_VN_CheckColorRange [▶ 1331] searches for red, green and blue objects in the image and labels them according to their color. The function is applied to RGB images. It is also possible to apply the function to images in other color spaces such as HSV, Lab and BGR, in which case the parameterization must be adapted accordingly.

Variables

```
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);

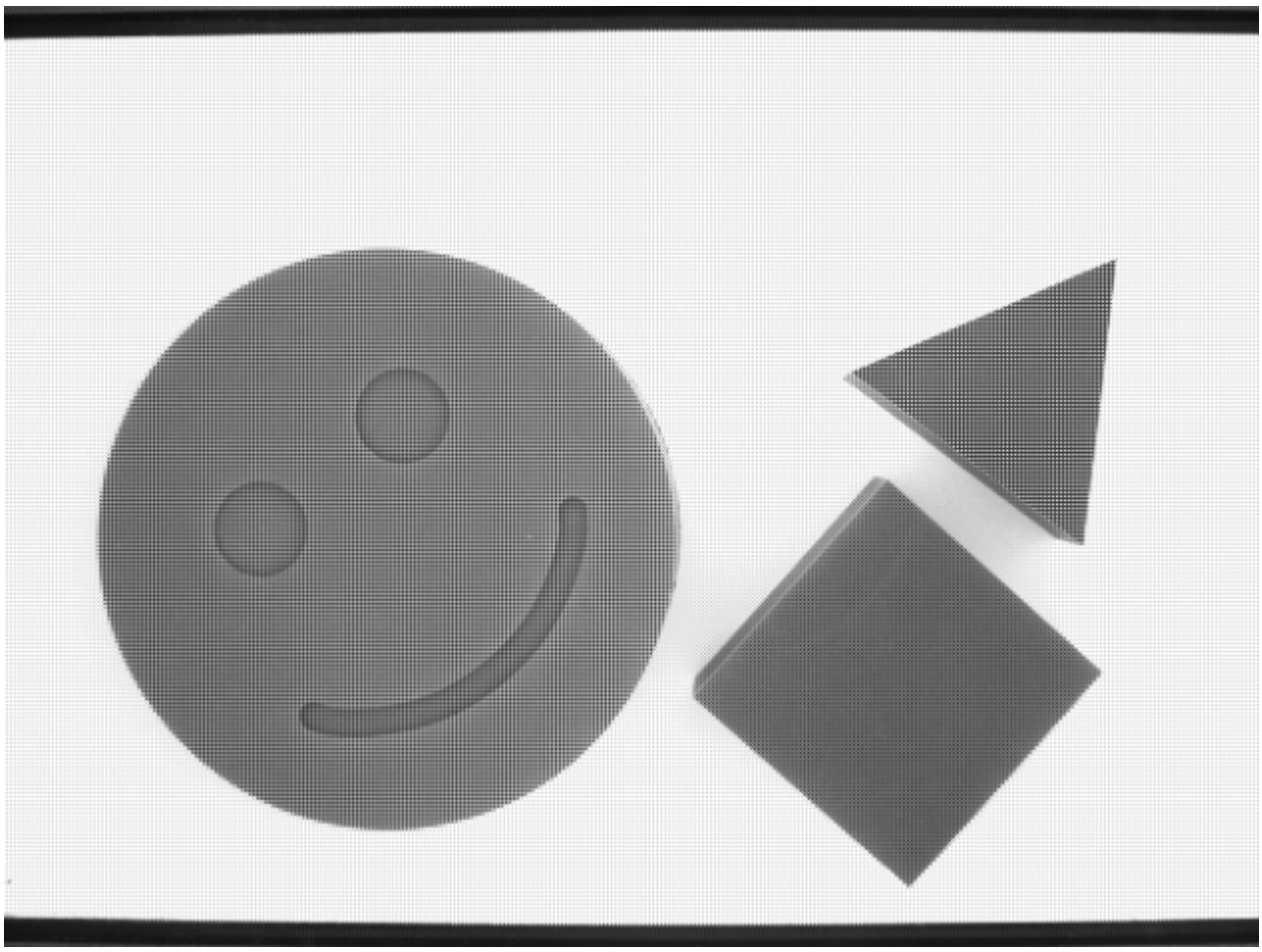
      IF fArea > 5000 THEN
         // Draw Results into an Image
         hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColor[iColor], 3, hr);
         hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
         hr := F_VN_PutText(aColorTxt[iColor], ipImageRes, LREAL_TO_UDINT(aCenter[0])-30,
LREAL_TO_UDINT(aCenter[1])+10, TCVN_FT_HERSHEY_PLAIN, 2, aColor[iColor], hr);
      END_IF

   END_WHILE

END_FOR
```

Result

The input image in Bayer RG format.



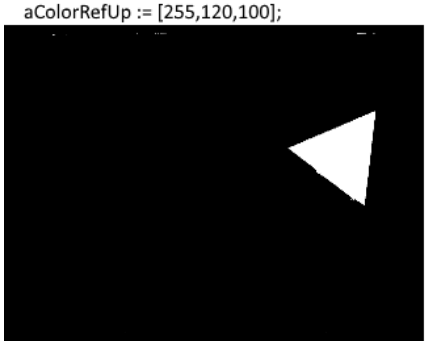
The input image is converted into an RGB image with the function F_VN_ConvertColorSpace [▶ 1241].

```
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);
```

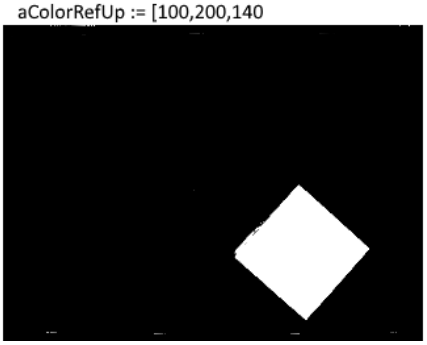
The function F_VN_CheckColorRange [▶ 1331] can then be used with the respective upper and lower threshold values.

```
hr := F_VN_CheckColorRange(ipImageRes, ipImageWorkCol, aColorRefLow[iColor], aColorRefUp[iColor],
hr);
```

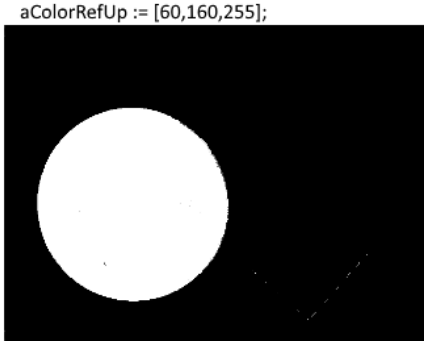
Red
aColorRefLow := [150,50,20];
aColorRefUp := [255,120,100];



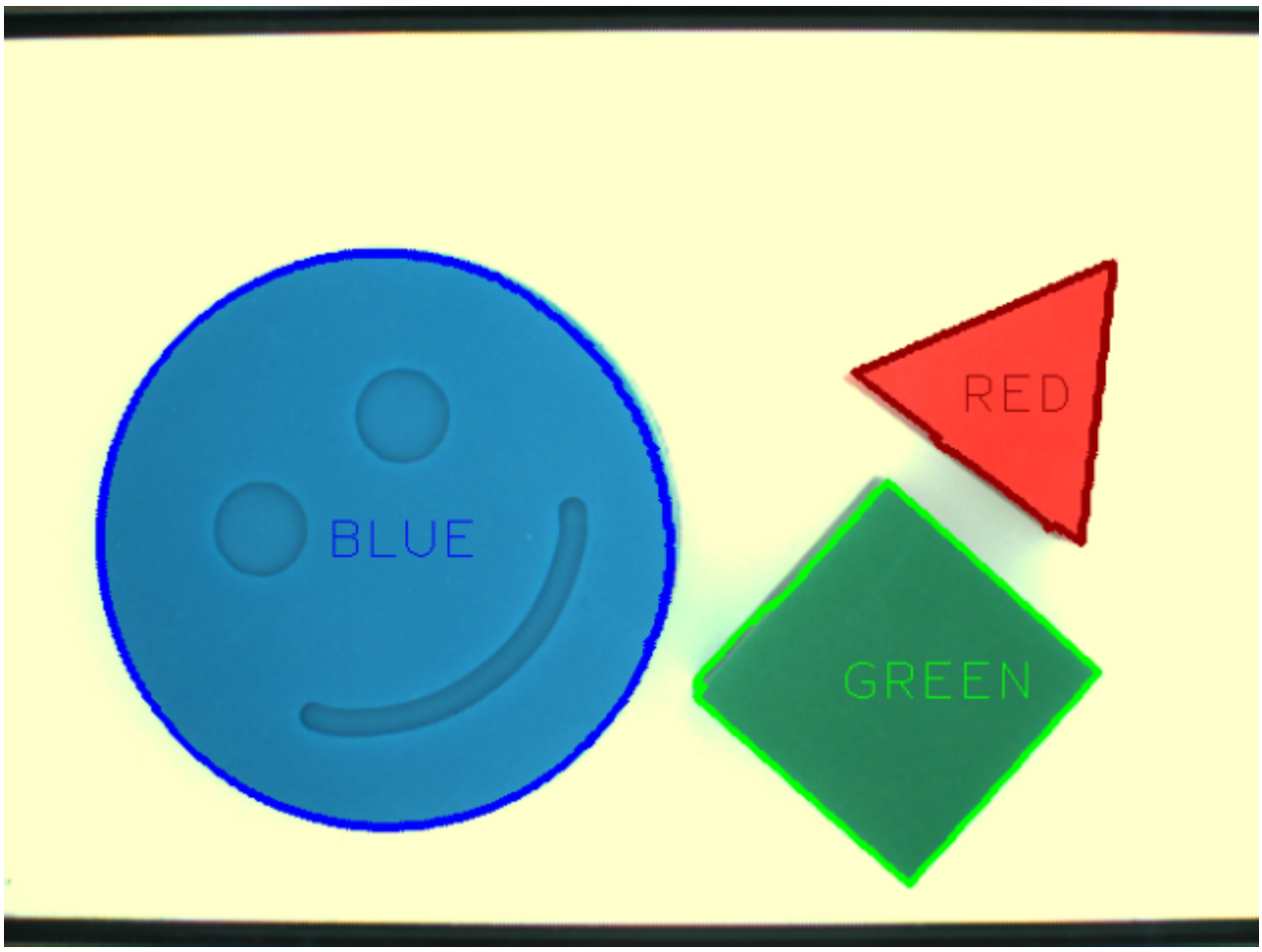
Green
aColorRefLow := [35,90,60];
aColorRefUp := [100,200,140



Blue
aColorRefLow := [20,40,130];
aColorRefUp := [60,160,255];



The result image, after subsequent detection



Similar samples

- Color similarity with RGB reference color [▶ 2800]

8.1.9 Measurement

8.1.9.1 Locate Circular Arc

In this example, the function F_VN_LocateCircularArc [▶ 1489] is used to locate and measure circular arcs. This function returns a TcVnCircularArc structure containing the center, radius and bounding angles as result. To measure complete circles, use the function F_VN_LocateEllipse [▶ 1507] instead.

The start position of the search should correspond approximately to the expected circle center of the circular arc, which is 400, 300 in the corresponding sample images.

To demonstrate that an approximate position is sufficient here, the corresponding parameter aCenter in the sample was set to 420, 310. The sample illustrates that it is possible to compensate small positional changes of the objects to be examined in the image.

Since the actual radius in the sample images is about 200 pixels and the starting point is deliberately inaccurate, the search radius fRadius was set to 270. In the specific sample, around 230 pixels would have been sufficient, but in practice it is always good to have an additional safety buffer, e.g. if the position of the object deviates more than expected.

For the sample images, the search direction must be selected so that the objects are found correctly in all images. A good direction would be down left, around 2,356 rad (135°). To illustrate that this direction does not have to be exact, fDirection was set to 2.1 rad (120.3°) in the sample.

In addition, a transition from light to dark, starting from the starting point, should be specified as a search criterion. The minimum contrast also has to be specified, above which the function detects an intensity difference between adjacent pixels as an edge.

In case these parameters are not sufficient to achieve good results, the expert function F_VN_LocateCircularArcExp [▶ 1492] offers a number of additional parameters, which are not considered in this sample.

Variables

```
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);
```

Results

For visualization, the circular arc that is found is drawn into the result image using the function F_VN_DrawCircularArc [▶ 1004]. The underlying numerical values are also displayed as text.

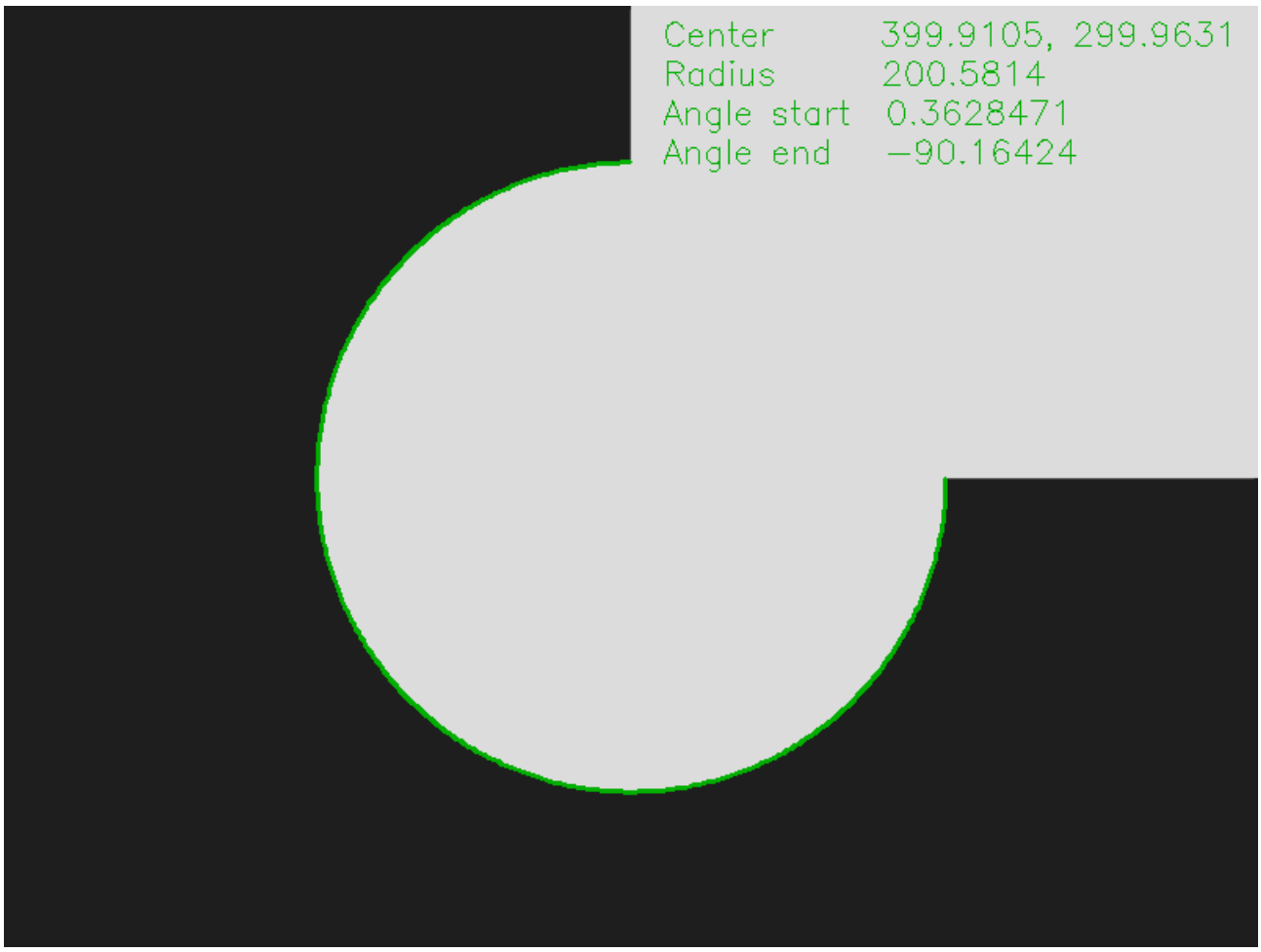


Fig. 28: LocateCircularArc_3QuarterCircle_Result

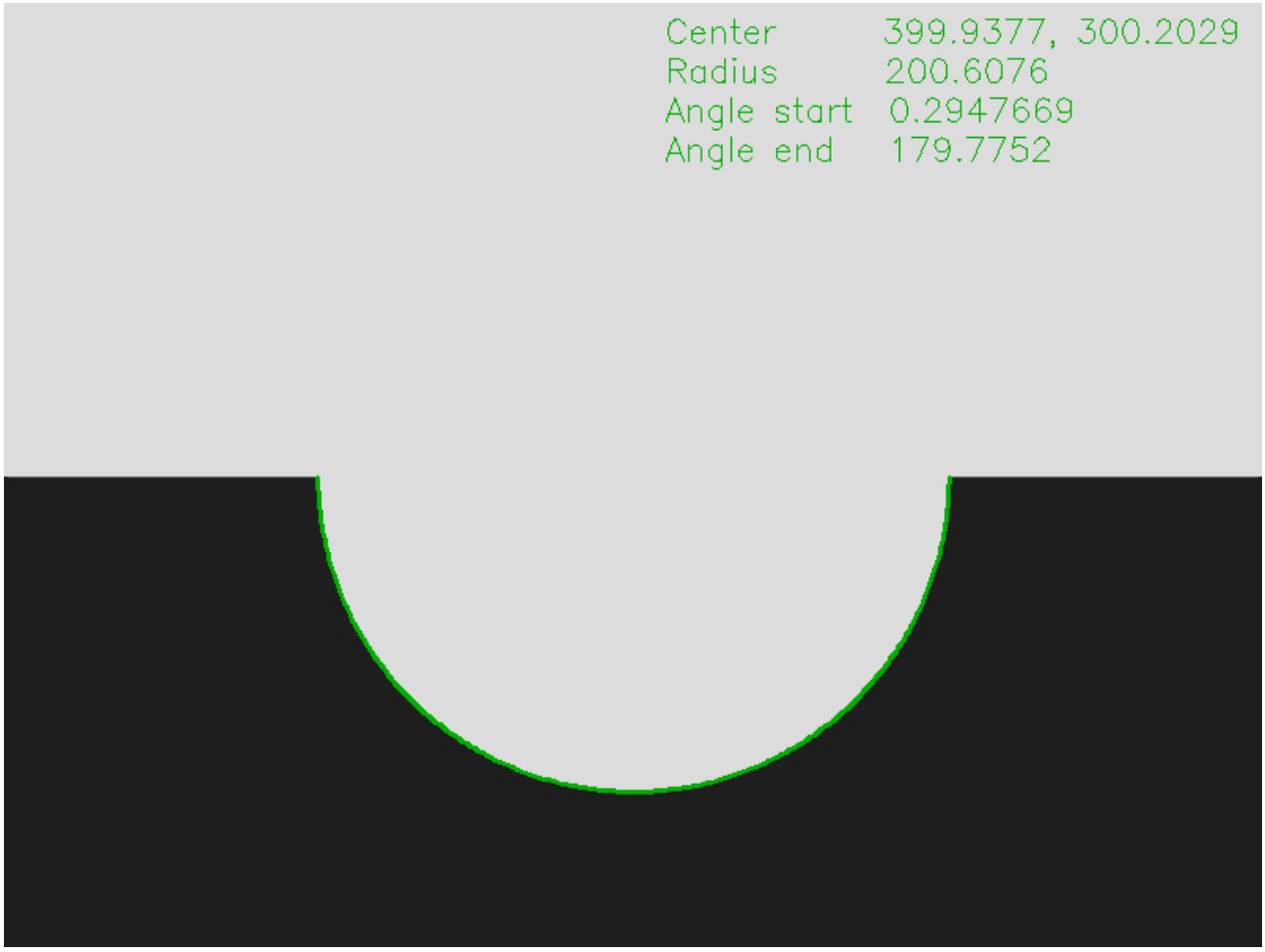


Fig. 29: LocateCircularArc_HalfCircle_Result

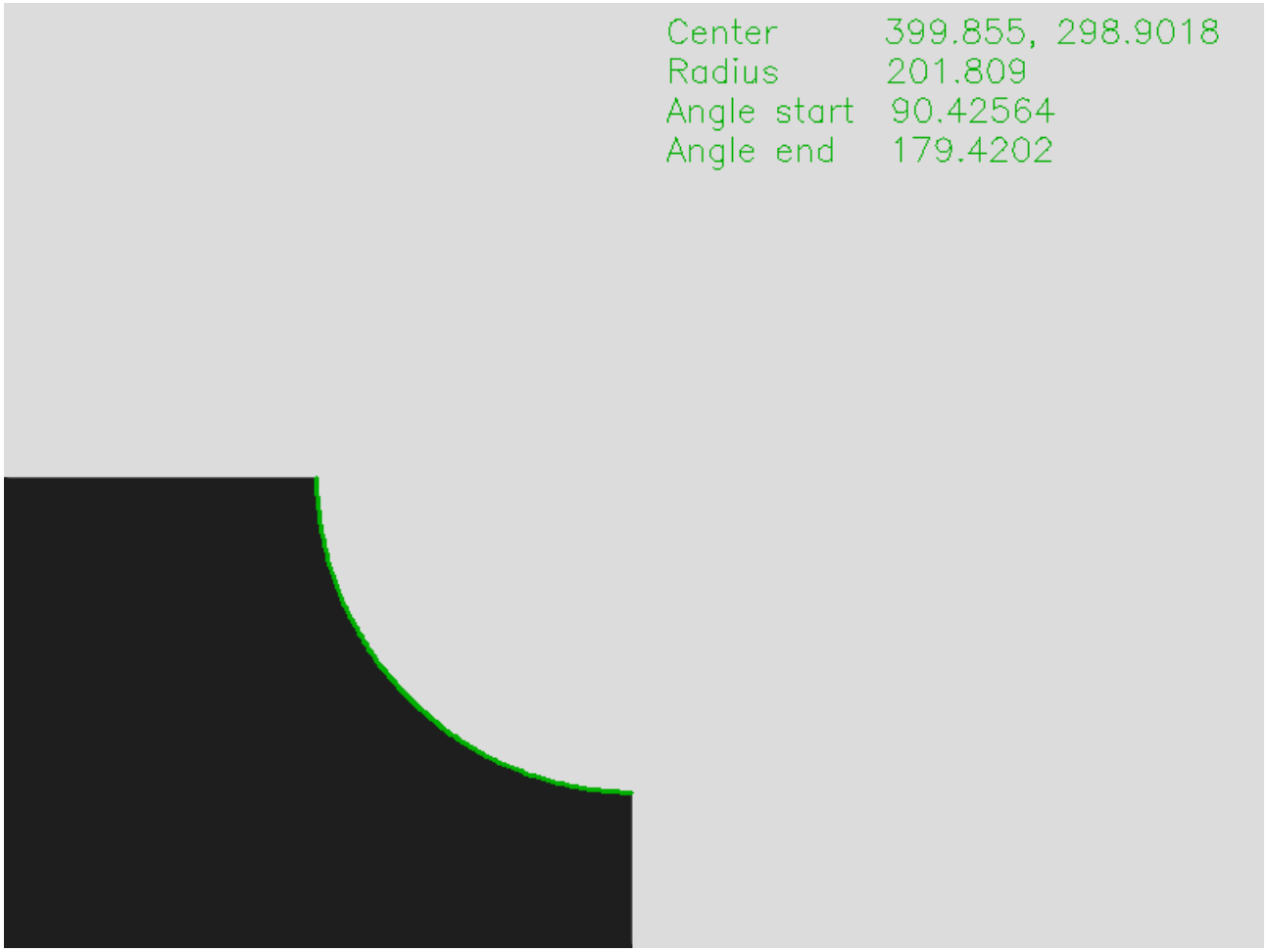


Fig. 30: LocateCircularArc_QuarterCircle_Result

Similar samples

- Locate Edge [▶ 2817]
- Locate Ellipse [▶ 2822]
- Measure Angle Between Edges [▶ 2827]

8.1.9.2 Locate Edge

In this sample,

- the function F_VN_LocateEdgeExp [▶ 1498] is used to locate an edge in a defined search window,
- and the execution time is monitored with a watchdog and limited if necessary.

Explanation

A search window can be used to locate edges with the function F_VN_LocateEdge [▶ 1495]. Some parameters are fixed for simplicity. Alternatively, the function F_VN_LocateEdgeExp [▶ 1498] that is used here provides full access to all parameters. This sample is intended to illustrate the individual parameters. We recommend trying out configuration changes in addition to the standard configuration and consider the effects on the edge result and on the processing time. A selection of configuration changes including descriptions can be found under Results.

Variables

```
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

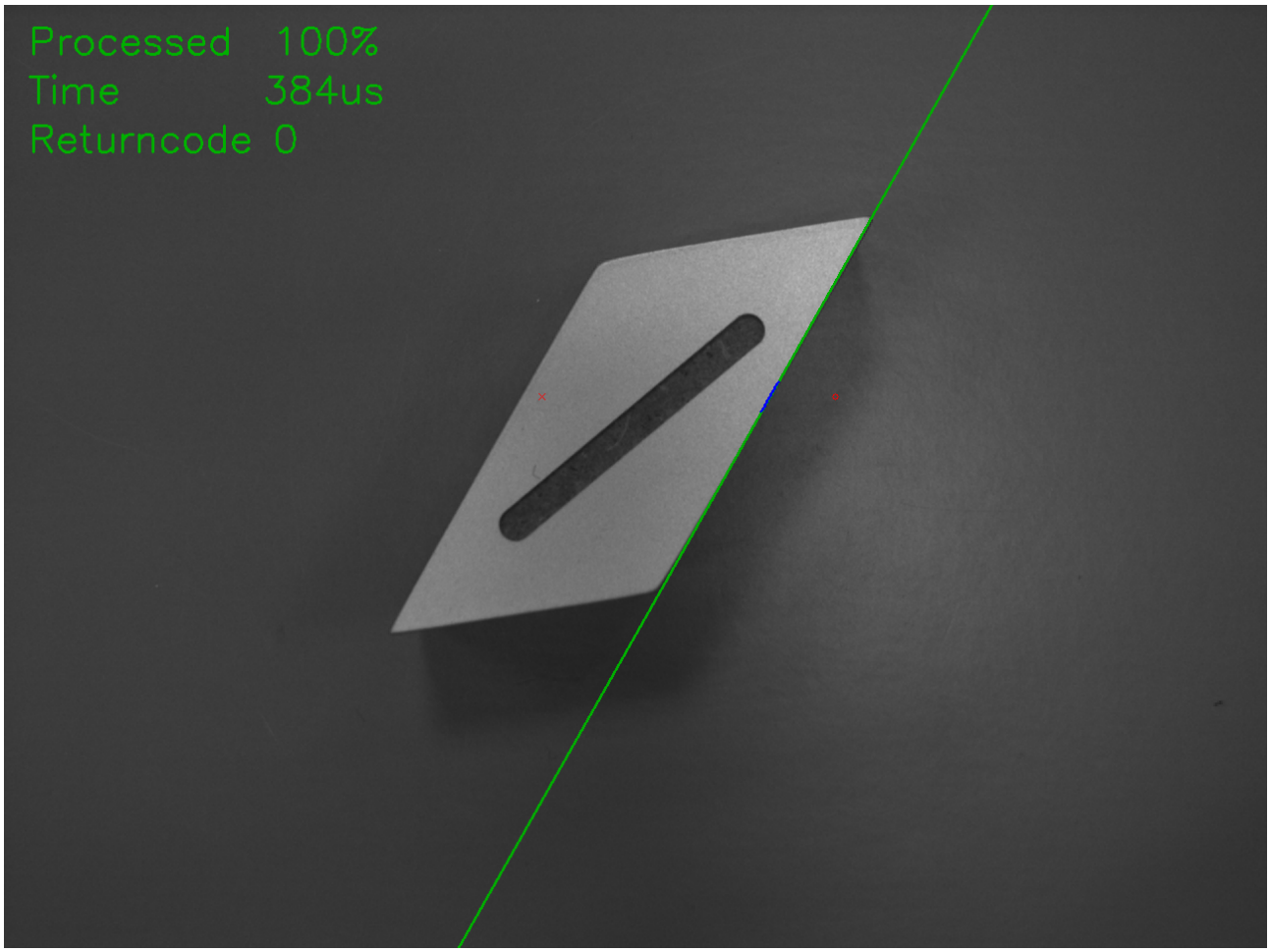
```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
   hrFunc := F_VN_LocateEdgeExp(
                  ipSrcImage             :=   ipImageIn,
                  ipEdgePoints           :=   ipEdgePoints,
                  aStartPoint            :=   aStartPoint,
                  aEndPoint              :=   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);

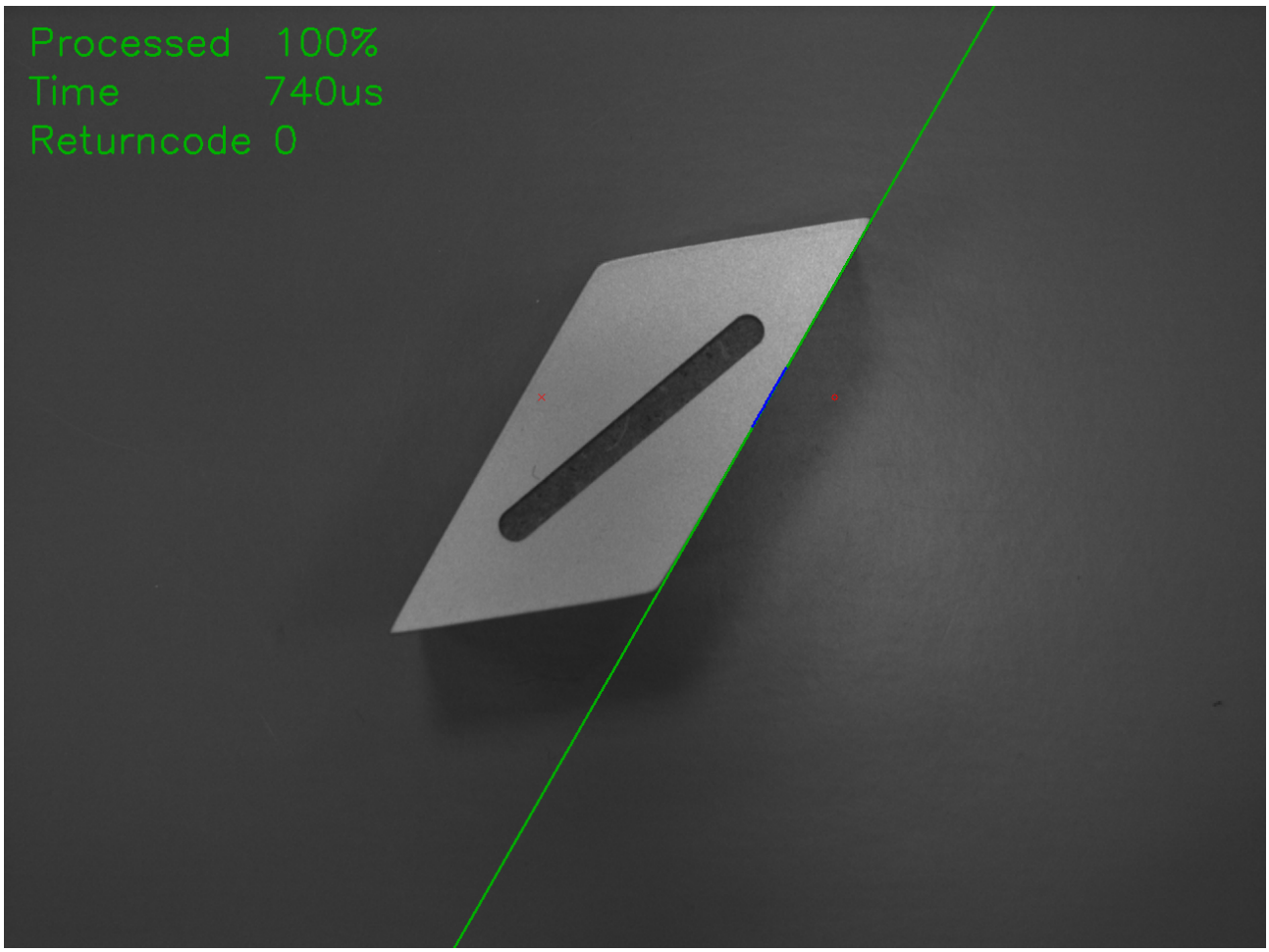
// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

Results

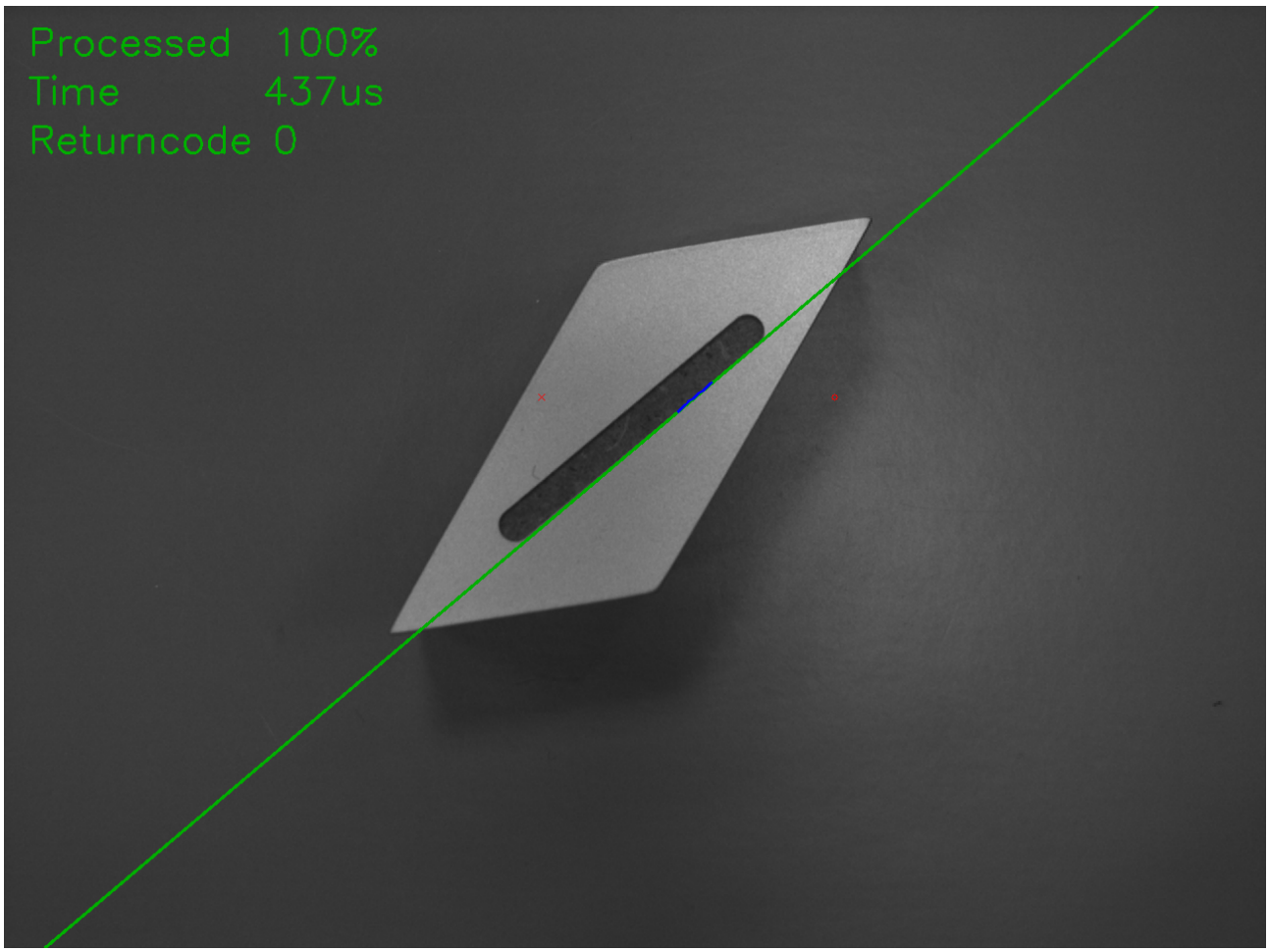
For visualization, the edge points that are found are shown in blue, and a regression line is drawn in green in the output image. In addition, the starting point (red circle) and the end point (red X) are drawn. The processing percentage, the required execution time and the return value of the function are displayed in the top left corner.
For the parameters used in this sample, the result looks like this:



If the number of search lines is increased from 31 to 61 with the parameter `nSearchLines`, the required computing time is also doubled. In return, the regression line becomes more precise:



If you want to find the milling edge of the component instead of the outer edge, change the parameter `eDirection` to `TCVN_ED_LIGHT_TO_DARK`. This ignores the outer edge transition from dark to light and instead finds the following edge with the transition from light to dark:



Reducing execution time

It is noticeable that the execution time, despite otherwise identical parameters, is higher than when locating the outer edge. This is because the edge that is found is further away from the starting point.

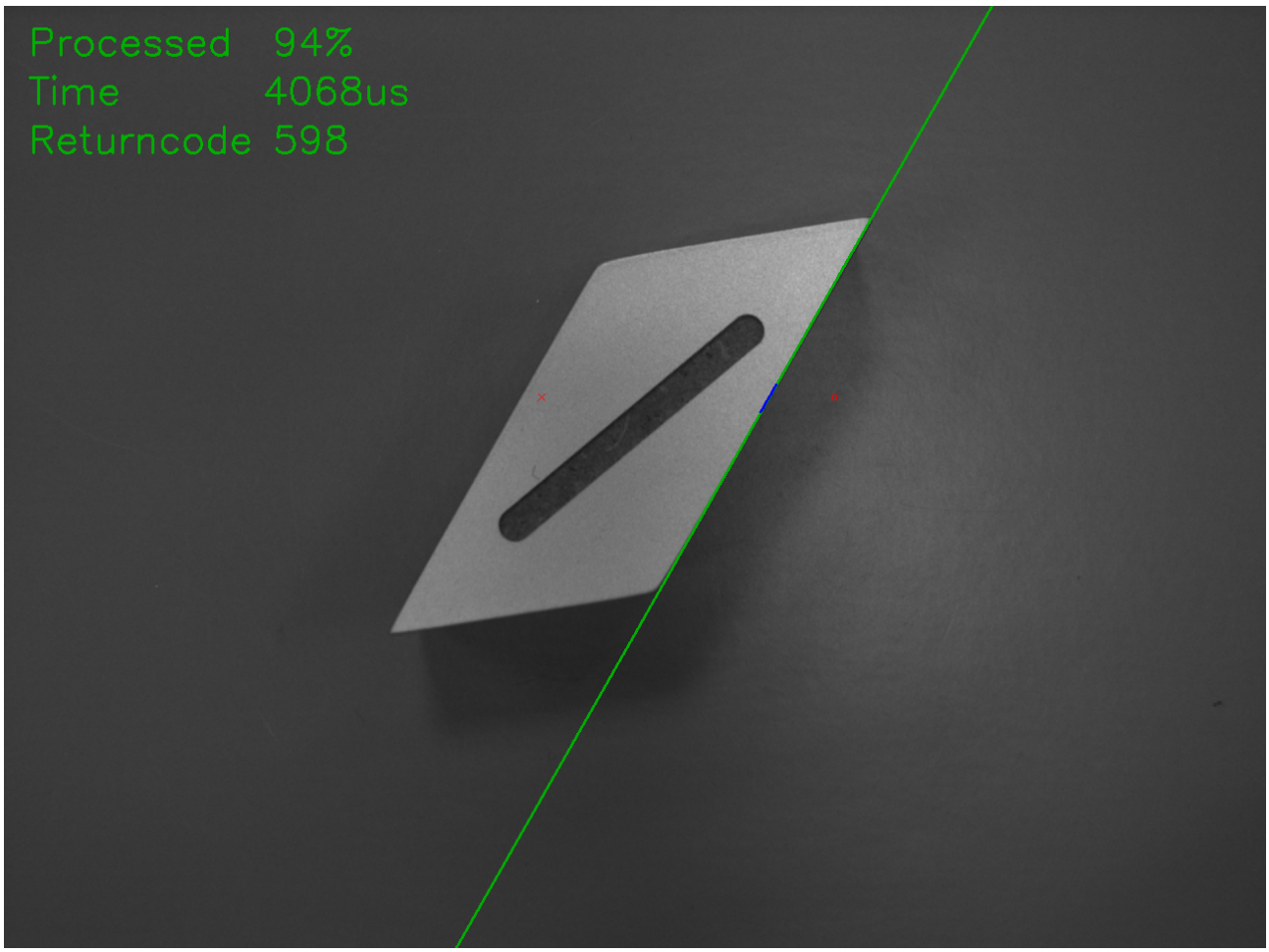
Conversely, this means that the execution time for a fixed search window varies slightly depending on the position of the component in the image, which can also be seen in the three sample images.

An even greater fluctuation can occur with the two approximation algorithms, as they require different iterations to approximate the model parameters using the surrounding pixel intensities.

In general, fewer iterations are required if the search lines of the search window hit the edge as orthogonally as possible. If the position and orientation of the objects in the image are subject to little or no fluctuation, the search window can be adjusted accordingly to achieve shorter execution times. In addition, the maximum time required can be reduced by reducing the maximum number of iterations, but this may lead to less accurate results.

Using a watchdog

In addition, the time can be limited with an external watchdog, which terminates the execution of the function if necessary and returns the existing partial results. For example, if the algorithm is changed to `TCVN_EDA_APPROX_ERF` and the maximum number of iterations is set to 60, the execution for the image LocateEdge2.bmp (outer edge orthogonal to the search lines) will take almost exactly 4 ms (depending on the installed CPU), while the other two images will take 4.3 and 4.4 ms. If the parameter `tStop` is set to 4000, the execution of the function is stopped as soon as possible after 4 ms, and the partial results available up to then are returned. In other words, the function on the image LocateEdge2.bmp continues to be processed 100%, whereas the function on the other two images (in this case LocateEdge3.bmp) is terminated prematurely:



Since it is only possible to abort at certain points in the algorithm, and the partial results (previously found edge points on the search lines) are returned, the function is not terminated after exactly 4000 µs but takes a little longer, which must be taken into account when selecting the abort time. The maximum additional time required generally depends on the algorithm and the parameterization.

Despite the termination, the partial results are sufficient to achieve the desired result, since only the results of one or two search lines are missing.

Similar samples

- Locate Circular Arc [▶ 2813]
- Locate Ellipse [▶ 2822]
- Measure Angle Between Edges [▶ 2827]

8.1.9.3 Locate Ellipse

In this sample,

- the function F_VN_LocateEllipseExp [▶ 1510] is used to locate a circular object in a defined search area
- and the execution time is monitored with a watchdog and limited if necessary.

Explanation

For a basic understanding of the measurement functions and their parameters, please follow the Locate Edge sample [▶ 2817] first. The function F_VN_LocateEllipseExp [▶ 1510] differs from F_VN_LocateEdgeExp [▶ 1495] in that it uses a circular search window instead of a rectangular one. In this sample a large, inaccurate search window is parameterized, since the position of the circular objects changes slightly. The circles can be determined with the same search window in all images.

The parameter `bInvertSearchDirection` which was set to `TRUE` in this sample, was used to define that the search direction is reversed, i.e. from the outside towards `aCenter`. This is necessary because the sample images also contain edges within the ellipse the system is looking for.

The search direction was set to `TCVN_ED_LIGHT_TO_DARK`, since in this sample the objects are darker than the background.

The parameter `nMaxThickness`, which defines the number of pixels with which `fMinStrength` must be reached, was set to "`7`".

Variables

```
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);
```

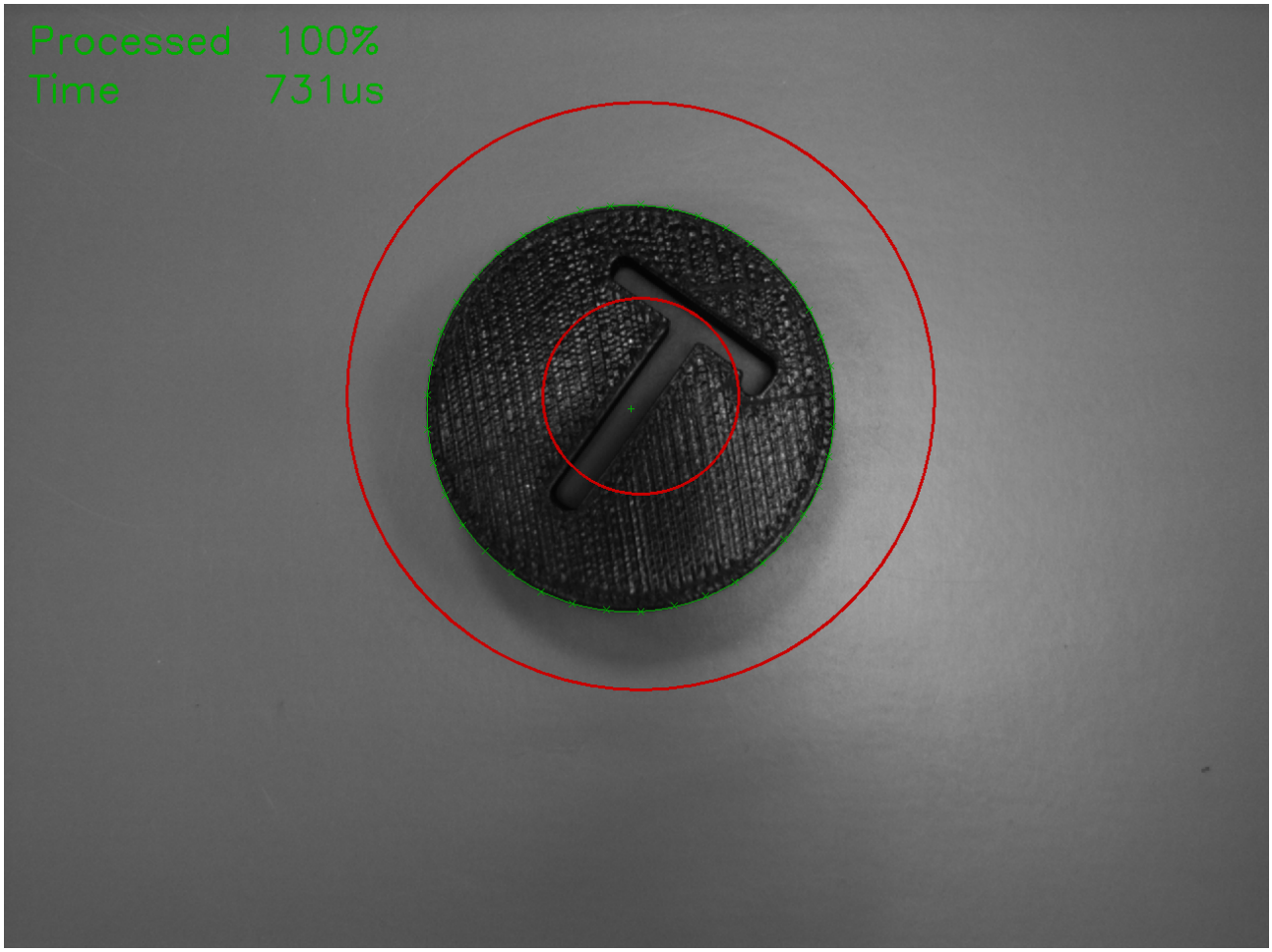
```
hr := F_VN_DrawPoints(ipContourPoints, ipImageRes, TCVN_DS_X, aColorGreen, hr);
hr := F_VN_DrawPoint(REAL_TO_UDINT(stEllipse.aCenter[0]), REAL_TO_UDINT(stEllipse.aCenter[1]),
ipImageRes, TCVN_DS_PLUS, aColorGreen, hr);
hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 1, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

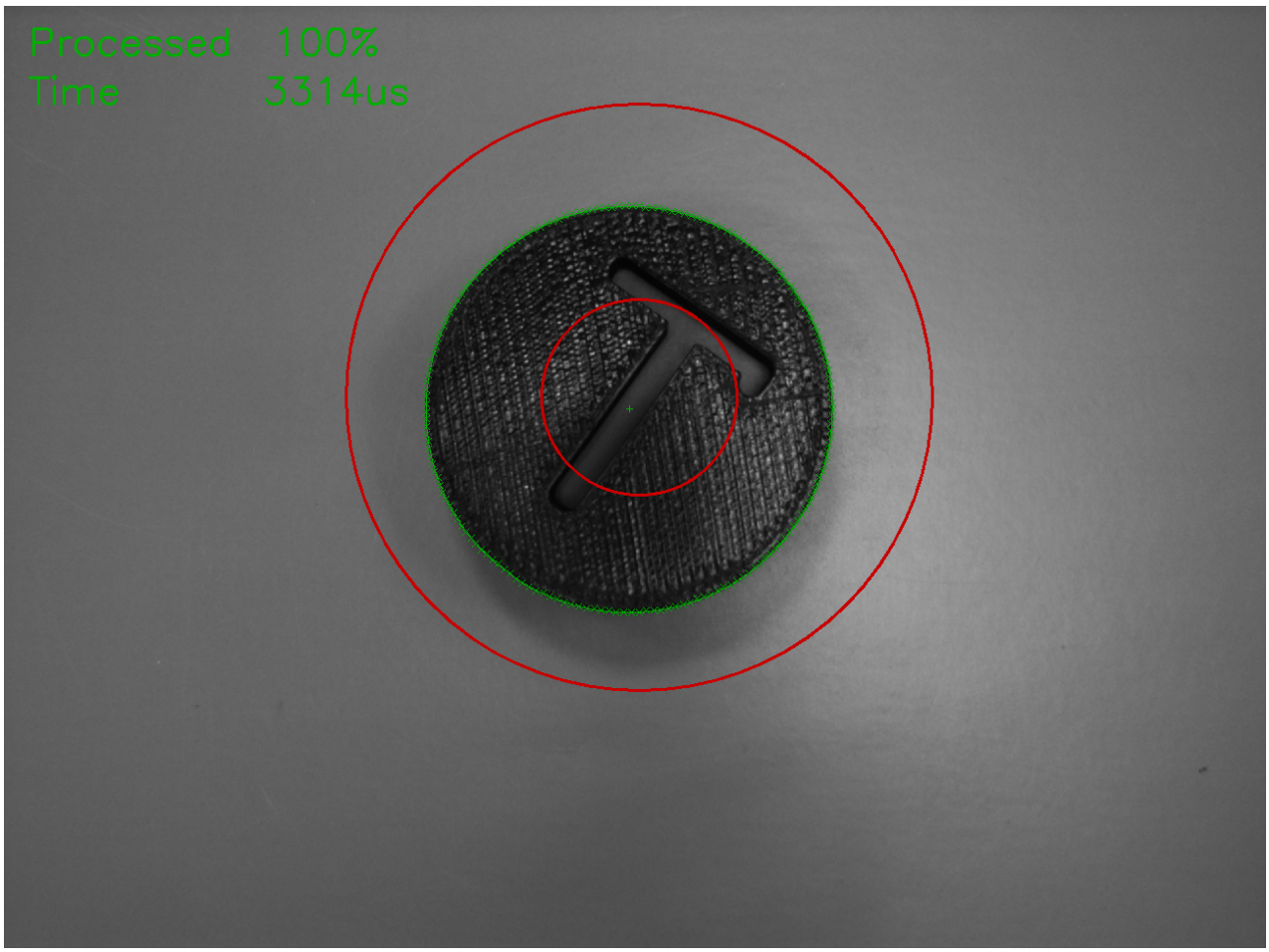
Results

The function returns the `TcVnRotatedRectangle` structure `stEllipse`, which defines the ellipse that was found by its center `aCenter`, its height and width `stSize.fHeight` / `stSize.fWidth` and its angle `fAngle`.
Optionally, one additional edge point per search line `ipContourPoints` can be returned from which the ellipse was approximated.

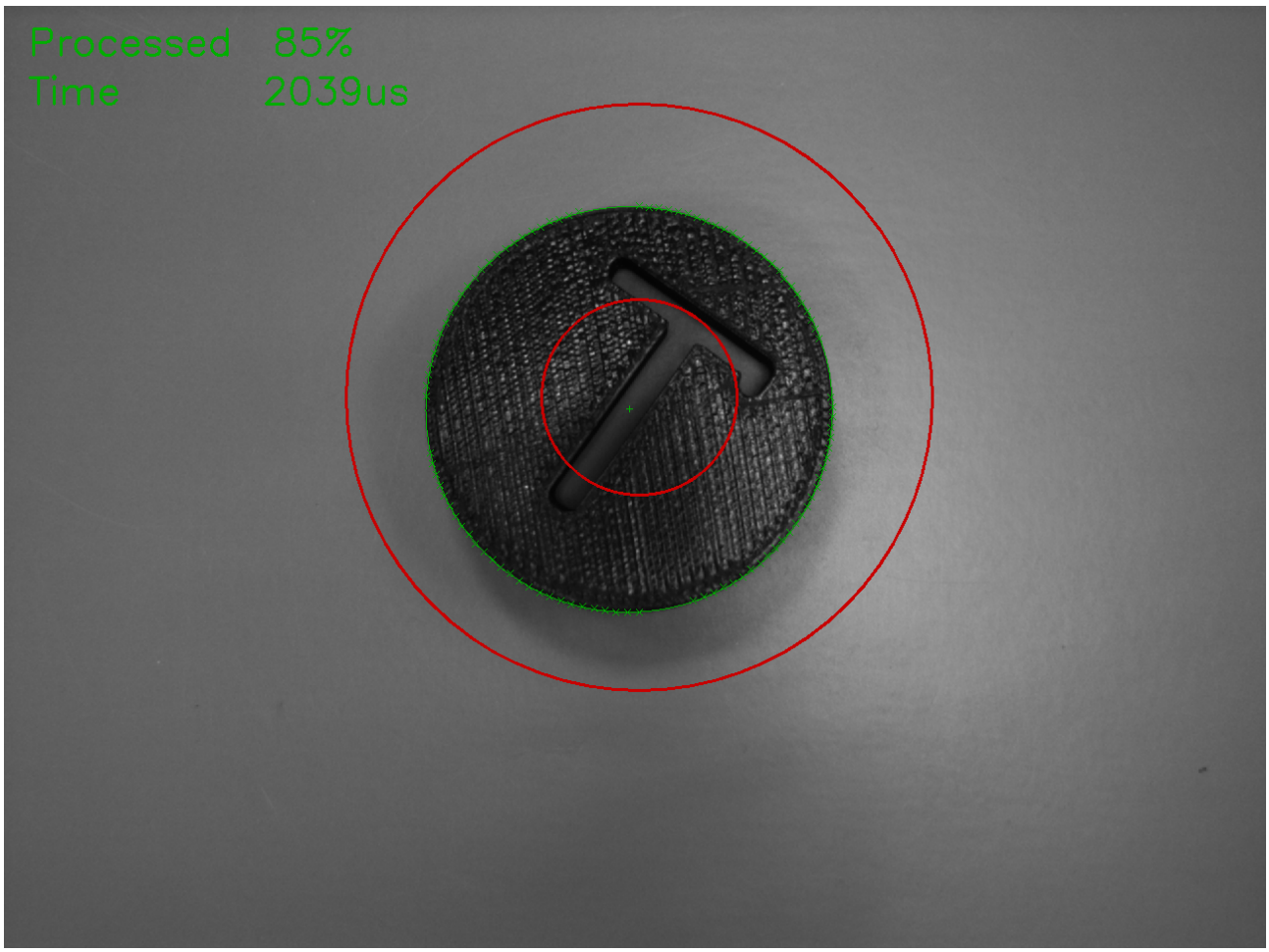
For visualization, the search area is first drawn into the image in the form of two red circles around the specified center point. The ellipse that was found is drawn as a green line and the corresponding center as a green „+“. The individual measuring points from `ipContourPoints` are drawn as green x. In addition, the processing percentage and the required computing time are shown at the top left. For the parameters used in this sample, the result for the input image *LocateEllipse_T.bmp* looks like this:



If the number of search lines is increased from 40 to 180, the required computing time increases linearly as expected:

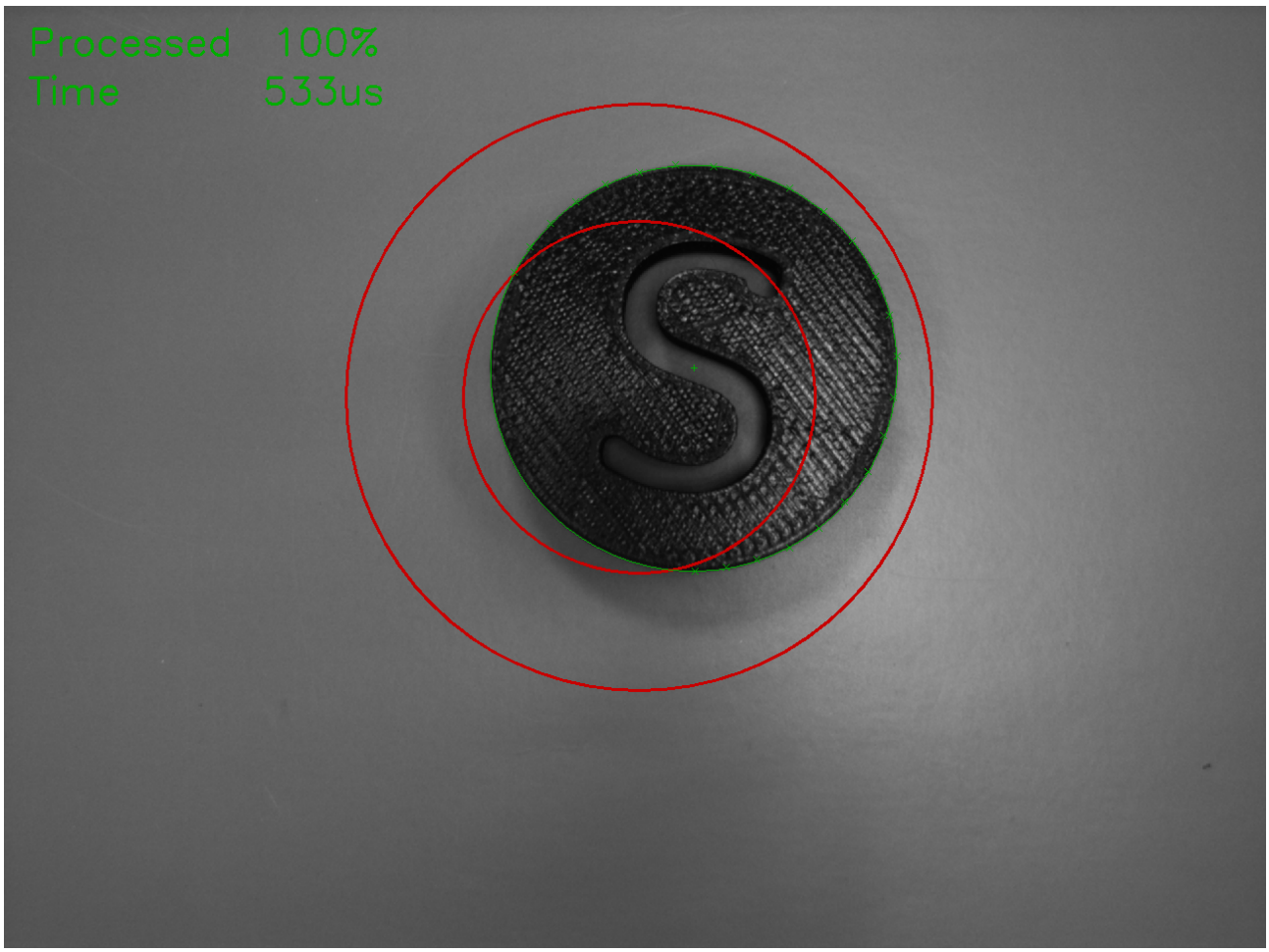


In practice the computing time varies somewhat, depending on the input image. In many cases it is therefore useful to limit the computing time by means of a watchdog. If, for example, the number of search lines is set to 120 and the watchdog timeout `tStop` to 2000 (depending on the CPU used), the function is terminated at around 85% of the processing time, and the partial results available at this point are returned:



Since the algorithm can only be interrupted and the partial results returned at specific points, the function does not abort exactly after 2000 µs but takes a little longer, which must be taken into account when selecting the abort time. The maximum additional time required generally depends on the algorithm and the parameterization.
Although the results of some search lines are missing due to the termination (four relatively large gaps between the green x), the ellipse could still be correctly approximated, i.e. the partial results can be used as if the function had been processed 100%.

This is also possible to a limited extent if the position of an object is partly outside the defined search range, e.g. because a previous positioning step was incorrect. If, for example, `fMinRadius` is set to 180, part of the outer contour of some sample images lies outside the specified range. Nevertheless, enough search points are available to approximate an ellipse. In practice, however, once should not rely on this and should set the search area large enough:



Similar samples

- Locate Circular Arc [▶ 2813]
- Locate Edge [▶ 2817]
- Measure Angle Between Edges [▶ 2827]

8.1.9.4 Measure Angle Between Edges

In this sample,

- the function F_VN_MeasureAngleBetweenEdgesExp [▶ 1515] is used to measure the angle between two edges of an object
- and the execution time is monitored with a watchdog and limited if necessary.

Explanation

After defining the search windows, the search direction was set to `TCVN_ED_LIGHT_TO_DARK`, because in this sample the search is performed from the inside to the outside and the objects are lighter than the background.

The parameter `nMaxThickness`, which defines the number of pixels with which `fMinStrength` must be reached, was set to "`7`".

Variables

```
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

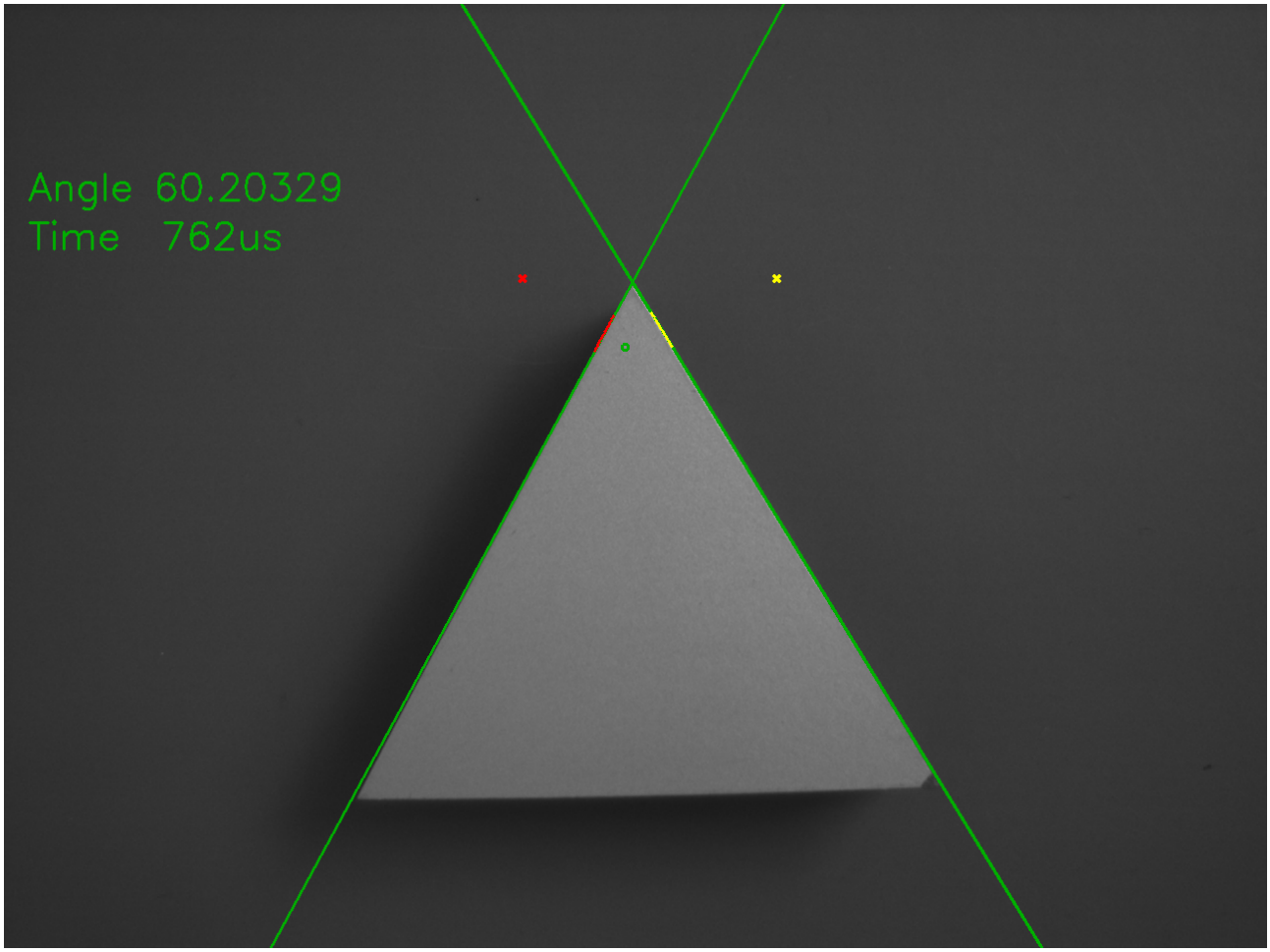
```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
   hrFunc := F_VN_MeasureAngleBetweenEdgesExp(
                ipSrcImage              :=   ipImageIn,
                fAngle                  :=   fAngle
                aInnerPoint             :=   aInnerPoint,
                aOuterPoint1            :=   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);

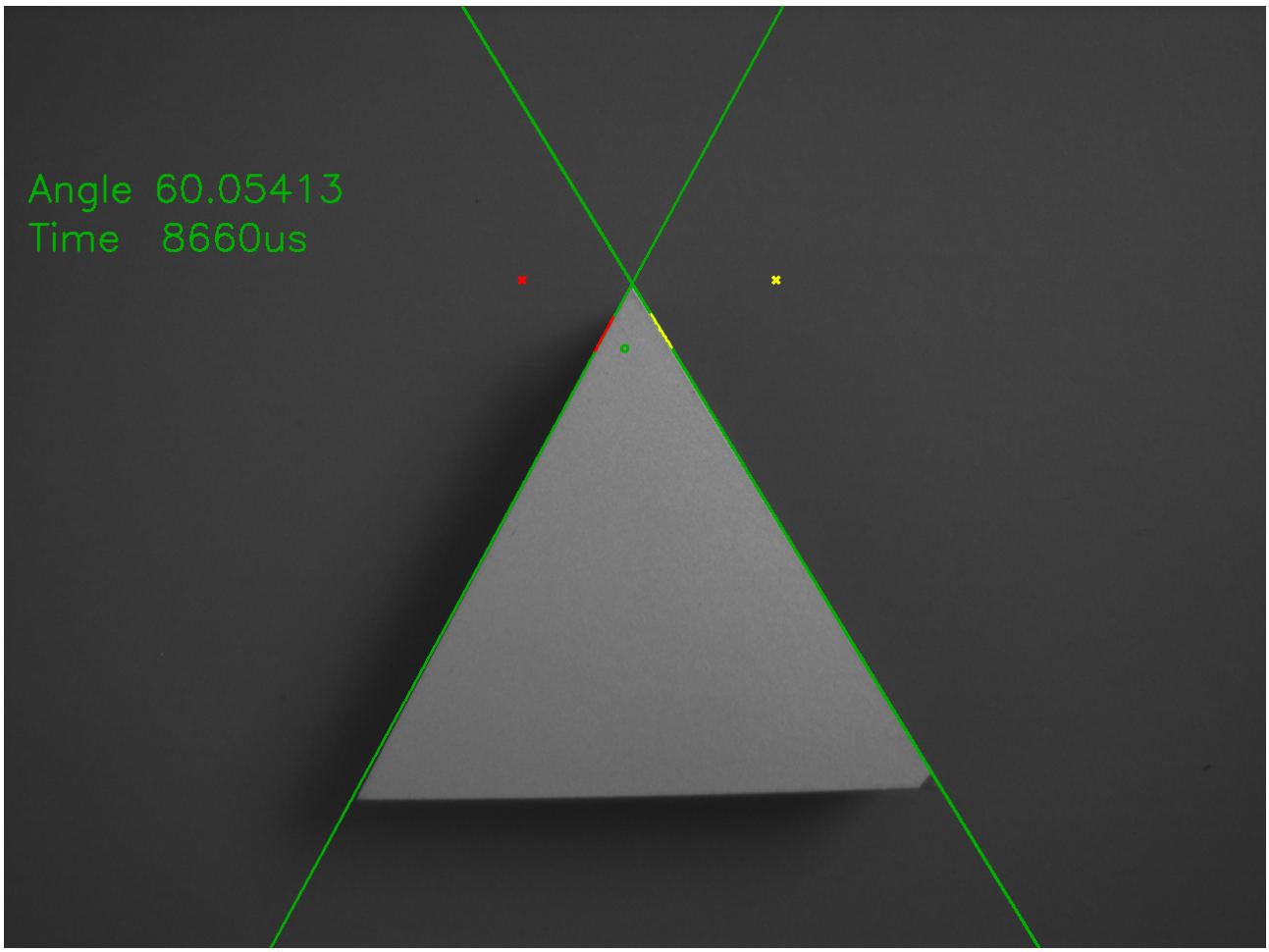
// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

Results

For visualization purposes, `aInnerPoint` is first displayed as a green circle, `aOuterPoint1` as a red x and `aOuterPoint2` as a yellow x. The corresponding edge points `ipEdgePoints1` and `ipEdgePoints2` that were found are also drawn in red or yellow. The lines approximated from the edge points are drawn in green. The calculated angle `fAngle` in degrees and the required computing time in µs are displayed on the left in the image. For the parameters used in this sample, the result looks like this:



To get a more accurate result, `eAlgorithm` can be changed to `TCVN_EDA_APPROX_ERF` and `nSubpixIter` to `50`. However, this also significantly increases the computing time required:



Similar samples

- Locate Circular Arc [▶ 2813]
- Locate Edge [▶ 2817]
- Locate Ellipse [▶ 2822]

8.1.10 Self-written functions

This sample explains what needs to be observed when writing your own functions for Vision applications. It essentially concerns:

- the handling of the HRESULT [▶ 141]
- the handling of interface pointers [▶ 137]

Other structures too

This function implementation is based on the structure of the TwinCAT Vision API functions. Other structures are also conceivable, but please pay attention to the points explained here.

Methods too

In this sample a function is described. However, it applies equally to methods.

For this purpose, by way of example, a self-written function for the counting and painting of objects is considered:

```
FUNCTION F_CountAndDrawObjects : HRESULT
```

Declaration part

The attributes of the function are declared as follows in the VAR_INPUT part. The original image is transferred directly as ITcVnImage. Since no variable values are stored in a function or method, nothing else needs to be observed in this case. On the other hand, the result image is passed as REFERENCE TO ITcVnDisplayableImage, since it is created in the function and is to be returned externally for further processing. In this case, no copy of the interface pointer is created due to the reference.

```
VAR_INPUT
    ipSrcImage          :   ITcVnImage;
    ipDestImage         :   REFERENCE TO ITcVnImage;
    aColor              :   TcVnVector4_LREAL;
    nNumberOfObjects    :   REFERENCE TO ULINT;
    hrPrev              :   HRESULT;
END_VAR
```

In addition, some auxiliary variables required internally by the function are declared in the VAR.

```
VAR
    hr                  :   HRESULT;
    ipContours          :   ITcVnContainer;
    stParams            :   TcVnParamsBlobDetection;
END_VAR
```

Input checks

At the start of the function, some input checks are performed in order to ensure that the function can be executed properly.

Initially, FAILED(hrPrev) is used to check whether an error has occurred in the previous processing chain. If so, the same error is displayed and the function is ended directly.

```
IF FAILED(hrPrev) THEN
    F_CountAndDrawObjects := hrPrev;
    RETURN;
END_IF
```

A check is also performed to ascertain whether all interface pointers that are transferred to the function as VAR_INPUT are valid. Otherwise the function cannot do anything with them and returns INVALIDPARM as the return code.

```
IF ipSrcImage = 0 THEN
    F_CountAndDrawObjects := Tc2_System.E_HRESULTAdsErr.INVALIDPARM;
    RETURN;
END_IF
```

If all input checks were successful, the reference counter of all interface pointers transferred as VAR_INPUT must be incremented, because due to the transfer of the interface pointer a copy is created that is retained after the function block is executed and has to be mapped in the reference counter. The method TcAddRef can be executed here directly without a prior <> 0 check, as this has already taken place during the input checks. As already described in the declaration part, incrementing the reference counter is not necessary for functions and methods, because the variable values, like in this case the pointer address, are not preserved after execution.

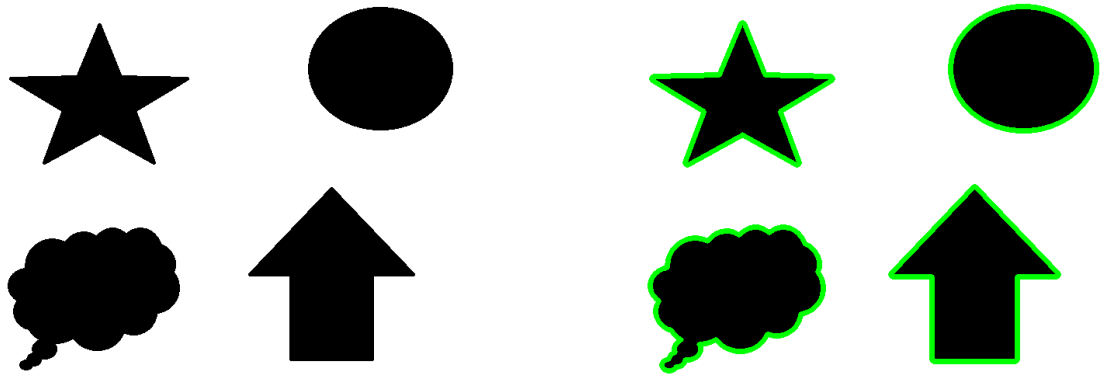
Main processing

The actual image processing sequence is located in the main part of the function. This is only an example here and will not be considered any further. The call of F_VN_ConvertColorSpace illustrates that interface pointers, which are transferred as REFERENCE TO for the purpose of the result return, can be written directly by TwinCAT Vision API functions.

```
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);
```

By means of this image processing sequence, objects are painted in the result image and their quantity returned as ULINT:

Original image **Result image**



Number of objects found: 4

Tidy up and return

Following the actual image processing sequence, the interface pointers declared within the function are released again. This is the case because all variables declared in the function (i.e. the interface pointers too) are deleted after ending the function. If there are still data in the background that have not yet been released, this will lead to memory leaks.

```
FW_SafeRelease(ADR(ipContours));
```

Finally, the `HRESULT` is assigned to the function as the return value in order to report any errors to the outside. So that it is not falsified, the return values of the `FW_SafeRelease` functions should not be assigned to the `HRESULT`.

```
F_CountAndDrawObjects := hr;
```

8.2 Function block samples

This section contains samples for the function blocks [▶ 1570].

8.2.1 Camera Register Access

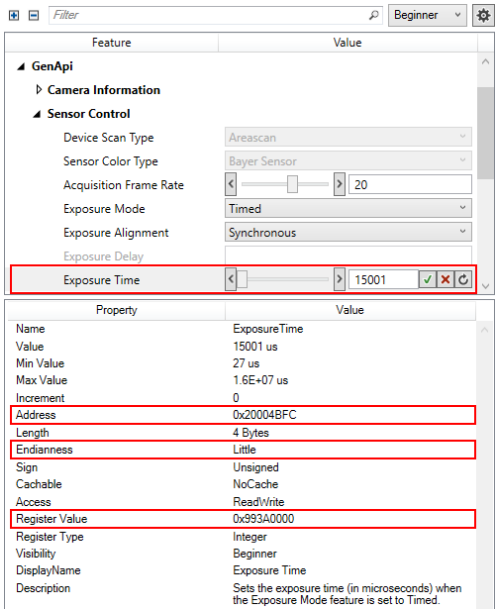
8.2.1.1 Reading/writing a register

All camera parameters that can be read or written via the Configuration Assistant in Config mode can also be read from the PLC via the function blocks FB_VN_ReadRegister_UDINT [▶ 1576] and FB_VN_ReadRegister_REAL [▶ 1574] or written via FB_VN_WriteRegister_UDINT [▶ 1584] and FB_VN_WriteRegister_REAL [▶ 1582]. The function block required depends on the data type of the parameter and can be read in the Property list on the Register Type. In this sample, the exposure time is written and read as an integer.

```
fbReadValue    :    FB_VN_ReadRegister_UDINT;
fbWriteValue   :    FB_VN_WriteRegister_UDINT;
```

Both function blocks require the address of the parameter and an indication of the byte sequence (endianness). The Write function block additionally requires the parameter value that is to be written to the camera. As the value displayed or entered under Value need not directly conform to how the camera processes it internally, e.g. in the case of enums or other byte sequences, this can be read under the **Value** register.

This information can be read from the Configuration Assistant:



NOTICE

Camera-specific parameters

The parameter names, properties and values shown here are vendor-specific or camera-specific and firmware-specific and may therefore differ.

Variables

```
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);
```

Conditions for the reading and writing of parameters

Parameters can generally only be read or written with an opened camera connection (`ETcVnCameraState >= TCVN_CS_OPENED`). Please refer to the respective vendor documentation for the camera for the exact conditions.

Both function blocks must be initialized with the **Image Provider** of the camera:

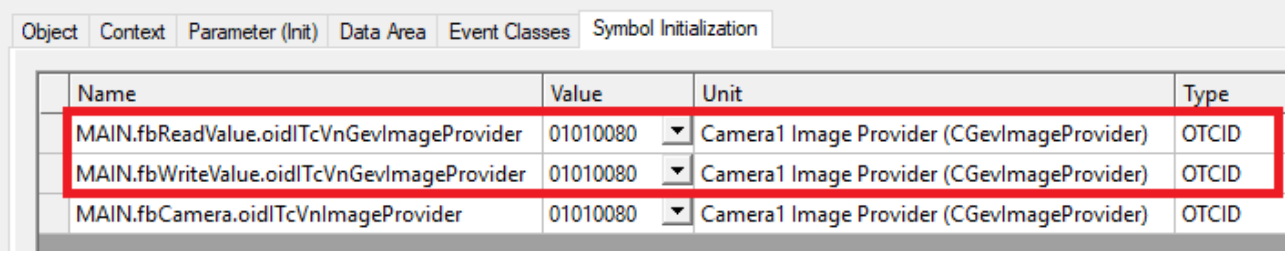


Fig. 31: Linking function blocks with a camera instance

8.2.2 File Access

8.2.2.1 Saving images from the PLC

In this sample, images from the PLC are saved to a local drive. The function block FB_VN_WriteImage [▶ 1605] is used for this purpose.

The save process takes place asynchronously over several cycles.

Duration of the save process

The duration of the save process depends on various factors and can therefore not be stated in general terms.

Variables

```
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 this sample an image can be saved by setting the trigger variable `bWriteImageTrigger` to `TRUE`.

Code

The `ipImage` transferred is already completely accepted on the first call of `fbWriteImage` with a rising edge at `bWrite` and can therefore be released immediately afterwards, e.g. with `F_VN_TransformIntoDisplayableImage`. Thereafter, the `fbWriteImage` must continue to be cyclically called until the `bBusy` output is `FALSE` again. This can be the case with an error or even with the first call, e.g. if no file extension or an incorrect one is specified. An evaluation of whether writing was successful or whether an error occurred takes place in the IF block below. If successful, `bWriteImageDone` is set and remains unchanged until the next time `bWriteImageTrigger` is set.

```
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 Serial loading of images into the PLC

In this sample you load images serially into the PLC. For this you use:

- FB_VN_ReadImage [▶ 1598]

This sample applies analogously to the loading of containers with FB_VN_ReadContainer [▶ 1596].

Explanation

A frequent use case is where reference images are saved for comparison with the live image in the controller. These reference images should be replaced during operation, e.g. when a new reference is introduced. Images may be reloaded any number of times and at any interval.

So as not to have to inform the PLC of the file name for every new reference image, define a naming convention. This could be, for example, `Image<Index>.png`, where `<Index>` is to be replaced by a sequential integer. Each new reference image is then given a file name with an index incremented by 1. The PLC thus knows what the name of the next reference image must be and can purposefully attempt to load it.

NOTICE

Risk of file corruption

Never replace the content of an image file; instead, only create new images in the file system! Firstly, replaced images are not registered due to internal caching mechanisms of the Vision service. Secondly, simultaneous access to an image by several processes can lead to a file corruption. The cache settings can be adjusted alternatively.

Application

6 color images with the name `image<Index>.png` and the empty folder `load` are attached to this sample. Define the absolute path of the `load` folder as `sBasePath`. After starting the sample PLC, move the images in order (starting with `Index=0`) to the `load` folder and observe in the ADS Image Watch that the image `ipImageRefDisp` changes accordingly.

Program

In the cyclic program you initially calculate the path of the file to be loaded, based on the index `nIndex`. Also, adjust the base path `sBasePath` in the variable declaration so that it matches the memory location of your images.

```
sPath := CONCAT(CONCAT(sBasePath, TO_STRING(nIndex)), '.png');
```

Using the calculated path, cyclically call the function block fbReadImage of the type FB_VN_ReadImage [▶ 1598].

```
fbReadImage(
    sFilePath   :=  sPath,
    ipDestImage :=  ipImageIn,
    bRead       :=  TRUE,
    nTimeout    :=  T#1S
);
```

In addition to calling the function block, you must react when it is ready. On the one hand, you reset the function block by means of a call with `bRead:=FALSE`; on the other, you increment the index and forward the image to the desired interface pointer, provided the execution is successful. Querying whether an error has occurred is very important, as you may only increment the index if the current image has already been loaded successfully.

```
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
```

In order to save memory and computing time, the forwarding of the image takes place here via an orderly transfer of the interface and not via a copy.

8.2.3 Image acquisition

8.2.3.1 Triggering an image by file name

In this sample an image is purposefully triggered on the basis of the file name in the function block FB_VN_FileSourceControl [▶ 1613].

Application

The sample displays alternately two images with the file names *Image1.bmp* and *Image2.bmp* from the file source in ADS Image Watch [▶ 131]. The sample includes the images with corresponding file names. You can replace the sample images with your own images or add them additionally. To do this, change or add to the file names in `sFileName` and adjust the number of elements if necessary. Note that all images to be displayed must be added in the File Source Control [▶ 126]. By setting `bLoop` to `false` you stop the automatic switching of images and then you can select the image to be triggered by setting `nImageIndex`.

Program

```
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	Function block for controlling the File Source [▶ 125] of the type FB_VN_FileSourceControl [▶ 1613].
eState	State of the File Source
ipImageIn	Input image
ipImageInDisp	Displayable input image
bTrigger	Trigger flag
sFileName	File name – change this variable in order to load a different image. However, this image must be added in the File Source Control [▶ 126].
hr	Status variable of the type HRESULT [▶ 141].

In general, the usual state machine is used to operate the file source:

```
eState := fbFileSource.GetState();
CASE eState OF

<...>

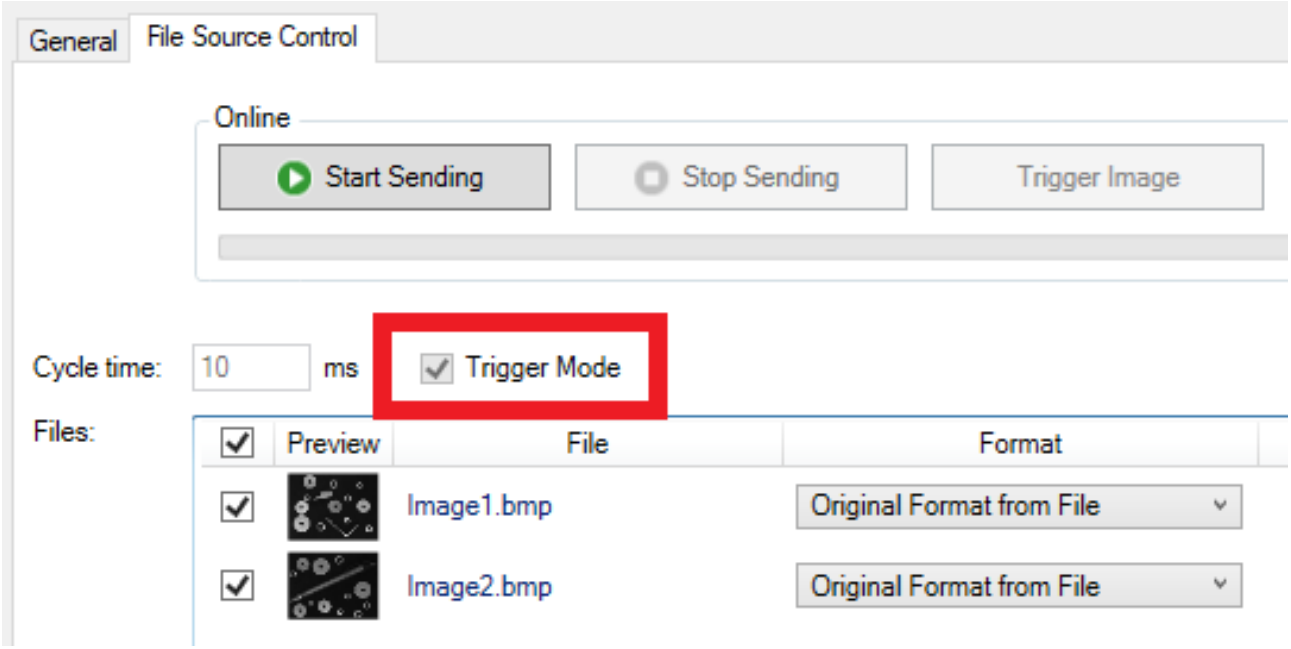
END_CASE
```

Resetting in case of errors and the starting of the image acquisition take place just as normal in the corresponding states:

```
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();
```

However, so that the file source doesn't load images immediately when the image acquisition is started, the **Trigger Mode** must be activated in the File Source Control:



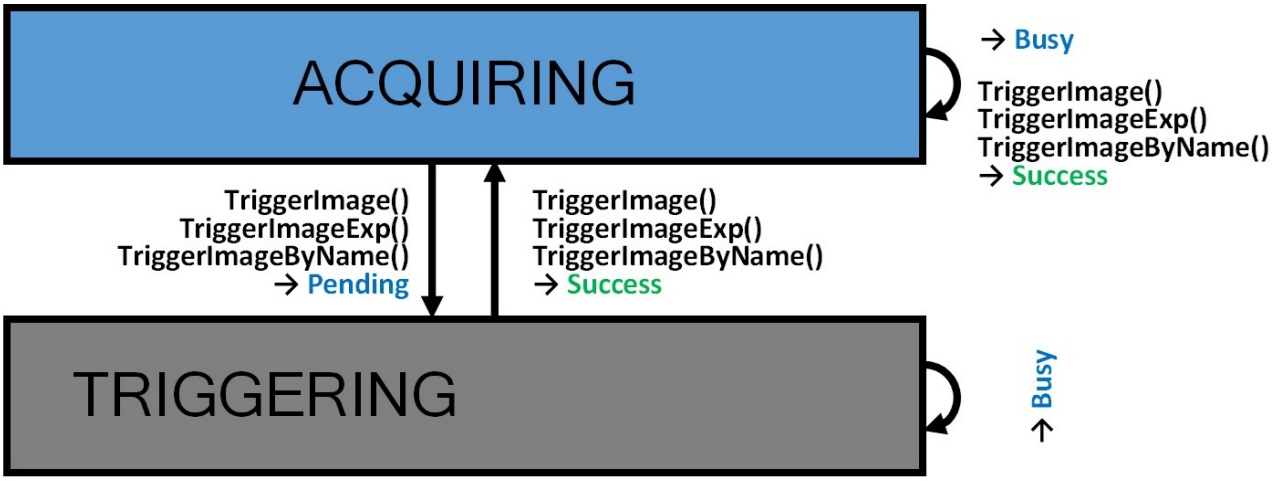
In the ACQUIRING state, the image with the corresponding file name will now be alternately triggered and then received. The method TriggerImageByName [▶ 1621] is explicitly used here to trigger the image. This method exists only in the function block FB_VN_FileSourceControl [▶ 1613] and not in FB_VN_SimpleCameraControl [▶ 1646].

```
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
```

As the File Source temporarily enters the `TRIGGERING` state when triggering the image, triggering must continue in this state. This continues until the File Source has fully loaded the image and thus changed its state back to `ACQUIRING`. The file name of the image does not need to be transferred again for further triggering; it is sufficient to call the method TriggerImage [▶ 1620].

```
TCVN_CS_TRIGGERING:
    fbFileSource.TriggerImage();
```



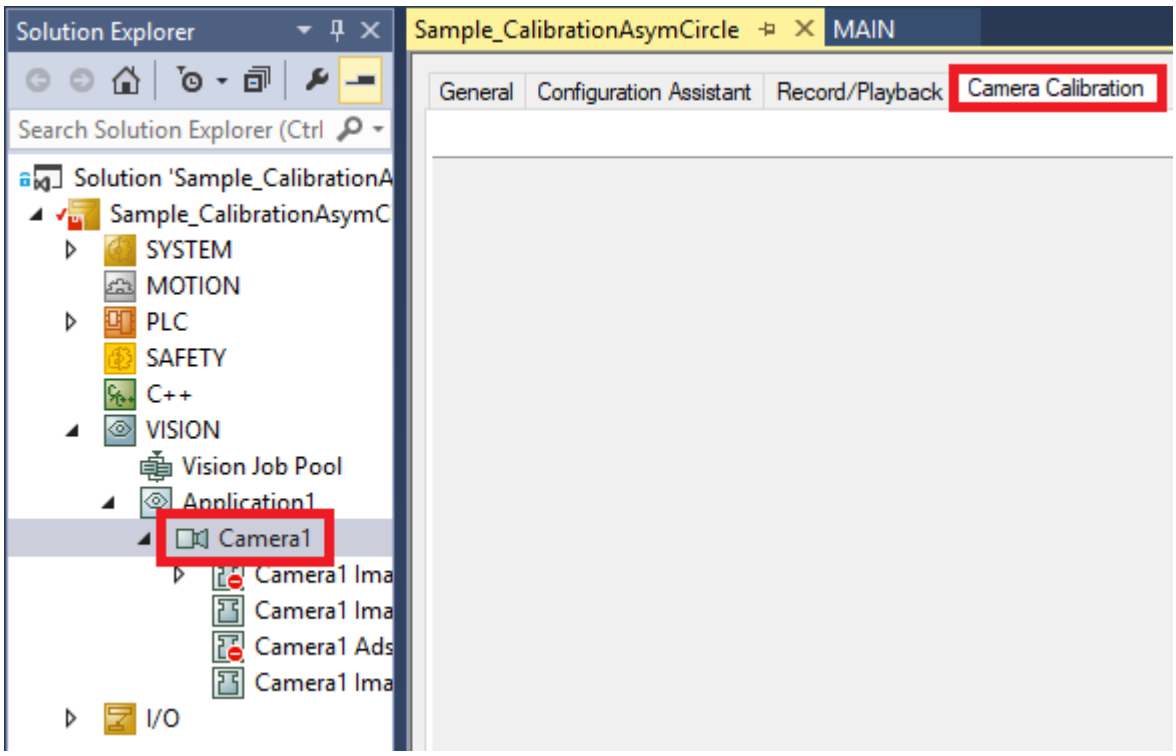
In this sample, the variable `bTrigger` is automatically set to `TRUE` for demonstration purposes when an image has been successfully loaded. In a real application the trigger would be set at a point that makes sense in terms of the sequence.

8.3 Samples of assistants

8.3.1 Calibration Assistant

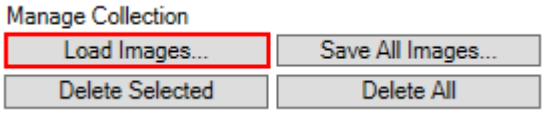
A geometric camera calibration is required to convert image points to world points, during which the required parameters are determined. This sample shows the geometric calibration of a camera using an asymmetric circle pattern and the Calibration Assistant. The steps can also be performed analogously for the other calibration patterns (see Camera Calibration [▶ 107]). Only step 2 has to be adapted accordingly.

To run this sample, first open the project with the camera created and then open the calibration assistant for the camera instance.



Step 1 - Image acquisition

First, images of the calibration pattern taken by the camera to be calibrated must be loaded into the Assistant. This can be done either by direct image acquisition with the camera or by loading existing images. For this sample, you will find four suitable images *ImgCalibAsymCirclesx.bmp* of an asymmetrical circle pattern in the corresponding image folder. Load these images into the Calibration Assistant via **Load Images....**

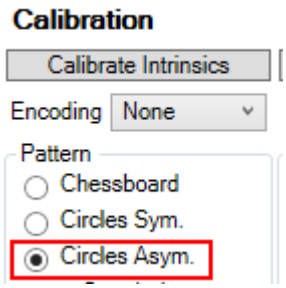


Step 2 - Define asymmetric circle pattern

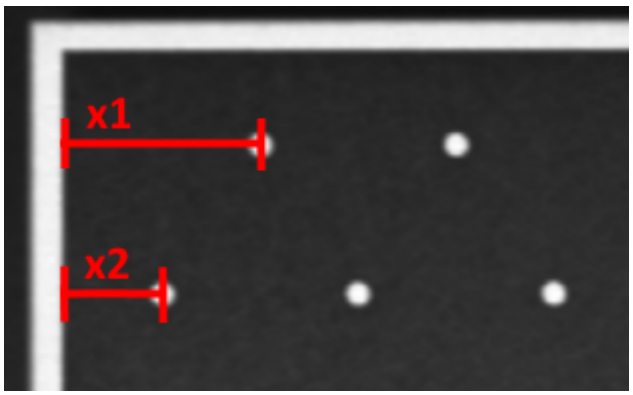
In this sample an asymmetric circle pattern is used as calibration pattern. In order to use one of the other pattern types, observe the respective definition:

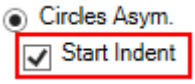
- Chessboard pattern [▶ 116]
- Symmetrical circles pattern [▶ 117]
- Asymmetrical circles pattern [▶ 118]
- Individual circles pattern [▶ 119]

The asymmetric circle pattern must be selected in the Calibration Assistant for the loaded sample images:

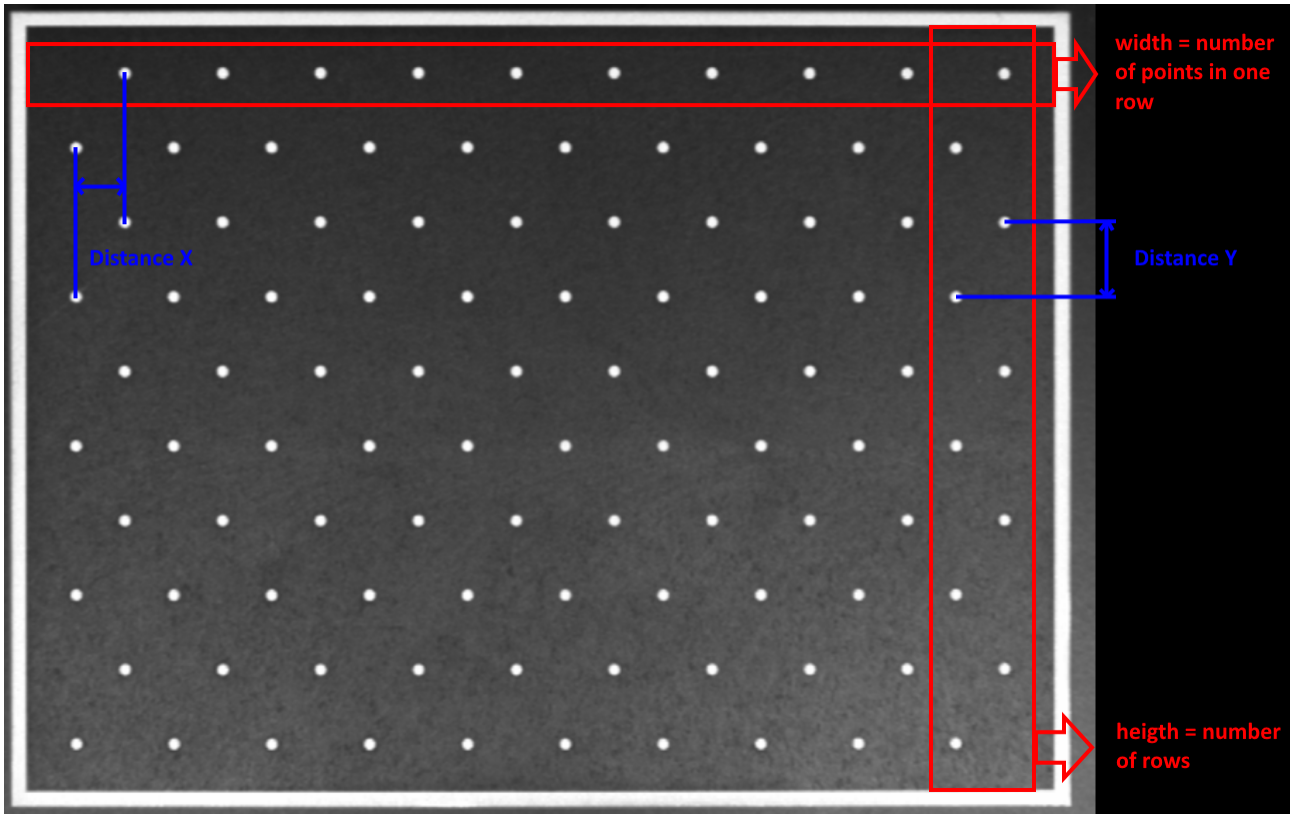


Due to the asymmetry, either the first (x1) or the second (x2) row of the circle pattern is indented. In this case the first (x1) row is indented, so **Start Indent** should be selected.

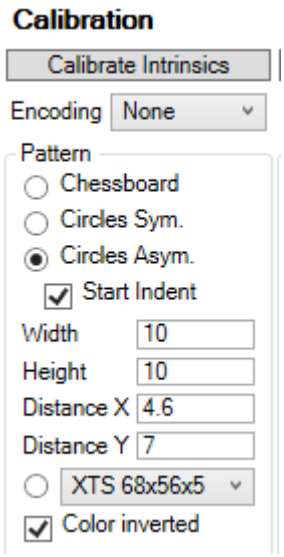




The width of the **Width** pattern corresponds to the number of points/circles in a row, while the **Height** corresponds to the number of rows. **Distance X** is used to specify the shortest distance between two circles in X direction. In the asymmetrical pattern, this is the distance to the next circle in the row below / above. Note the marking in the image. The distance between two rows is specified in **Distance Y**.

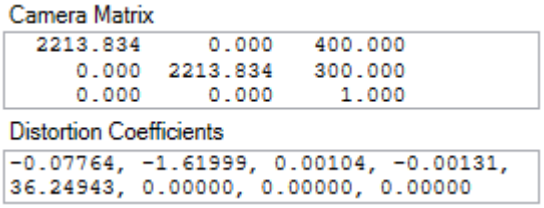


The Assistant expects black circles on white background. Since this example uses a pattern of white circles on a black background, the color must be inverted by selecting **Color inverted**.



Step 3 - Calculate intrinsic parameters

Once the calibration pattern has been fully defined, the intrinsic parameters can be calculated using the **Calibrate Intrinsics** button. The **Reproj. Error** should then be between 0 and 1, and the **camera matrix** and the **distortion coefficients** should be determined.

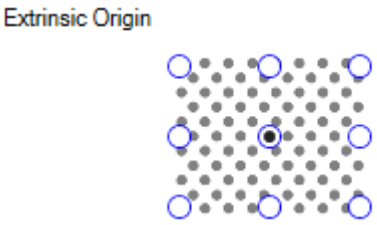


If this is not the case, the tooltip for the **Reproj. Error** provides information about the possible error. In addition, an image series is created under *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput* that provides further information on the recognition of the pattern. Write access for the directory is required.

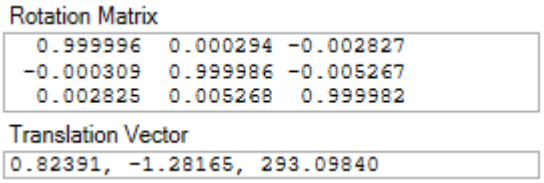
Step 4 - Calculate extrinsic parameters

After the intrinsic parameters the extrinsic parameters can be calculated. Only one image is used for this purpose, namely the image where the pattern is located in the plane on which the components are later measured, for example. This is the first of the sample images. Therefore, click on the first loaded image so that it appears in the image preview.

The zero point of the **Extrinsic Orgin** pattern can also be specified. By default, this is in the center of the pattern.

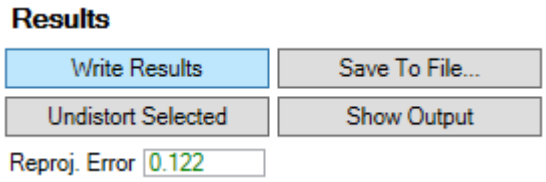


The **Calibrate Extrinsics** button then calculates the **rotation matrix** and the **translation vector**. The results can be found under **Results**.

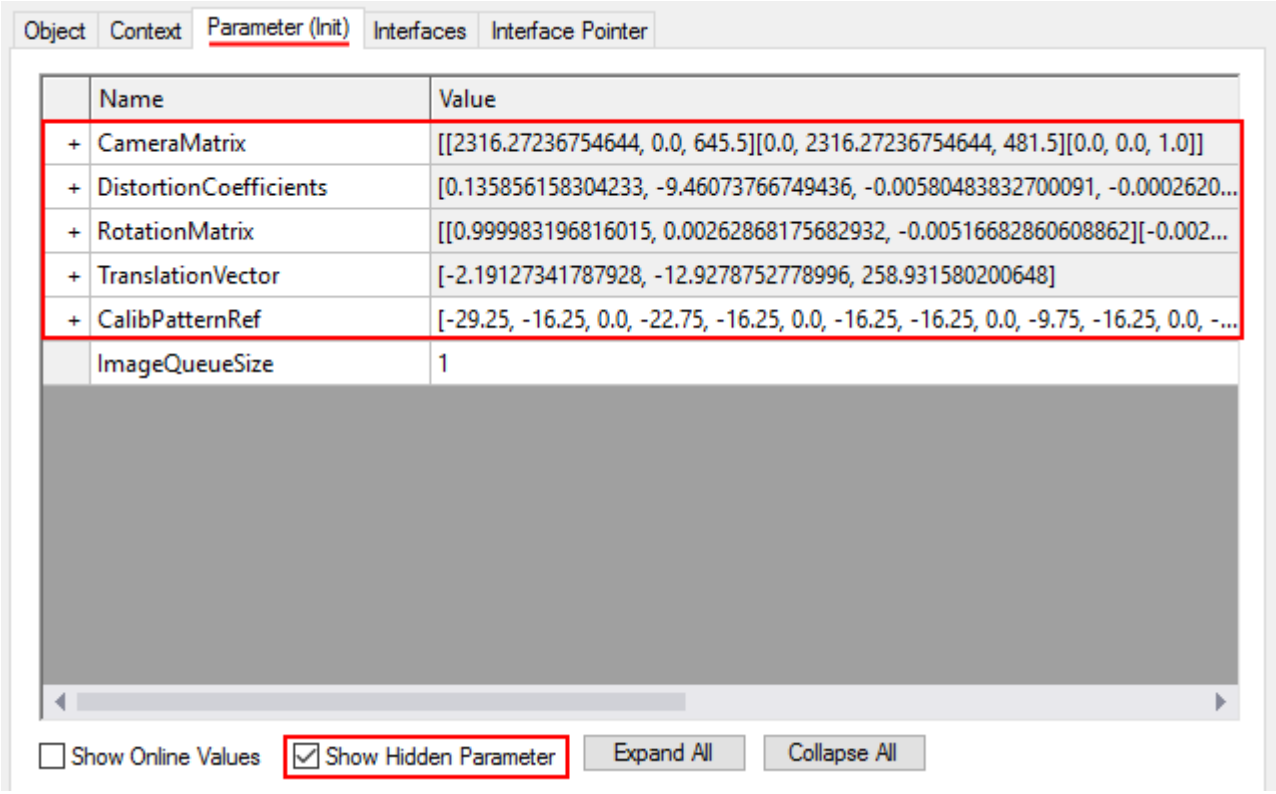


Step 5 - Write results to the image provider of the camera

If the calibration was successful, the results can be written to the Image Provider of the camera via **Write Results**. This step is necessary so that the camera function block FB_VN_GevCameraControl [▶ 1621] in the PLC can access the calibration results.



In the Image Provider, the calibration results can be found under **Parameters (Init)** if **Show Hidden Parameters** is selected.



After this step, make sure to activate the configuration by clicking on  . Otherwise the calibration results are not available in the PLC.

Step 6 - Load calibration results into the PLC

The function block FB_VN_GevCameraControl [▶ 1621] can be used to load the calibration parameters into the PLC:

Variables

```
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);
```

Step 7 - Transform pixels into world coordinates

Then use the function F_VN_TransformCoordinatesImageToWorld_Points [▶ 1108] to determine the coordinates of a point in the image in the real world in relation to the zero point of the calibration pattern, for example:

Variables

```
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);
```

Step 8 - Transform world points into image coordinates.

This transformation can be reversed with the function F_VN_TransformCoordinatesWorldToImage_Points [▶ 1112], so that points from the world coordinate system are transformed in such a way that they can be displayed at the correct position in the image. This is useful for displaying auxiliary lines for better orientation in the image. The following code shows the world coordinate system in the image.

Variables

```
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

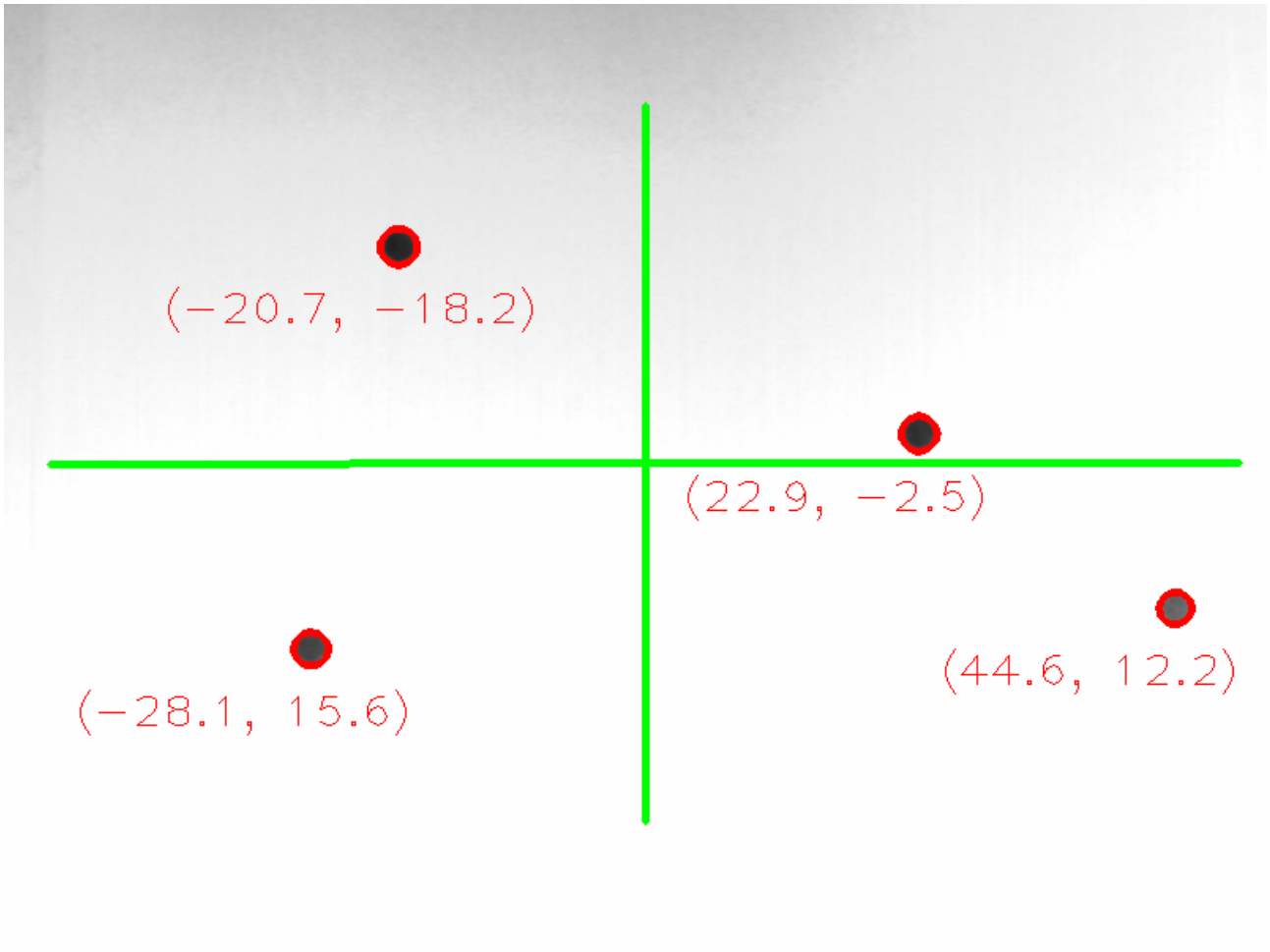
```
FOR i:= 0 TO 4 DO
    hr := F_VN_TransformCoordinatesWorldToImage_Point(
            aSrcPoint               :=  aCoordinatesWorld[i],
            aDestPoint              :=  aCoordinatesImage[i],
            aCameraMatrix           :=  aCameraMatrix,
            aDistortionCoefficients :=  aDistortionCoefficients,
            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
```

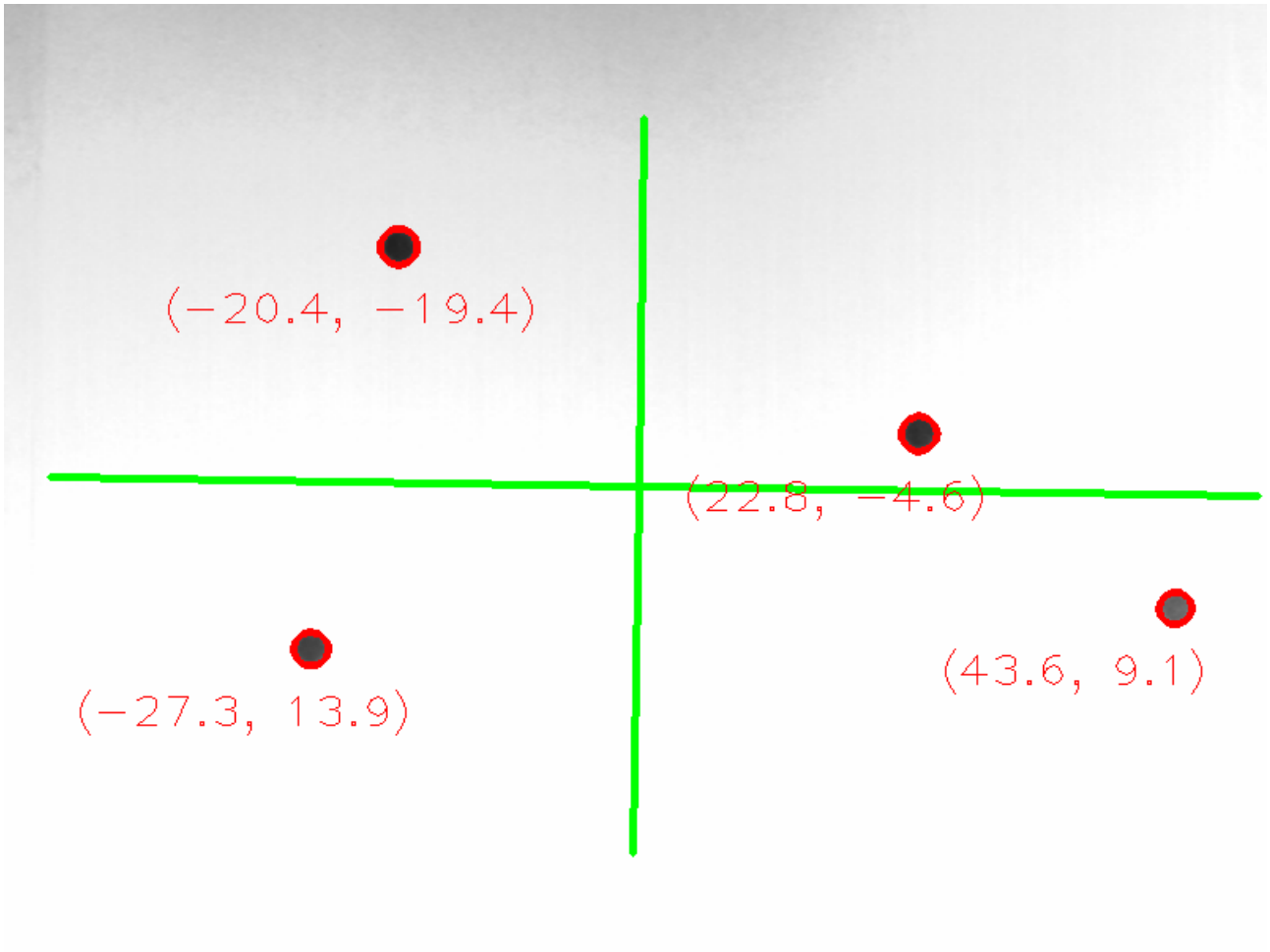
Step 9 – View the results

When the program is executed, the playback of the camera stream [▶ 104] must be started first. To do this, open the **Record/Playback** tab of the camera object, select the CameraSimulation.tcs file in the corresponding image folder and start the playback.

The resulting image can then be evaluated in the ADS Image Watch [▶ 131]. One can see the drawn coordinate axes and the determined world coordinates of the individual points on the image.



The result will be different if a different image or coordinate origin is used for the extrinsic calibration in step 4. The coordinate axes are shifted and the world coordinates of the pixels are also adjusted.



8.4 Camera configuration samples

Since the parameters of each GigE Vision camera may differ, it is not possible to provide a complete description. The following section provides an overview of frequently used camera parameters and how to configure them. Instructions for the configuration of GenAPI parameters can be found in the chapter on the Configuration [▶ 78].

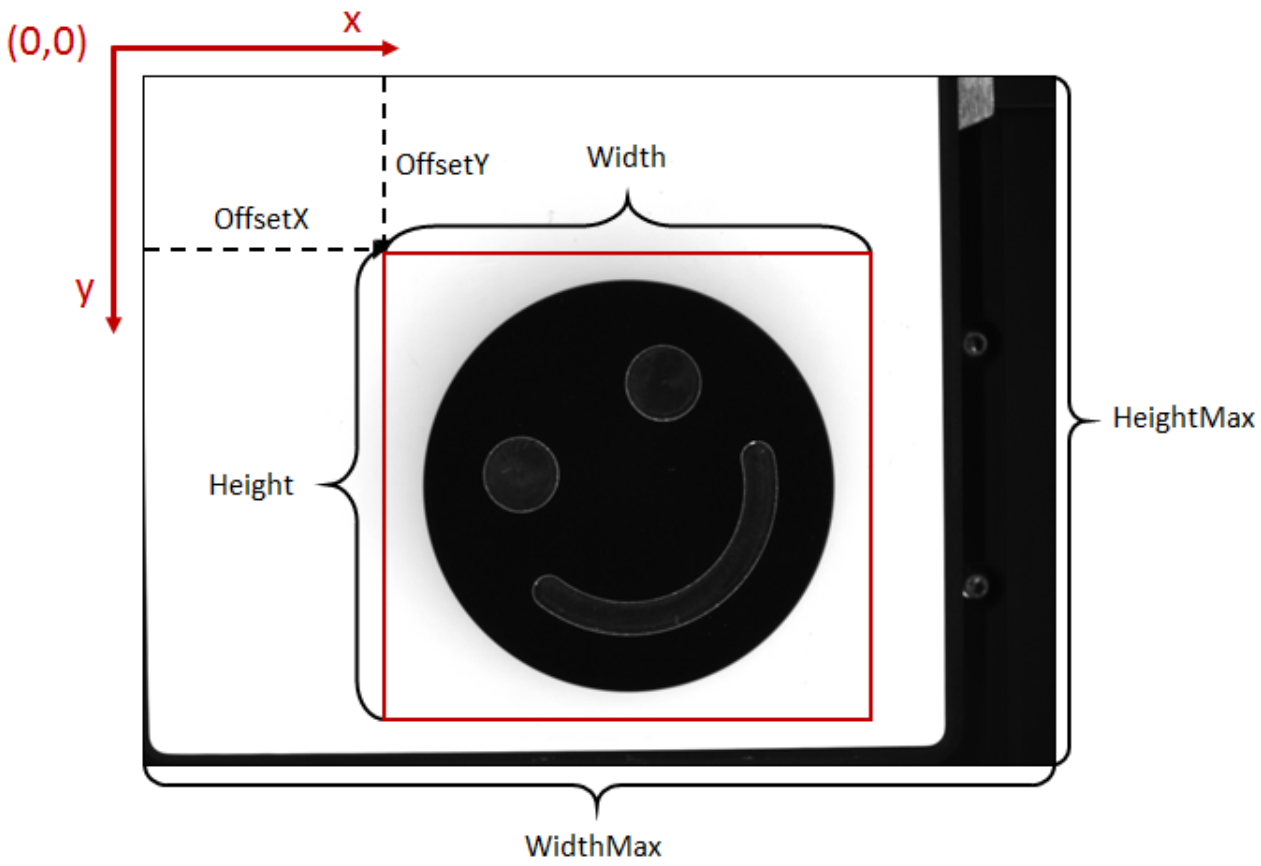
8.4.1 Region of Interest (ROI)

If the camera sensor records a larger field of view than required, it is advisable to place a Region of Interest (ROI) on the actually required field of view:

- The sensor can be read out faster, depending on the camera sensor and the selected range.
- The amount of data to be transferred is reduced, which also reduces the data transfer time.
- At the same time, the processing time of all image processing algorithms, which take place over the entire image, is also reduced.
- In addition, image noise can be reduced, making image processing more robust and easier.

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.



The features for setting a ROI can be found under `ImageFormatControl` or the corresponding category on your camera:

- `WidthMax` and `HeightMax` specify the maximum width and height of the image in pixels. These values may vary depending on other features (e.g. binning [▶ 2846]).
- `Width` and `Height` specify the width and height of the ROI, i.e. the field of view to be viewed, and must be set accordingly.
- `OffsetX` and `OffsetY` are used to position the ROI or the field of view. The starting point is the upper left corner of the image.

When setting the features, note that:

- `OffsetX` + `Width` <= `WidthMax`
- `OffsetY` + `Height` <= `HeightMax`

8.4.2 Binning

Binning is the summation or averaging of several pixel values into one.

It can be used to:

- increase the light sensitivity,
- improve the contrast,
- reduce the image resolution with the same field of view, enabling faster frame rates and processing times.

It should not be used if the required accuracies can no longer be achieved when the image resolution is reduced.

When summing up or averaging pixels, a distinction is made between horizontal and vertical binning.

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

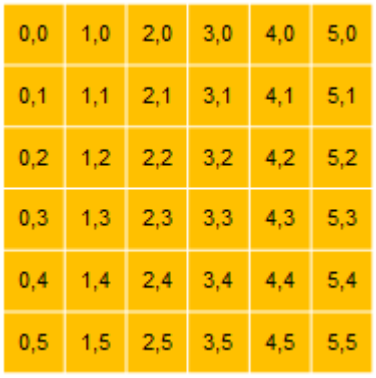
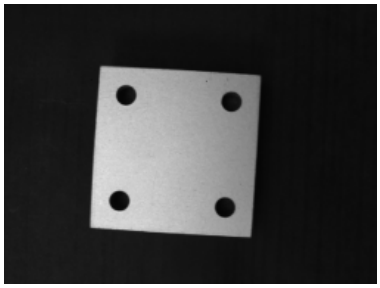
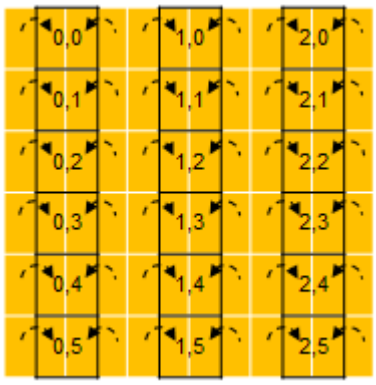
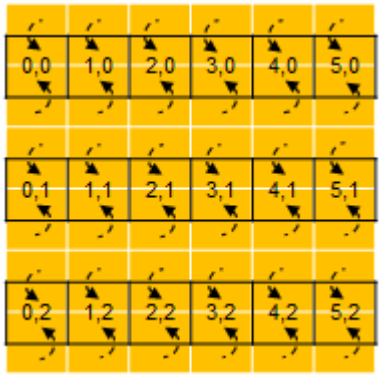
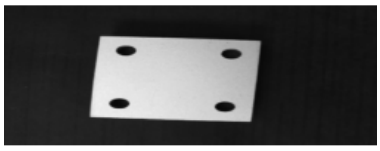
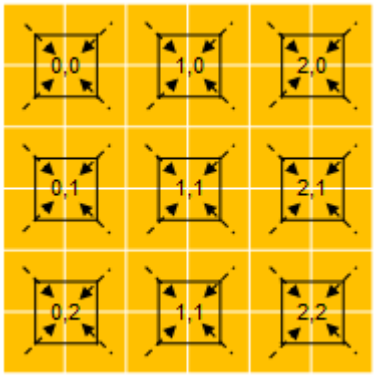
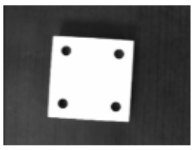
The features for setting up binning can be found under `ImageFormatControl` or the corresponding category on your camera:

- `BinningSelector` indicates whether the binning applies to the entire `sensor` or an ROI [▶ 2845].
- `BinningHorizontalMode` and `BinningVerticalMode` indicate whether the pixels should be added up or averaged.

`BinningHorizontal` and `BinningVertical` indicate how many pixels should be combined in horizontal and vertical direction. A value of 1 indicates that there is no binning in the corresponding direction.

Binning with a monochrome camera

Four possible configurations for a monochrome camera are shown as examples (the coordinates of the pixels are shown):

BinningHorizontal = 1 BinningVertical = 1 □ no merging of pixels		
BinningHorizontal = 2 BinningVertical = 1 BinningHorizontalMode = sum		
BinningHorizontal = 1 BinningVertical = 2 BinningVerticalMode = sum		
BinningHorizontal = 2 BinningVertical = 2 BinningHorizontalMode = sum BinningVerticalMode = sum		

Binning with a color camera

Binning support is rarer in color cameras than in monochrome cameras. It can be implemented in different ways. For details please refer to the camera user manual.

8.4.3 Image acquisition and trigger

The acquisition and trigger features are interrelated and are therefore be considered together below. The term `Acquisition` basically describes a state of cameras in which images can be taken, see ImageAcquisition [▶ 1611]. Whether and when images are actually taken and sent is determined by the configuration of the `Trigger` parameter.

The various configuration options and relationships between acquisition and trigger settings are described in more detail in the following chapters, which are arranged according to the `AcquisitionMode`.

By far the most commonly used image capture options are:

- Continuous Image Acquisition [▶ 2851] - the camera streams, it captures images at regular intervals without any other triggers.
- Single image trigger with continuous image acquisition [▶ 2851] - the camera takes exactly one image after a trigger signal (hardware or software). The camera remains in the state `ACQUIRING`, therefore it is not necessary to write the `AcquisitionStart` command repeatedly before each recording.

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Acquisition Mode

The `AcquisitionMode` feature mainly specifies how many images are taken during an acquisition and refers to the `TCVN_CS_ACQUIRING` state of the camera state machine. To end a recording, the state must be exited with a `AcquisitionStop` command before the next recording can begin again with a `AcquisitionStart` command. Therefore, the `Continuous` mode is the most frequently selected setting, since you can then remain permanently in the acquiring state.

- `Continuous`
  - Image acquisition begins with the `AcquisitionStart` command and is only terminated by the `AcquisitionStop` command. In between, images are continuously captured depending on the trigger settings.
  - Continuous image acquisition [▶ 2850]
- `MultiFrame`
  - Images are captured until the number of images specified in `AcquisitionFrameCount` is reached. A `AcquisitionStart` command is required for each image acquisition sequence.
  - Multi-image acquisition [▶ 2852]
- `SingleFrame`
  - A single image is captured. A `AcquisitionStart` command is required for each image acquisition.
  - Single image acquisition [▶ 2854]

Trigger Source

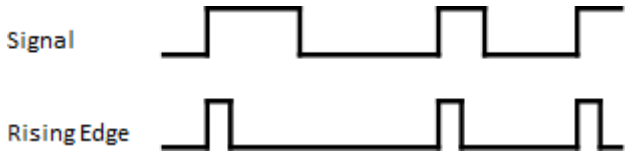
The `TriggerSource` feature specifies which internal or physical signal is to be interpreted as a trigger. The list of possible sources is long and varies depending on the camera. Some samples are provided below:

- `Software`
  A software command sets the trigger. The PLC function blocks FB_VN_GevCameraControl [▶ 1621] and FB_VN_SimpleCameraControl [▶ 1646] provide the method TriggerImage [▶ 1645] for sending the command.
- `Line0, Line1, ...`
  A physical camera input is the trigger source. Pay attention to the necessary edge steepness with the signal source connected to it. This may vary depending on the camera.

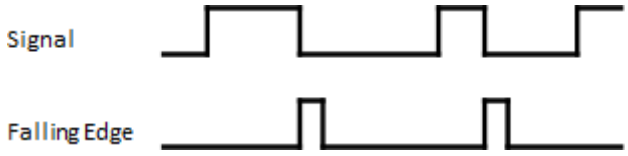
- `Encoder0, Encoder1, ...`
  Line scan cameras are often connected to an encoder to trigger image acquisition. If no physical encoder is available, but the corresponding axis values are available in TwinCAT, the encoder signal can be output via an EL2521 or EL2522.

Trigger Activation

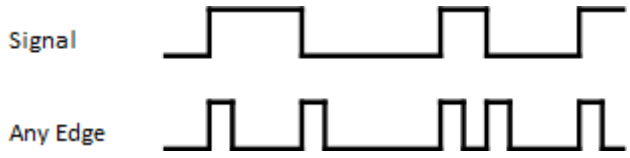
- `RisingEdge`
  The trigger is activated with a rising edge of the `TriggerSource`.



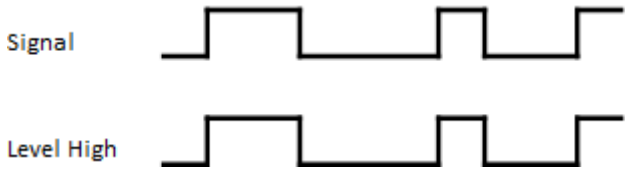
- `FallingEdge`
  The trigger is activated with a falling edge of the `TriggerSource`.



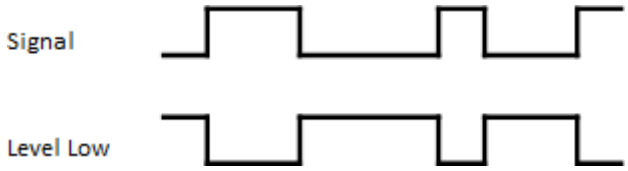
- `AnyEdge`
  The trigger is activated with a rising or falling edge of the `TriggerSource`.



- `LevelHigh`
  The trigger is activated as long as a signal is present at the `TriggerSource`.



- `LevelLow`
  The trigger is activated as long as no signal is present at the `TriggerSource`.



8.4.3.1 Continuous image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the `AcquisitionMode` is set to `Continuous`.

The other cases are described under:

- Multi-image acquisition [▶ 2852]
- Single image acquisition [▶ 2854]

Checking the features of the camera

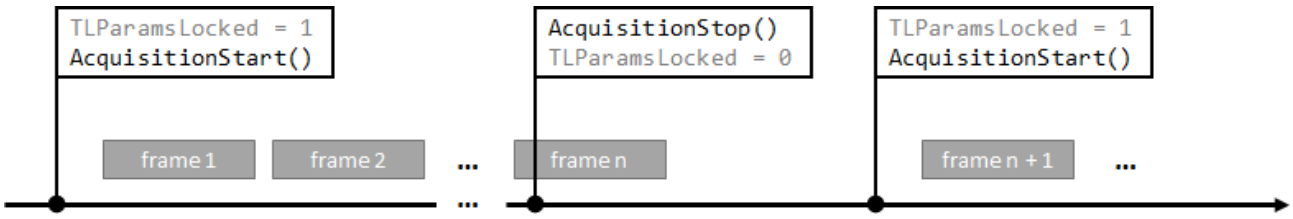
Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Continuous image acquisition

The following configuration is required if the camera is to stream, i.e. take images at even intervals without further triggers:

- `AcquisitionMode = Continuous`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = Off`

The `AcquisitionStart` command then starts the streaming and the `AcquisitionStop` command ends the streaming. Recording of the current image is completed, and the image is transferred.

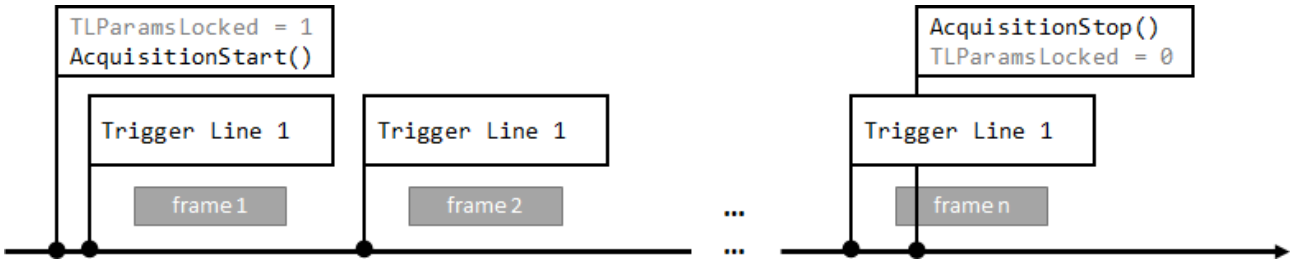


Single image trigger for continuous image acquisition

If the camera is to capture just one image after a trigger signal, the following configuration is required:

- `AcquisitionMode = Continuous`
- `TriggerSelector = FrameStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

As in the other cases, the camera is ready to take images after the `AcquisitionStart` command. Each image is recorded after a trigger signal from the selected source. Details of the trigger signal are specified in `TriggerActivation`.

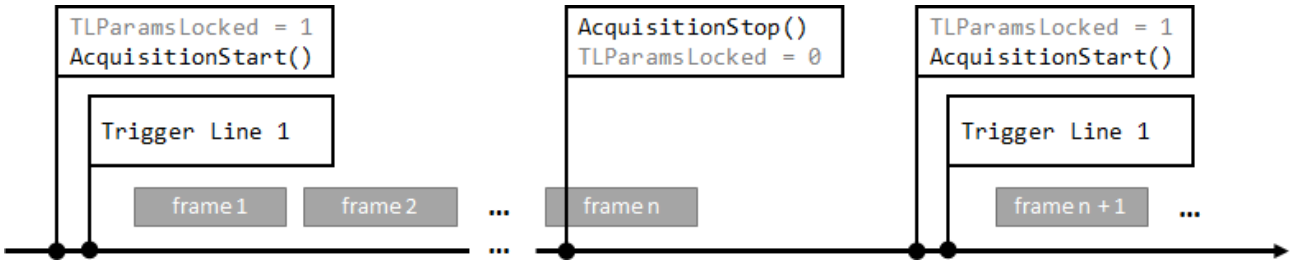


Continuous image acquisition after trigger start signal

If you require the camera to continuously stream images after a trigger signal at even intervals, the following configuration is required:

- `AcquisitionMode = Continuous`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

After the `AcquisitionStart` command, the camera is ready to capture and transfer images. However, the recording only begins after a trigger signal, in the sample after a hardware trigger signal on Line 1. Streaming is ended by the `AcquisitionStop` command; however, the recording of the current image is still completed and transferred.

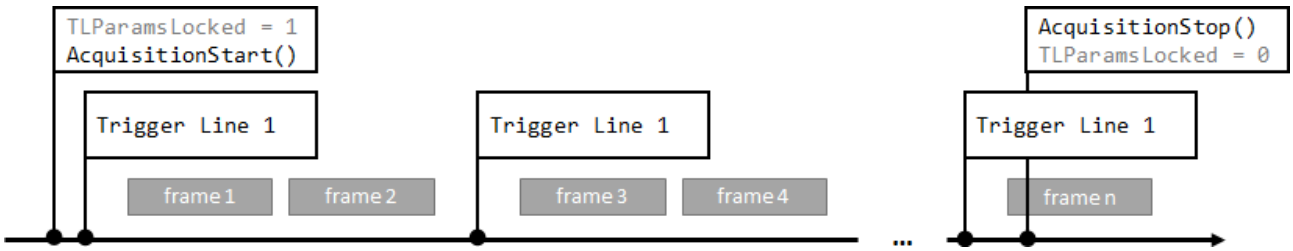


Multi-image trigger for continuous image acquisition

If the camera is to record a certain number of images after a trigger signal, the following configuration is required. The number is set to 2 as an example:

- `AcquisitionMode = Continuous`
- `AcquisitionBurstFrameCount = 2 (exemplarisch)`
- `TriggerSelector = FrameBurstStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

As in the other cases, the camera is ready to take images after the `AcquisitionStart` command. The defined number of images is recorded after a trigger signal of the selected source. Details of the trigger signal are specified in `TriggerActivation`. If the `AcquisitionStop` command is issued during image acquisition/transfer, the image that has already been started is completed, but any missing images are no longer created.



8.4.3.2 Multi-image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the `AcquisitionMode` is set to `MultiFrame`.

The other cases are described under:

- Continuous image acquisition [▶ 2850]
- Single image acquisition [▶ 2854]

Checking the features of the camera

Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

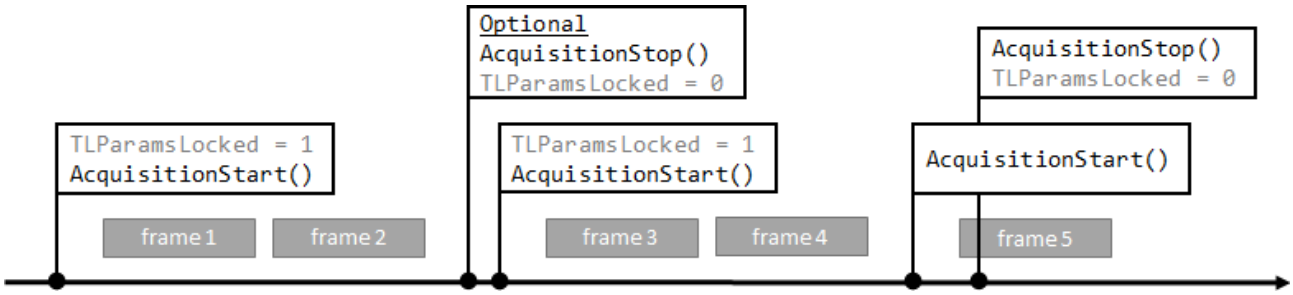
Multi-image acquisition

If the camera is to stream a certain number of images without using a trigger, the following configuration is required:

- `AcquisitionMode = MultiFrame`

- AcquisitionFrameCount = 2 (example)
- TriggerSelector = AcquisitionStart
- TriggerMode = Off

Streaming starts when the AcquisitionStart command is issued. It stops it when the specified number of frames (AcquisitionFrameCount) is reached. Optionally, the AcquisitionStop command can then be issued and the lock can be canceled. This makes it possible to change camera features before the next AcquisitionStart command. If the AcquisitionStop command is issued during image acquisition/ transfer, the image that has already been started is completed, but any missing images are no longer created.

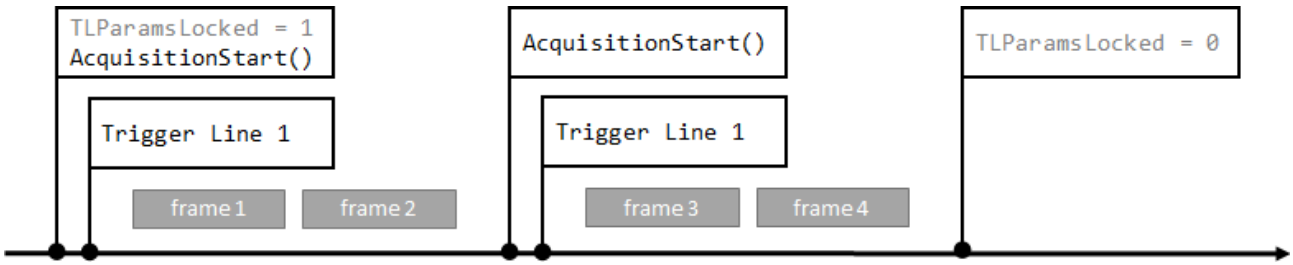


Multi-image acquisition after trigger start signal

If the camera is to stream a certain number of images after a trigger signal, the following configuration is required:

- AcquisitionMode = MultiFrame
- AcquisitionFrameCount = 2 (example)
- TriggerSelector = AcquisitionStart
- TriggerMode = On
- TriggerSource = Line1 (example)

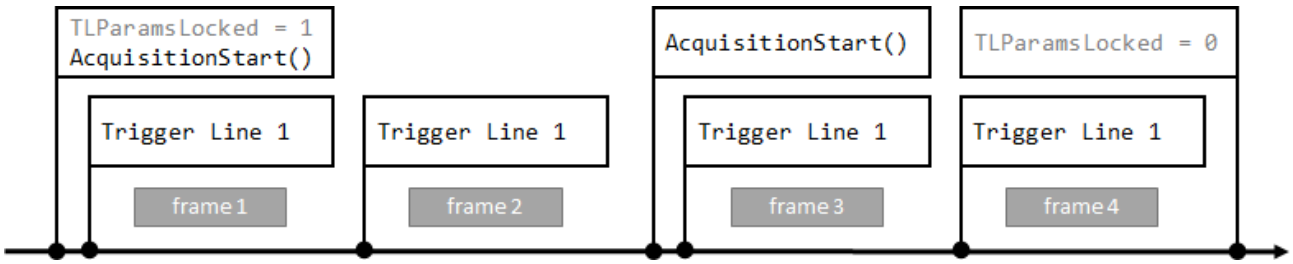
The AcquisitionStart command activates streaming. However, the recording of the specified images only begins after a trigger signal, in the sample after a hardware trigger signal on Line 1. As with the multi-image acquisition (without trigger), the AcquisitionStop command and the cancelation of the lock can optionally take place after reaching the specified number of images.



Multi-image acquisition with single-image trigger

Configuration:

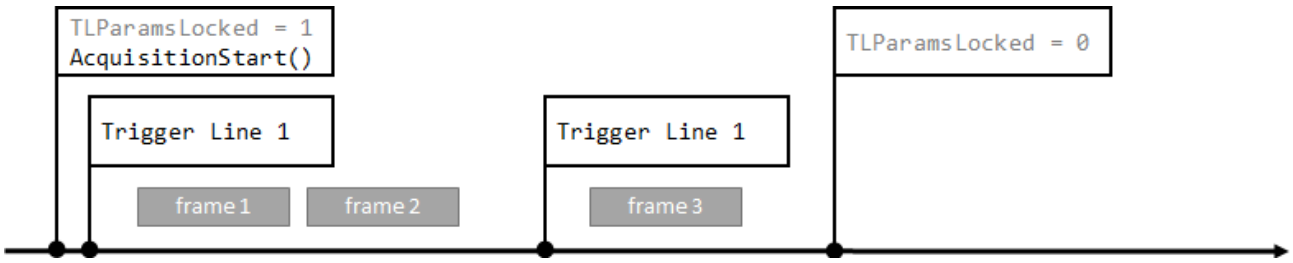
- AcquisitionMode = MultiFrame
- AcquisitionFrameCount = 2 (example)
- TriggerSelector = FrameStart
- TriggerMode = On
- TriggerSource = Line1 (example)



Multi-image acquisition with multi-image trigger

Configuration:

- `AcquisitionMode = MultiFrame`
- `AcquisitionFrameCount = 3 (example)`
- `AcquisitionFrameBurstCount = 2 (example)`
- `TriggerSelector = FrameBurstStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (example)`



8.4.3.3 Single image acquisition

The settings for image acquisition and triggering are interlocked by default. The following explanatory section assumes that the `AcquisitionMode` is set to `SingleFrame`.

The other cases are described under:

- Continuous image acquisition [▶ 2850]
- Multi-image acquisition [▶ 2852]

Checking the features of the camera

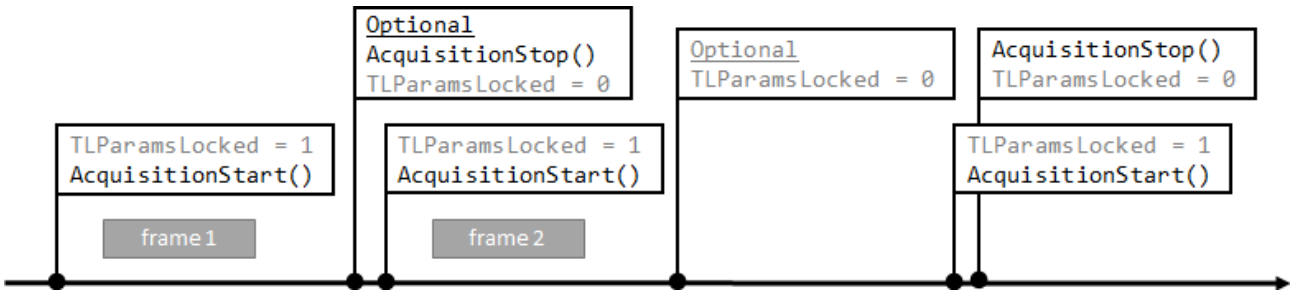
Please refer to the manual for the camera to see whether the camera supports corresponding features. The same applies to the feature names that are actually used. The procedure using the feature names according to the GenICam Standard Features Naming Convention (Version 2.4) is described below.

Single image acquisition

If just one image is to be recorded after each `AcquisitionStart` command, the following configuration is required:

- `AcquisitionMode = SingleFrame`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = Off`

After the `AcquisitionStart` command, just one image is captured. Between two `AcquisitionStart` commands an `AcquisitionStop` command can be issued, and the lock can be canceled. Unlocking facilitates changing camera features between `AcquisitionStart` commands. No image is captured if an `AcquisitionStop` command is issued too soon after the `AcquisitionStart` command, when image acquisition has not yet started.



Single image acquisition with image acquisition trigger or single image acquisition with single image trigger

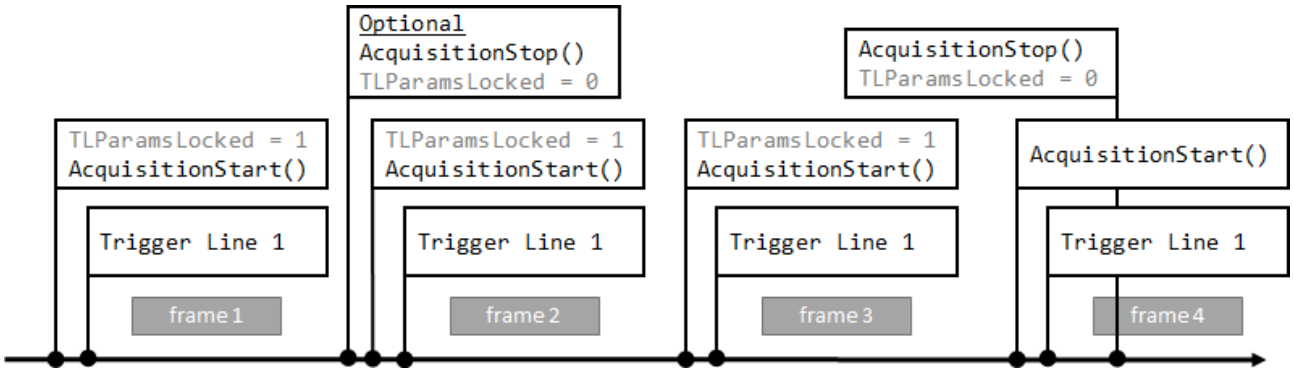
The following configurations can be used if just one image is to be recorded after each `AcquisitionStart` command in situations where a trigger signal is required:

Configuration – single image acquisition with image acquisition trigger

- `AcquisitionMode = SingleFrame`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (example)`

Configuration – single image acquisition with single image trigger

- `AcquisitionMode = SingleFrame`
- `TriggerSelector = FrameStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (example)`



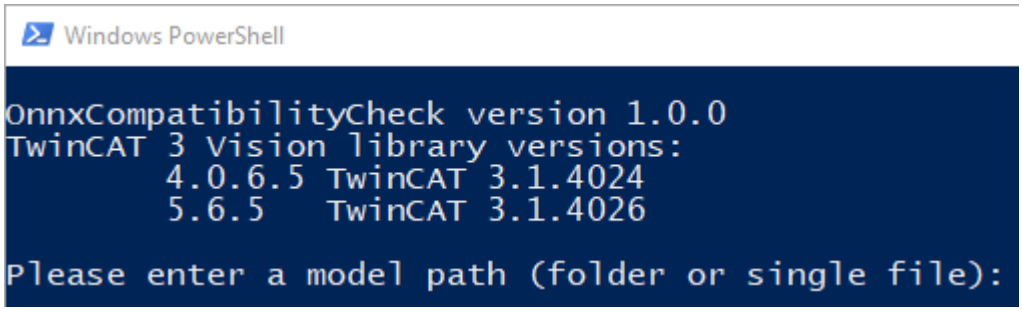
8.5 Neural Network Compatibility Check

This tool offers the possibility to check the compatibility of ONNX models with the functions of the product "TF7810 | TwinCAT 3 Vision Neural Network". The "TF7810_NeuralNetworkSupplements.zip" can be downloaded directly from the Beckhoff website.

The package contains the executable file "OnnxCompatibilityCheck.exe", the Power Shell script "CheckModelCompatibility.ps1" for simplified use and a folder "ONNX_Samples" with sample models.

Execution of the Power Shell script

After executing the script, the version information is displayed first. First from the application itself and then to the TwinCAT 3 Vision library versions to which the compatibility refers. As a user, you will be asked to specify a folder or model path. If a folder containing several ONNX models is specified, these are tested in alphabetical order. It is also possible to explicitly specify a model, e.g. "C:\ONNX_Samples\LemonModel.onnx".



A successful execution looks as follows, e.g. using the Lemon Sample model:

```
Please enter a model path (folder or single file): C:\ONNX_Samples\LemonModel.onnx

Successfully loaded 'LemonModel.onnx'
Model input shape: [1, 3, 224, 224], TCVN_DL_4D_NCHW
Model outputs: 1
Layer: '367' [1, 3]
The model can be executed with TwinCAT Vision.
```

At the end, the execution is paused so that you can view the output.

Options of the Power Shell script

When the user opens the script for editing, he will find the parameters $modelPath and $inputShape in lines 2 and 3. The first parameter $modelPath is empty by default, so that the user is prompted to enter the path. Alternatively, a folder or model path can be stored in the script. In this case, the user is not prompted to make an entry and the stored path is used directly.

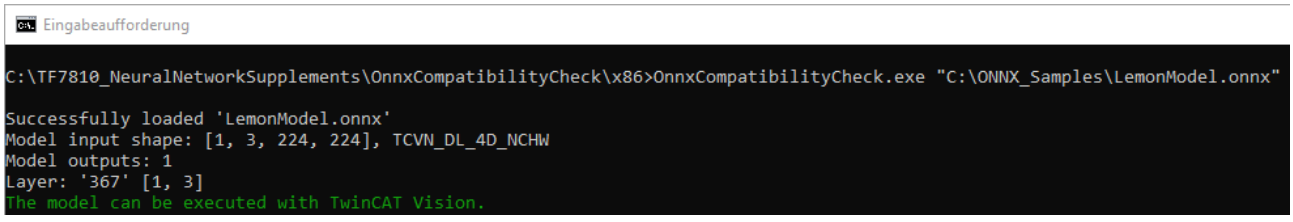
CheckModelCompatibility.ps1

```
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"
4  )
```

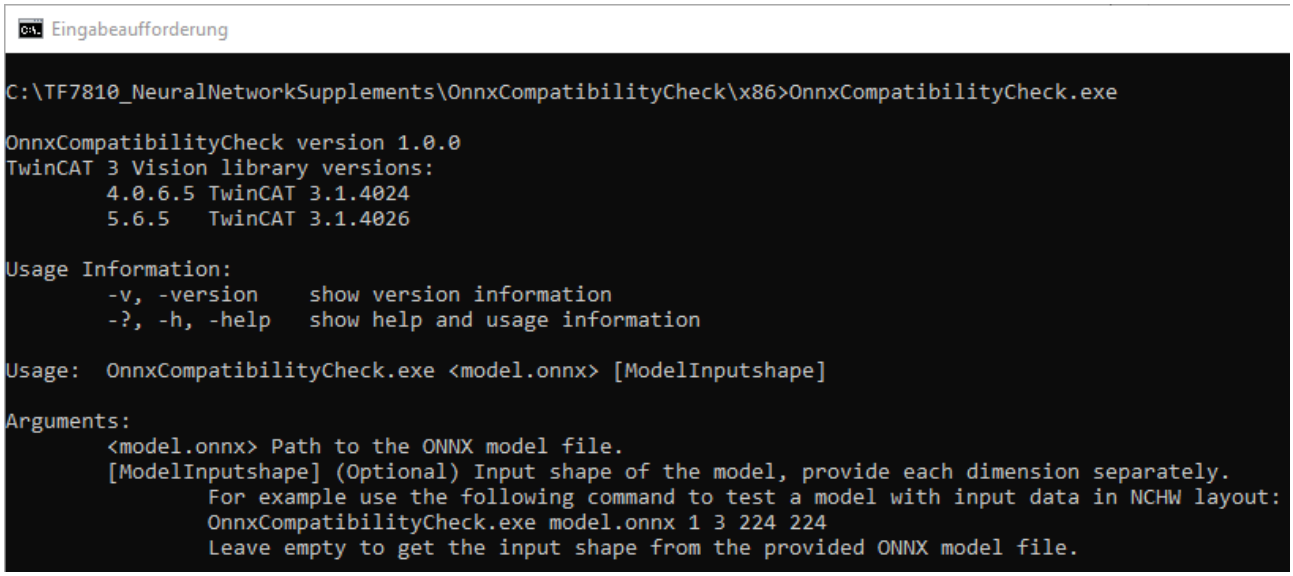
By default, the word "skip" is stored for the second optional $inputShape parameter, so that this query does not appear and the application is executed without the option. A fixed value can also be entered here, as in the comment with "1 3 224 224". Alternatively, the word "skip" can be deleted so that the user is prompted to enter it. This prompt for the "Model Input Shape" can also be skipped by pressing "Enter" when executing the script.

Application of the "OnnxCompatibilityCheck.exe"

The "OnnxCompatibilityCheck.exe" can be executed directly without a Power Shell script. In the Windows command prompt, the successful test of the Lemon Sample model looks like this, for example:



If only the name of the application is specified or -?, -h, -help is added and then confirmed with Enter, the following information and notes on the application are displayed.



Results

The following describes the results that can be output in different scenarios:

- At the start of execution, the system checks whether the path is valid and the file is in ONNX format. If this is not the case, execution is aborted and a corresponding message is displayed.
- If the ONNX model could be loaded successfully, this is displayed as the first information, including the file name.
- This is followed by information on the input and output layers of the model, which are read from the ONNX file:
  - The shape of the tensor is shown in square brackets for the input and output layers. The number of elements in the respective dimension is displayed.
  - If the input layer has more than two dimensions, the ETcVnElementType for the conversion (F_VN_ConvertDataLayout) from a 2D image to the respective input format of the model is also output. As a prerequisite for the correct detection of the ETcVnElementType, it is assumed that N (number of samples or batch size) = 1 and the number of channels is smaller than the width or height.
  - For the input layer of the Lemon Sample model, for example, this looks like this: "Model input shape: [1, 3, 224, 224], TCVN_DL_4D_NCHW".
  - If there is more than one input layer, their number and the respective shape of the tensor is displayed. As the execution of models with more than one input layer is not supported, the following message appears: "Not supported: Only models with a single input can be executed".
  - For the output layers, the number is output first, followed by a list of the layers with their names and the shape of the tensor.
  - For the output layer of the Lemon Sample model, for example, it looks like this:
    "Model outputs: 1
    Layer: '367' [1, 3]"
- Neural networks can either be designed for a fixed (static) input format or offer the possibility of processing several different input formats (dynamic). For most models, the input layer is fixed and the values of the tensor are stored in the ONNX file.
  The values can then be read, displayed and used directly for testing.
- If models support several input / image sizes, the information on the dimensions, such as the width and height of the image in the ONNX file, is missing in this case. This is referred to as a dynamic shape of the input layer. If this shape of the input layer is recognized, the following message is displayed:
  "Model input shape: dynamic
  To test the execution, please provide the optional [ModelInputshape]
  (e.g. OnnxCompatibilityCheck.exe DynamicShape.onnx 1 3 224 224)."
- In order to be able to test a model with a dynamic input layer, the values for the "Model Input Shape" can also be transferred when the application is called up. A sample of the call and the transfer of the values is also output or can be seen in the help.

- There are models in which the input layer is fixed in the ONNX file, but which can also be executed with other values. If the user has this information or knows the other values, they can also use the call with the additional specification of the "Model Input Shape".
- The information on the supported input formats can usually be found in the documentation of the model provider, the source of the model or in the associated research article. If this information is not explicitly specified, it can also be derived by examining the model architecture or by using tools to inspect the ONNX file.
- In the event that a model cannot be executed, an attempt is made to output a possible cause in addition to the basic information. If the cause is not clear, no additional information is displayed. Possible causes are, for example, unknown or unsupported operators / data types or a newer ONNX "opset number". If an operator is named as the cause, this can also be a subsequent problem. Therefore, the operators of the previous layer should also be analyzed.

9 Appendix

In this chapter you will find additional information such as FAQs, an overview of the paths used by TwinCAT Vision and a chapter on troubleshooting for the rectification of known problems.

9.1 FAQ

What is TwinCAT Vision?

TwinCAT Vision is an industrial image processing software. You can capture images with industrial cameras and calculate information about the recorded scene from them.

Frequent use cases are:

- Presence check
- Object recognition
- Optical measurement
- Code Reading

Can I test TwinCAT Vision free of charge?

Yes, you can generate 7-day trial licenses for the runtime components. The engineering is free of charge.

Do I need TwinCAT in order to use TwinCAT Vision?

Yes, TwinCAT Vision is fully integrated in TwinCAT. A compatible TwinCAT version must be installed before TwinCAT Vision can be installed.

What cameras can I use with TwinCAT Vision?

TwinCAT Vision supports cameras with the GigE Vision interface. GigE Vision cameras are connected to the IPC via an Ethernet cable.

Can I use several cameras at the same time?

Yes, with the TwinCAT GigE Vision Connector you can operate several cameras, depending on the license.

Can I connect several cameras via a single switch?

Yes. Pay attention to the maximum available bandwidth (e.g. 125 MB/s in case of 1 GigE, effectively nearer 110-115 MB/s).

Are there samples for TwinCAT Vision?

Yes, in the chapter Samples [▶ 2753].

Are there benchmarks for the TwinCAT Vision functions?

No, because the execution time of vision algorithms strongly depends on the performance of the IPC, the image size, the function parameters and the image content. Therefore, it is helpful to have a basic understanding of how algorithms work and what factors influence the respective computing time.

Can other image processing software also be used in TwinCAT as an alternative to TwinCAT Vision?

Images can be recorded via GigE Vision with the TC3 GigE Vision Connector. The image processing can then take place in various ways:

- Use of the TwinCAT Vision API via the PLC library Tc3_Vision
- Use of the self-written TwinCAT PLC or C++ algorithms
- Using the TwinCAT Vision API in a C++ module

- Use of Matlab/Simulink
- Retrieving image data via ADS from other applications

What is the difference between image processing software such as TwinCAT Vision and smart cameras?

Image processing software such as TwinCAT Vision:

- Image data received from a camera
- Process this on a computer
- Send processed information to the controller
- Advantage of TwinCAT Vision: control and image processing are the same software and run in the same system. Therefore, additional interfaces and protocols are not required.

Smart cameras:

- Have an internal processor
- Carry out image processing operations directly on the camera
- Supply processed information such as good/bad information or the position of a workpiece via additional interfaces

Can TwinCAT Vision be executed on a GPU?

No, TwinCAT Vision runs exclusively on a CPU in the TwinCAT real-time environment. Alternatively, TwinCAT Vision offers the option of an automatically parallelized execution by using the multi-core functionality of TwinCAT (in interaction with job tasks [▶ 64]).

9.2 Important paths

Paths for Vision data

NOTICE

Paths that are relevant for data

Make sure that the following standard paths are not affected by Write filters or similar programs. Failure to do so may result in data loss. If the corresponding path settings were adjusted manually, the same applies to the set paths.

Use	Path
Storage location for File Source images	C:\Users\Public\TcVision\FileSources\<UniqueId> (Automatically defined by the File Source. <UniqueId> is different for every project.)
Storage location for camera streams	C:\Users\Public\TcVision\CameraStreams (May differ depending on the setting in Record/Playback [▶ 101].)
Storage location for images that are read or written by PLC functions.	C:\Users\Public\TcVision\Images (May differ depending on the setting in Service Configuration [▶ 66].)
Storage location for containers that are read or written by PLC functions.	C:\Users\Public\TcVision\Containers (May differ depending on the setting in Service Configuration [▶ 66].)
Storage location for ML models that are read or written by PLC functions.	C:\Users\Public\TcVision\MlModels (May differ depending on the setting in Service Configuration [▶ 66].)
Storage location for calibration results that are read or written by PLC functions.	C:\Users\Public\TcVision\CalibrationResults (May differ depending on the setting in Service Configuration [▶ 66].)
Storage location for individual calibration patterns	C:\Users\Public\TcVision\CalibrationPattern

Paths for log files

Log file	Path
TwinCAT Vision installation log (all other TwinCAT installation logs are also located in this folder)	C:\Users\<*User*>\ TF7xxx-Vision.exe.log From Build 4026: C:\ProgramData\Beckhoff\TcPkg\logs\ The individual product log files are located in the MSI subfolder.
TwinCAT Vision Service Log	C:\ProgramData\Beckhoff\TcVnService\TcVnService.log
General TwinCAT Vision log	C:\ProgramData\Beckhoff\Vision (May vary depending on the setting in Logging [▶ 68])
Result images of the Calibration Assistant [▶ 107] for analyzing the recognition of feature points on the calibration pattern	C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput

9.3 Troubleshooting

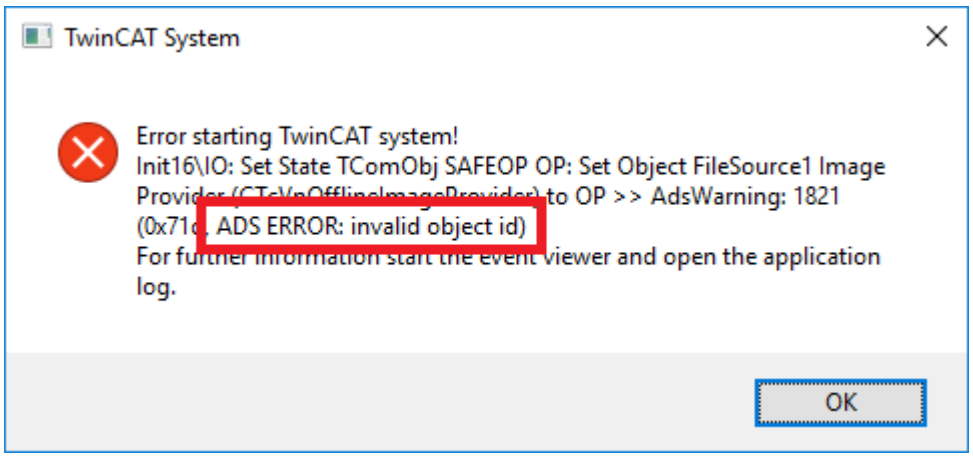
Known error situations and corresponding solutions are described below. If you can attribute a problem to a specific function or function block, first read the corresponding entry in the API reference [▶ 137]!

9.3.1 TwinCAT system start

Solutions to situations in which the start of the TwinCAT system is prevented are explained below.

9.3.1.1 TwinCAT system does not start

If the TwinCAT system cannot be started, the following message frequently appears. On the basis of the error code with the red border, instructions for rectifying the problem can be given. Further below you will find information on the usual error codes with TwinCAT Vision applications.



Invalid Object ID

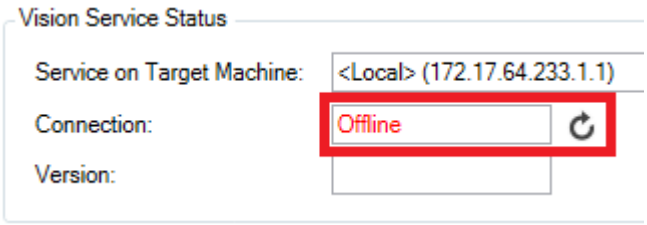
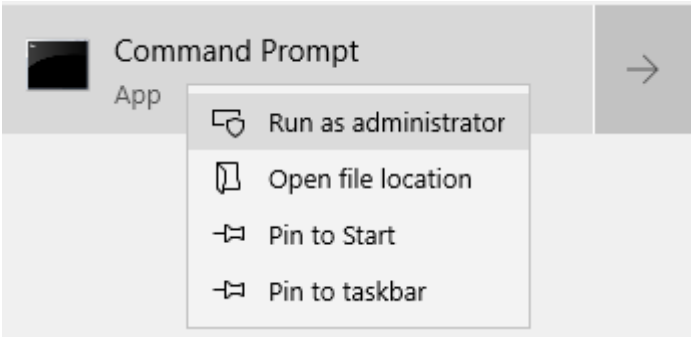
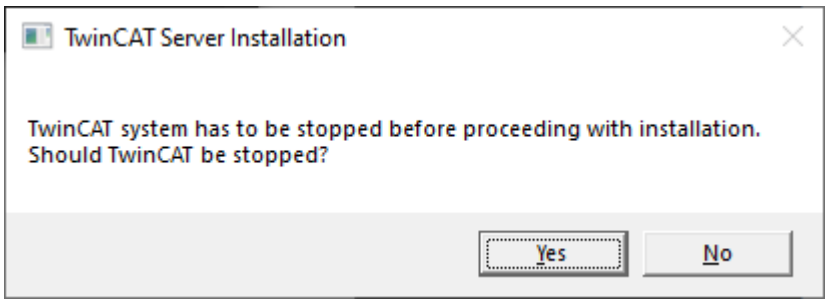
Error message	ADS ERROR: invalid object id
Solution 1	**No task linked** 1. Check whether the corresponding tasks have been created for each Vision device used and add missing tasks. ◦ Check whether a task for image acquisition has been created for each File Source Control (default cycle time 10 ms). ◦ Check whether each GigE Vision camera is associated with a task for Image Acquisition (default cycle time 1 ms) and, if there is an Ads Communicator, this is also associated with a task (default cycle time 10 ms). 2. The tasks must be assigned to the corresponding TcCom modules of the Vision devices. Check the assignment and correct it if necessary. ◦ Check whether the Image Provider Task of the File Source is linked to the Image Provider of the File Source Control. ◦ Check whether the Image Acquisition Task of the camera is linked to the Image Acquisition and Image Acquisition Simulation of the camera. ◦ Link the Ads Communicator Task with the Ads Communicator of the camera. ◦ Do not assign a task to the Image Provider of the camera.
Solution 2	**Network adapter invalid** The network adapter was not adapted, e.g. following a system change. Adapt the network adapter to the current target system. In the respective network adapter, open the **Adapter** tab and select a new compatible adapter with **Search…**
Solution 3	**Network adapter deleted** The network adapter was deleted, for example, on the assumption that it is not required for the simulation of the camera. However, each camera object always requires a network adapter; if necessary it can be deactivated but not deleted. Create a new network adapter with an IpStack and link the IpStack on the **Interface Pointer** tab of the acquisition object of the camera.

Further information on the linking of Vision TcCOM objects can be found in the corresponding chapters for the camera [▶ 123] and the File Source [▶ 130].

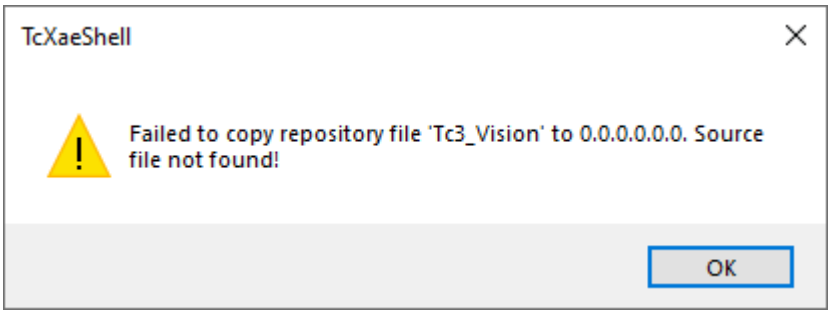
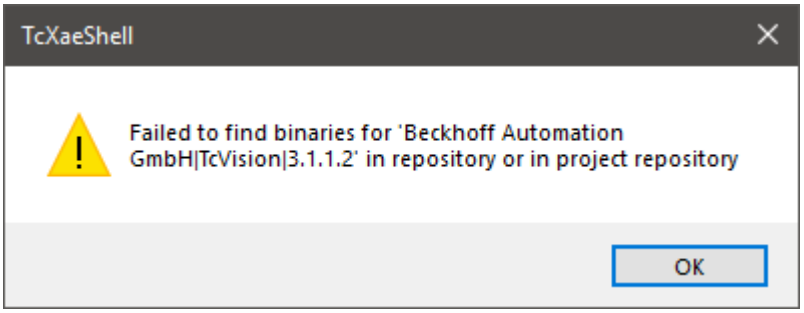
General ADS Error

Error message	ADS ERROR: General ADS Error
Solution 1	**More cameras than configured** The default setting of the network adapter allows up to 2 cameras to be operated simultaneously. If you wish to operate more cameras on the same adapter, increase the Init parameters `ipMaxReceiver` and `UdpMaxReceiver` to twice the number of connected cameras (see Configuration of the network adapter [▶ 56]).

9.3.1.2 Vision Service doesn't start

Situation	The TwinCAT Vision Service doesn't start. This can be seen by the connection status "Offline" on the **Service Configuration** tab in the VISION node. 
Solution	If the TwinCAT Version Service only fails to start once or fails, it can be reactivated with a TwinCAT restart (click ). However, if it generally fails to start (after the system startup), the service must be re-registered as a TwinCAT service. To do this, follow these steps: 1. Switch TwinCAT to the Config mode  2. Open a prompt as an administrator.  3. Switch to the corresponding folder via "`cd C:\TwinCAT\3.1\Components\Vision\TcVision`". 4. Execute the "`TcVnService.exe /tc3serverreg`" command to start the Vision Service. 5. Confirm the subsequent dialog by clicking **Yes**.  ⇨ The Vision Service is started. **TwinCAT must be in Config mode so that the Vision service can be started. Therefore, the machine must be restarted if the Vision service is to be restarted.** If the Vision service is to be manually stopped, you can use the command "`TcVnService.exe /unregtc3server`".

9.3.1.3 Binary or repository could not be found/loaded

Situation	One of the two messages appears when the configuration is activated. Depending on the cause and the installed version, other file names or version numbers may appear in the message. The reason for the message is that the version used in the project does not match the version installed on the system.  
Solution 1	**Checking the Vision Library** Check the version of the Tc3_Vision library included in the project. If the version does not match the one in the folder C:\TwinCAT\3.1\Repository\Beckhoff Automation GmbH\Tc3_Vision of the engineering system, either update the library in the project or install another setup version.
Solution 2	**Checking the TMC version of the camera or FileSorce** Check under **SYSTEM** > **TcCOM Objects** > **Project Objects** the TMC version of the listed objects used in the project. If the version is highlighted in yellow, the specified version does not exist on the system or does not match. Now either update the corresponding TMC versions (select the version via pull-down of the version, then right click on one of the object IDs in the first column and select **Reload TMI/TMC Description(s) with changed version**) to the available version or install another setup version.
Solution 3	**The vision component is not available** The corresponding repository file is missing on the current system because the vision component was not selected during installation. Basically, the versions of the engineering and target systems must match. Therefore, if the systems are different, check both installations. On pure runtime systems XAR, only the Vision Service can be installed since all other files are transferred by the engineering system XAE. Install the missing vision component or run the setup with the repair option.

9.3.2 PLC runtime environment

Solutions for situations in which errors occur when the PLC is running are explained below.

9.3.2.1 Vision device in the ERROR state

Camera function blocks switch between the ERROR and INITIAL state

Situation	The instance of the function block FB_VN_Gev_CameraControl [▶ 1621] or FB_VN_SimpleCameraControl [▶ 1646] alternately returns TCVN_CS_ERROR and TCVN_CS_INITIAL on calling the method GetState.
	In the TCVN_CS_ERROR state, the Reset() method of the function block is executed successfully and the camera instance changes to the TCVN_CS_INITIAL state. If the camera remains in the TCVN_CS_ERROR state, refer to Device is permanently in ERROR state [▶ 2867].
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_ERROR THEN`
	`  fbCamera.Reset();`
	`END_IF`
	
	In the TCVN_CS_INITIAL state, either function block method OpenCamera() or StartAcquisition() is called. The call fails and the instance returns to state TCVN_CS_ERROR.
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_INITIAL THEN`
	`  fbCamera.OpenCamera();`
	`END_IF`
	or
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_INITIAL THEN`
	`  fbCamera.StartAcquisition();`
	`END_IF`

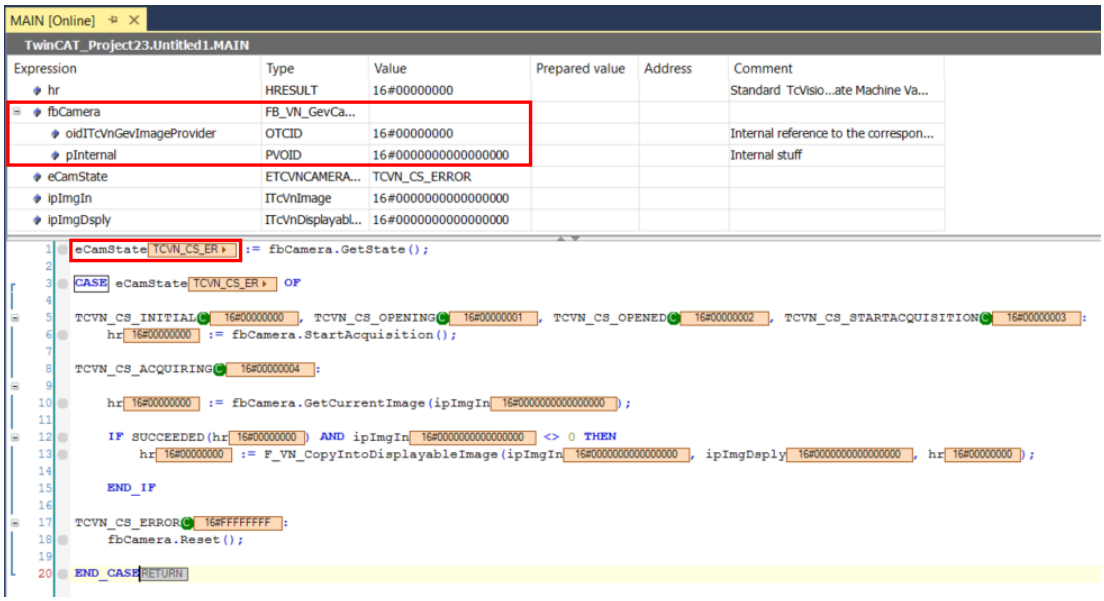
Solution 1	Check the physical connection to the camera and the power supply of the camera.
Solution 2	Check the settings for the camera connection (IP address of the camera, network port).

Function block permanently in the ERROR state

Situation

The instance of the function block `FB_VN_GevCameraControl`, `FB_VN_SimpleCameraControl` or `FB_VN_FileSourceControl` permanently returns the state `TCVN_CS_ERROR`. Even when calling the function block method Reset, there is no change of state to `TCVN_CS_INITIAL` (if there is, see Device switches between ERROR and INITIAL state [▶ 2866]).

```
fbCamera.Reset();
```

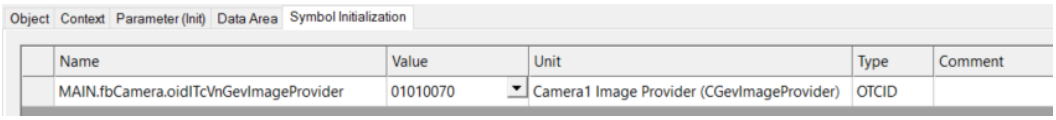


Solution 1

The symbol instance must be linked to the Image Provider - each instance of a `FB_VN_GevCameraControl`, a `FB_VN_SimpleCameraControl` and a `FB_VN_FileSourceControl` must be linked to a camera or File Source Control created in the Vision node.

Proceed as follows:

1. Perform a successful rebuild of the PLC project.
2. Then double-click on the instance of the PLC project.
3. Under the **Symbol Initialization** tab, assign the Image Provider of the corresponding camera or the corresponding File Source Control to the instances of `FB_VN_GevCameraControl`.
4. Reactivate the configuration.



The FB_VN_GevCameraControl can only be linked to the Image Provider of a camera.

The FB_VN_FileSourceControl can only be linked to the Image Provider of a File Source Control.

The FB_VN_SimpleCameraControl can be linked to the Image Provider of a camera or to the Image Provider of a File Source Control.

Solution 2

The instance of `FB_VN_GevCameraControl` or `FB_VN_SimpleCameraControl` or `FB_VN_FileSourceControl` is assigned the Image Provider of a camera or a File Source Control, yet the device is permanently in error state.

Check the method calls of the instance of `FB_VN_GevCameraControl` or `FB_VN_SimpleCameraControl` or `FB_VN_FileSourceControl`. Many methods may only be called in a certain state. A detailed description of the states of Vision devices can be found in the section Function blocks for image acquisition [▶ 1611].

- In all states

- GetState
- GetCalibPatternRef (from FB_VN_GevCameraControl)
- GetCameraMatrix (from FB_VN_GevCameraControl)
- GetDistortionCoefficients (from FB_VN_GevCameraControl)
- GetRotationMatrix (from FB_VN_GevCameraControl)
- GetTranslationVector (from FB_VN_GevCameraControl)
- SetCameraMatrix (from FB_VN_GevCameraControl)
- SetDistortionCoefficients (from FB_VN_GevCameraControl)
- SetRotationMatrix (from FB_VN_GevCameraControl)
- SetTranslationVector (from FB_VN_GevCameraControl)

• In the INITIAL state and in the INITIALIZING state
- InitializeCamera (from FB_VN_GevCameraControl)

• In INITIAL state and in OPENING state
- OpenCamera (from FB_VN_GevCameraControl)
- StartAcquisition

• In OPENED state and in STARTACQUISITION state
- StartAcquisition

• In OPENED state and in CLOSING state
- CloseCamera (from FB_VN_GevCameraControl)

• In ACQUIRING state
- GetCurrentImage
- GetCurrentImageUndistorted (from FB_VN_GevCameraControl)
- GetCurrentImageWithGvspInfo (from FB_VN_GevCameraControl)
- GetCurrentImageAndFileName (from FB_VN_FileSourceControl)
- ClearImageQueue (from FB_VN_GevCameraControl)
- GetLastImageFromQueue (from FB_VN_GevCameraControl)
- GetOmittedImagesNum (from FB_VN_GevCameraControl)

• In ACQUIRING state and TRIGGERING state
- TriggerImage
- TriggerImageExp (from FB_VN_FileSourceControl)
- TriggerImageByName (from FB_VN_FileSourceControl)

• In ACQUIRING state and in STOPACQUISITION state
- StopAcquisition

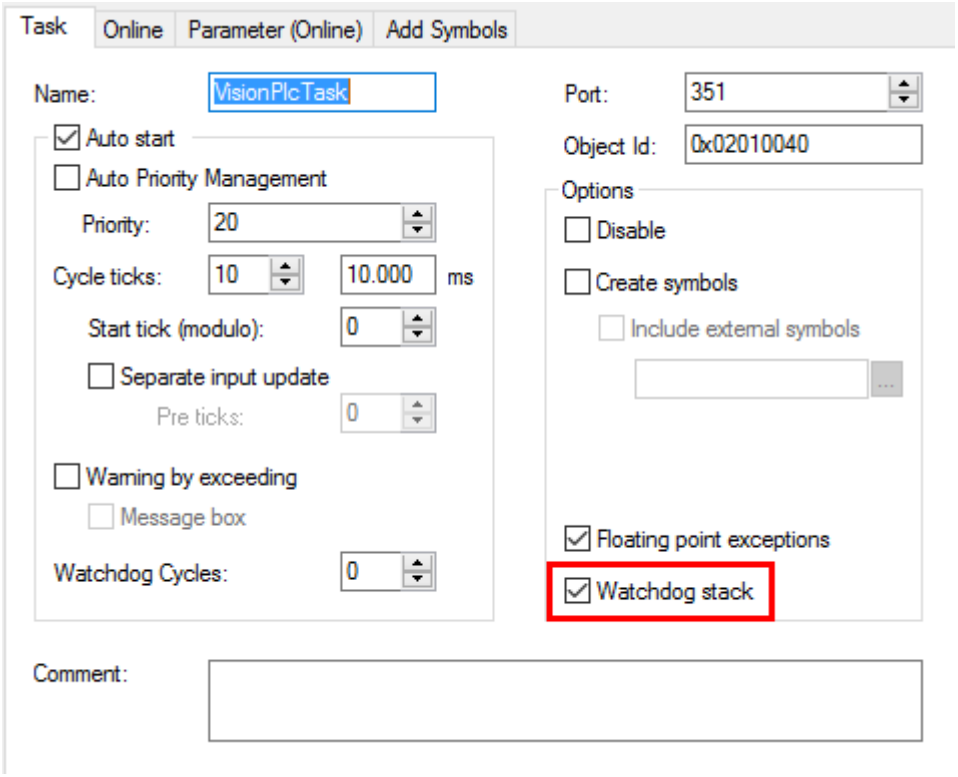
• In ERROR state
- Reset

9.3.2.2 Camera not found after restart

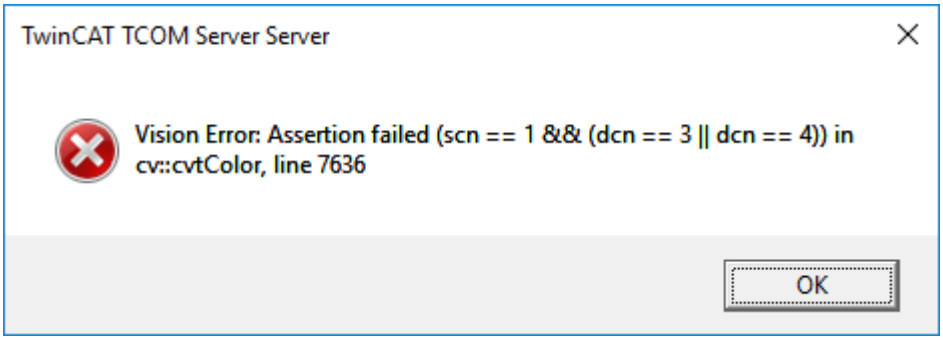
Situation	Application is working entirely as usual. No connection can be established with the camera after restarting the machine.
Solution 1	The IP address has changed. Use the Force-IP option in the Configuration Assistant [▶ 75] of the camera.

9.3.2.3 Error when starting a watchdog

Situation	The function F_VN_StartAbsWatchdog [▶ 990] or F_VN_StartRelWatchdog [▶ 991] permanently returns the error code 16#71A (NOINTERFACE).
	
	`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`

Solution	Activate the watchdog stack in the corresponding PLC task and then re-activate the configuration  . 

9.3.2.4 Vision Error: Assertion failed

Error message	During the execution of a Vision application, an error message similar to the following appears. At the same time, TwinCAT enters the EXCEPTION state.  This message indicates that there is a conflict in the execution of a Vision function that could not be intercepted. This essentially refers to the correct format of the input data and should only occur if the function is not used in accordance with the API reference. The error specification behind "VisionError: Assertion failed" may vary depending on the use case.
Solution with F_VN_ConvertColorSpace [▶ 1241]	An image can be either single-channel or multi-channel. For certain function calls, an image with a defined number of channels is expected. For example with the function `F_VN_ConvertColorSpace`, where the enum `ETcVnColorSpaceTransform` is used to specify the format of the input image. If Engineering is logged in, the corresponding function call is marked in the PLC code. The function `F_VN_GetPixelFormat`, which returns the actual number of channels, can help to analyze why the actual number of channels differs from the expected number. Typical causes: • File Source Control In File Source a different image format is set than expected or the image format is set to "Original Format from File" and the actual image format is different than expected. For example, images that look monochrome are not always stored as monochrome images (single-channel image). • Camera For many cameras, the pixel format can be set in the configuration. Depending on the selected format, the number of image channels also changes. • Previous processing steps A previous function call may have changed the image and thus also the number of channels.

9.3.2.5 Error: SSE Invalid Operation

Error message	You get an error message called **SSE Invalid Operation**.
Solution	This error can occur unnecessarily if the **Floating point exceptions** handling is activated on the executing task. Follow the instructions in the section CPU cores and tasks [▶ 61], by deactivating the option **Floating point exceptions**.

9.3.2.6 File Source

Error state

Error pattern	The File Source constantly switches to the error state when the PLC is running.
Solution 1	**Too little memory** The router memory is set too small. Therefore, the File Source cannot load any image into the PLC and enters the error state when attempting to do so. Enlarge the router memory.

Images from FS Control not in PLC

Error pattern	Images are selected in the File Source Control and the File Source function block in the PLC is in the `ACQUISITION` state. Nevertheless, no images can be retrieved via `GetCurrentImage`.
Solution 1	**BaseDir not set** The **Read from Target** button was actuated for the first time after TwinCAT was activated. In this case, the path from which the File Source images are read has not yet been transferred to the running TcCOM object. Therefore no images can be loaded, even though they are selected in the File Source Control. Reactivate TwinCAT after you have pressed **Read from Target**.
Solution 2	**Trigger mode** The Trigger mode is selected in the File Source Control, but only the method `GetCurrentImage` is called in the PLC, not `TriggerImage`. Trigger an image by calling the method `TriggerImage` in the PLC, or deactivate the Trigger mode in the File Source Control.

9.3.3 Assistants

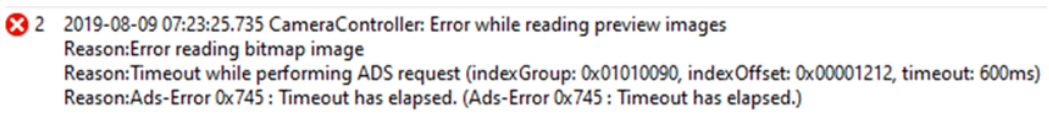
Solutions for situations in which problems occur with the operation of assistants are explained below.

9.3.3.1 Timeouts

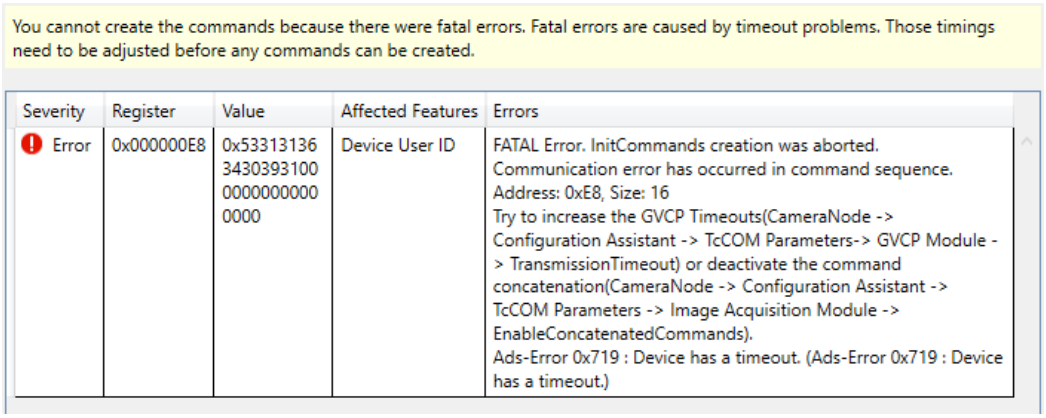
Situation

A timeout error occurred during the communication with a camera. This can happen in different places as illustrated by the following cases.

During image display in the development environment:



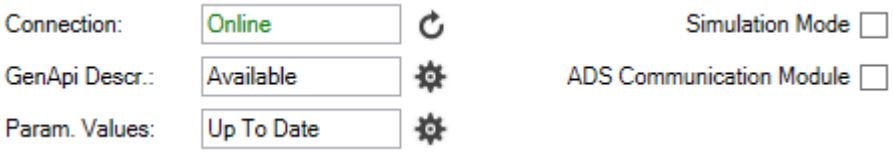
When creating the init commands:



When using the ForceIP command:



When establishing a connection in the camera object:



Solution

In the case of timeout errors, two different types of timeout are essentially relevant: One is the transmission timeout, which limits the communication between the Image Acquisition module and the camera. The other is the ADS communication timeout, which limits the communication between the user interface and the Image Acquisition module.

Adjust the transmission timeout in the TcCOM parameters of the GVCP module in the configuration tree [▶ 78]. Also adjust the number of MaxTimeouts.

Make sure that, if necessary, you adjust the ADS communication timeout in the Settings [▶ 84], as this should be larger than the product of MaxTimeouts and Transmission Timeout.

Gradually increase the times and, if necessary, the number of attempts until no more error messages occur. Make sure, however, that you do not set the times and number of attempts unnecessarily high, as otherwise very long waiting times can occur.

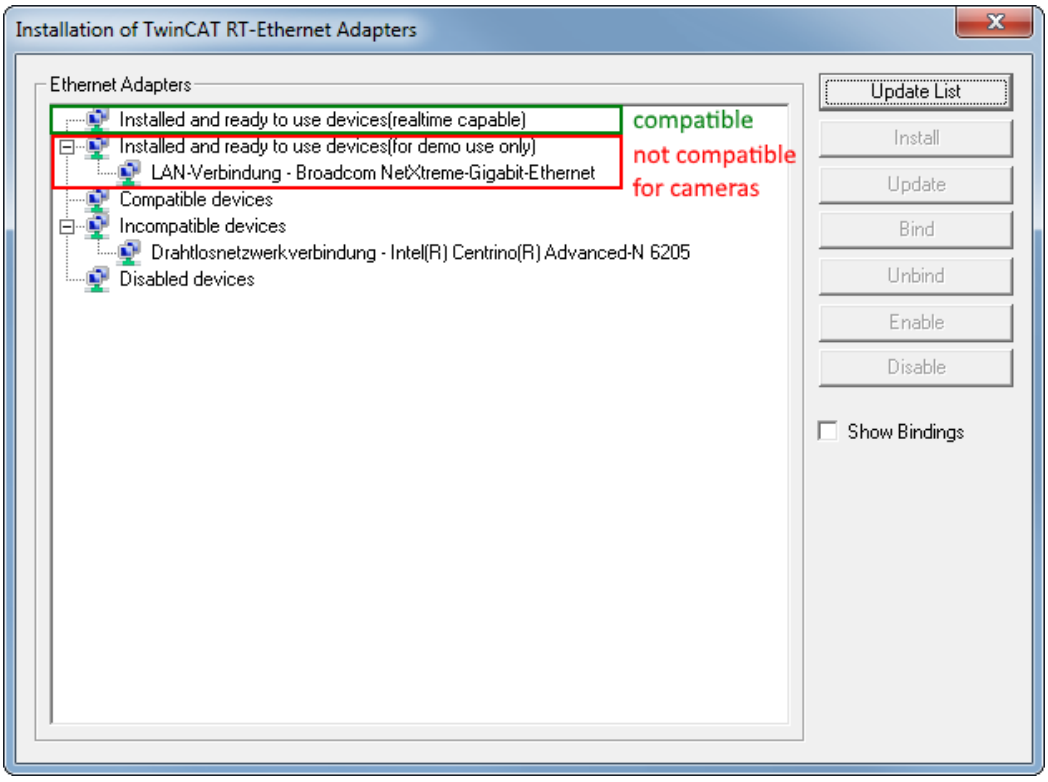
9.3.3.2 Stalling image display

Error pattern	The image display in the ADS Image Watch and camera assistants occasionally stalls.
Solution 1	**Remote Desktop Protocol** In individual cases it is possible that the Remote Desktop Protocol and the network card do not work well with the transfer from the TwinCAT Vision development environment or the image display tools. Work either locally and connect yourself with the target in TwinCAT or work directly on the target computer.

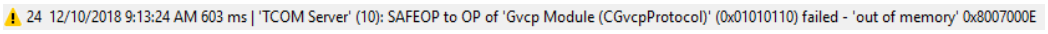
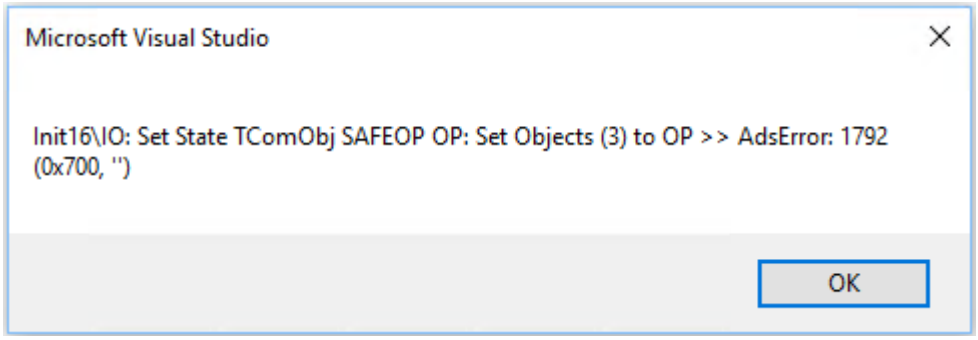
Error pattern	Not all images are displayed in the ADS Image Watch and camera assistants.
Solution 1	**Reduction of the data quantity** As the images are retrieved from the controller by ADS in order to display them, it may be the case in particular with large images, high frame rates or several ADS Image Watch windows that not all image data can be retrieved. Therefore reduce the number of images to be displayed simultaneously, reduce the frame rate or the data quantity per image by using ROIs.

9.3.3.3 Camera not found with Device Discovery

Situation	You are in the process of creating a GigE Vision Camera object. You have successfully set up a TwinCAT RT-Ethernet adapter with IP stack. Nevertheless, the Device Discovery dialog remains empty and there is no error message.
Solution 1	**Wrong network adapter selected**
Solution 2	**No GigE Vision camera** A camera with a Gigabit Ethernet interface does not necessarily have to ‘speak’ GigE Vision. Please pay close attention to the manufacturer's instructions for the transmission protocol.
Solution 3	**Incompatible network adapter** The network adapter used...



9.3.3.4 Error when loading camera objects

Error message	 In configuration mode, the following Visual Studio message appears when the camera modules are reloaded or a camera is to be selected via the Discover Devices button in the Choose Target Camera dialog. 
Solution	More cameras are connected than have been configured. The default setting of the network adapter allows up to 2 cameras to be operated simultaneously. If you wish to operate more cameras, increase the Init parameters `ipMaxReceiver` and `UdpMaxReceiver` of the network adapter to twice the number of connected cameras (see Configuration of the network adapter [▶ 59]).

9.3.3.5 Warning: Reading/Writing is not permitted

Warning	 This warning may appear when connecting a camera.
Solution	The warning indicates that a write or read command that TwinCAT Vision sends to the camera is not allowed. This can happen if the definition of the access rights on a register in the GenAPI description deviates from the actual behavior of the camera. Usually this camera register is unimportant for the user. To ascertain which register this concerns, you can search for the displayed register address in the Configuration Assistant [▶ 75].

9.3.4 TwinCAT HMI packages

Solutions for situations in which TwinCAT HMI packages do not work as expected are explained below.

9.3.4.1 Server extension

Images are not displayed

Situation	Images from the PLC are not displayed in the HMI, although the server extension [▶ 2704] is installed and the image symbols are appropriately linked.
Solution 1	**Missing administrator rights** You have chosen "Stream" as the image format, but you haven't started the HMI server as an administrator. Either select a different image format (BMP, PNG or JPG), or start the HMI server as an administrator.

Images missing from a published page

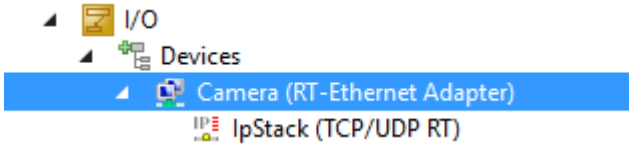
Situation	The TcHmiVision [▶ 2704] server extension has been successfully integrated into the project. Images are displayed in the Live View or on the HMI provided by the Engineering server. The project could also be published error-free on the standalone TcHmi server. However, no images are displayed on the published HMI.
Solution 1	**Missing target licenses** A target license is included in the TC3 HMI server license. Additional licenses are required if the HMI server is to communicate with several targets (ADS connections).
Solution 2	**Wrong configuration of the runtime environment** The default configuration used in Engineering may differ from the published configuration (different AmsNetIds). Therefore, check the configuration of the runtime environment and adjust it if necessary.
Solution 3	**Restricted user rights** The TwinCAT HMI user management is used and the currently logged-in user has no permissions to view the images or receive the extension data. Check the authorizations of your user groups.
Solution 4	**Limit the data quantity or the refresh interval** As the data quantity can be quite large, especially in the case of several images that have to be retrieved simultaneously from the controller by ADS and converted by the HMI server, this can lead to bottlenecks. Several options are available to remedy this: reducing the size of the images in the controller, increasing the refresh rates of the individual images, using different refresh rates or using a compressed image format.

9.3.5 Camera communication

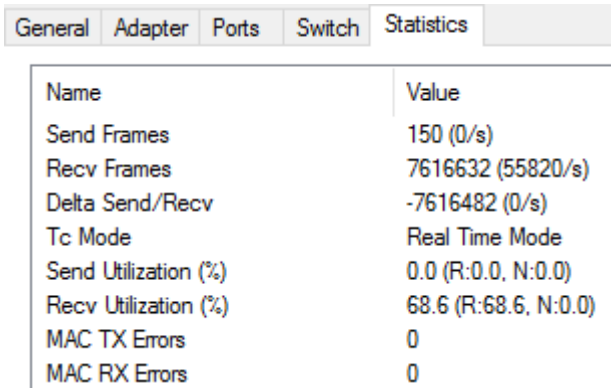
In case of problems with the connection, especially with sent or received frames, as well as problems with the parameterization or missing images, you will find the following diagnostic information in the network adapter and under the Image Acquisition Module of the camera object.

Diagnostic options in the RT Ethernet adapter

1. First select the RT Ethernet adapter used for the respective camera communication.



2. Open it by double-clicking and go to the **Statistics** tab.



⇨ Here you can see whether frames are generally sent and received or whether errors occur. Furthermore you can read the utilization of the connection in percent at **Recv Utilization**. A very high utilization of more than 98% already indicates an overload, especially if several cameras are operated on one adapter.

Further diagnostic information can be found in the IpStack.

1. To do this, first open the underlying IpStack by double-clicking on it.
2. Go to the **Parameter (Online)** tab.

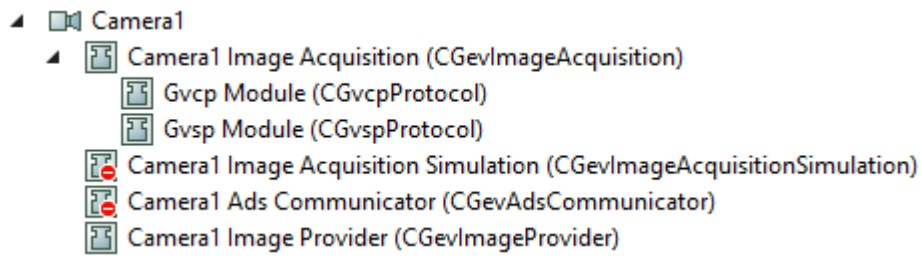
⇨ Here you can see the counter values for the individual categories.

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	

Diagnostic options in the Image Acquisition modules

1. First select the respective camera object.



2. Open the **Parameters (online)** tab under the Gvcp module.

⇨ Here you get the information about the Control Protocol communication. This includes e.g. the heartbeat, reading and writing of registers and commands like **Start Acquisition**.
If everything is OK, only the two nSuccess values should count up. If timeouts occur, refer to this chapter Timeouts [▶ 2873] for remedies. For other problems a deeper diagnosis with Wireshark is necessary.

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

The diagnostic information for the Streaming Protocol can be found under the Gvsp module on the **Parameter (online)** tab. These counters refer purely to the transmitted image data and are divided into the individual telegrams/packages and complete images/blocks.

If everything is Ok, only the nSuccess and the nCompletedBlocks value should count up.

The nResendRequests already indicate problems, but in this case they could still be answered, because all other counters are still zero. In case of a timeout, the BlockTimeout should be increased in the Configuration Assistant under the GVSP Module settings [▶ 101]. Counting up the nBlockIdOverflows is normal and does not represent an error. The counter will eventually overflow due to the value range or will be reset when the camera is restarted.

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

Grouping of error codes:
Global error codes: ADS Return Codes [▶ 2879]... (0x9811_0000 ...)
Router error codes: ADS Return Codes [▶ 2880]... (0x9811_0500 ...)
General ADS errors: ADS Return Codes [▶ 2880]... (0x9811_0700 ...)
RTime error codes: ADS Return Codes [▶ 2882]... (0x9811_1000 ...)

Global error codes

Hex	Dec	HRESULT	Name	Description
0x0	0	0x98110000	ERR_NOERROR	No error.
0x1	1	0x98110001	ERR_INTERNAL	Internal error.
0x2	2	0x98110002	ERR_NORTIME	No real time.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Allocation locked – memory error.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Wrong HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Target port not found – ADS server is not started, not reachable or not installed.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Target computer not found – AMS route was not found.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unknown command ID.
0x9	9	0x98110009	ERR_BADTASKID	Invalid task ID.
0xA	10	0x9811000A	ERR_NOIO	No IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unknown AMS command.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 error.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port not connected.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Invalid AMS length.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Invalid AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installation level is too low –TwinCAT 2 license error.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	No debugging available.
0x12	18	0x98110012	ERR_PORTDISABLED	Port disabled – TwinCAT system service not started.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port already connected.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 error.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync error.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	No index map for AMS Sync available.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Invalid AMS port.
0x19	25	0x98110019	ERR_NOMEMORY	No memory.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP send error.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host unreachable.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Invalid AMS fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS send error – secure ADS connection failed.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Access denied – secure ADS access denied.

Router error codes

Hex	Dec	HRESULT	Name	Description
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Locked memory cannot be allocated.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	The router memory size could not be changed.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	The Debug mailbox has reached the maximum number of possible messages.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	The router is not initialized.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	The port number is already assigned.
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	The port is not registered.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	The maximum number of ports has been reached.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	The port is invalid.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	The router is not active.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	The mailbox has reached the maximum number for fragmented messages.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	A fragment timeout has occurred.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	The port is removed.

General ADS error codes

Hex	Dec	HRESULT	Name	Description
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	General device error.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by the server.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Invalid index group.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Invalid index offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Reading or writing not permitted. Several causes are possible. For example, an incorrect password was entered when creating routes.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parameter size not correct.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Invalid data values.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Device is not ready to operate.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Device is busy.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Invalid operating system context. This can result from use of ADS blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Insufficient memory.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Invalid parameter values.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Not found (files, ...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax error in file or command.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objects do not match.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Object already exists.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol not found.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Invalid symbol version. This can occur due to an online change. Create a new handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Device (server) is in invalid state.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	No further handle available.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Notification size too large.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Device not initialized.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Device has a timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface query failed.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Wrong interface requested.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID is invalid.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID is invalid.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Request pending.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Request is aborted.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal warning.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Invalid array index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol not active.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Access denied. Several causes are possible. For example, a unidirectional ADS route is used in the opposite direction.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Missing license.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	License expired.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	License exceeded.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Invalid license.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	License problem: System ID is invalid.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	License not limited in time.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Licensing problem: time in the future.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	License period too long.
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception at system startup.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Invalid signature.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Invalid certificate.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public key not known from OEM.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	License not valid for this system ID.

Hex	Dec	HRESULT	Name	Description
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo license prohibited.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Invalid function ID.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Outside the valid range.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Invalid alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Invalid platform level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Context – forward to passive level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Context – forward to dispatch level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Context – forward to real-time.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Client error.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Service contains an invalid parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling list is empty.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var connection already in use.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	The called ID is already in use.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Error in Win32 subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port not open.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	No AMS address.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Internal error in Ads sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Hash table overflow.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Key not found in the table.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	No symbols in the cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Invalid response received.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port is locked.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	The request was canceled.

RTime error codes

Hex	Dec	HRESULT	Name	Description
0x1000	4096	0x98111000	RTERR_INTERNAL	Internal error in the real-time system.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer value is not valid.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task pointer has the invalid value 0 (zero).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack pointer has the invalid value 0 (zero).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	The request task priority is already assigned.
0x1005	4101	0x98111005	RTERR_NOMORETCB	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	No free semaphores available. The maximum number of semaphores is 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	No free space available in the queue. The maximum number of positions in the queue is 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	No external sync interrupt applied.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Application of the external synchronization interrupt has failed.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in the BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Activation of Intel VT-x fails.

Specific positive HRESULT Return Codes:

HRESULT	Name	Description
0x0000_0000	S_OK	No error.
0x0000_0001	S_FALSE	No error. Example: successful processing, but with a negative or incomplete result.
0x0000_0203	S_PENDING	No error. Example: successful processing, but no result is available yet.
0x0000_0256	S_WATCHDOG_TIMEOUT	No error. Example: successful processing, but a timeout occurred.

TCP Winsock error codes

Hex	Dec	Name	Description
0x274C	10060	WSAETIMEDOUT	A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.
0x274D	10061	WSAECONNREFUSED	Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.
0x2751	10065	WSAEHOSTUNREACH	No route to host - a socket operation referred to an unavailable host.
More Winsock error codes: Win32 error codes			

9.5 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service

- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

9.6 Third party Components

This software contains third-party components. For further information, please refer to the license file, which is provided in the following folder:

Different memory paths in TwinCAT 3.1 4024 and TwinCAT 3.1 4026

Up to and including Build 4024:

<TwinCatInstallPath>\3.1\System\Legal\TF7xxx Additional license information.txt

From 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*

More Information:
www.beckhoff.com/twincat-vision/

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com