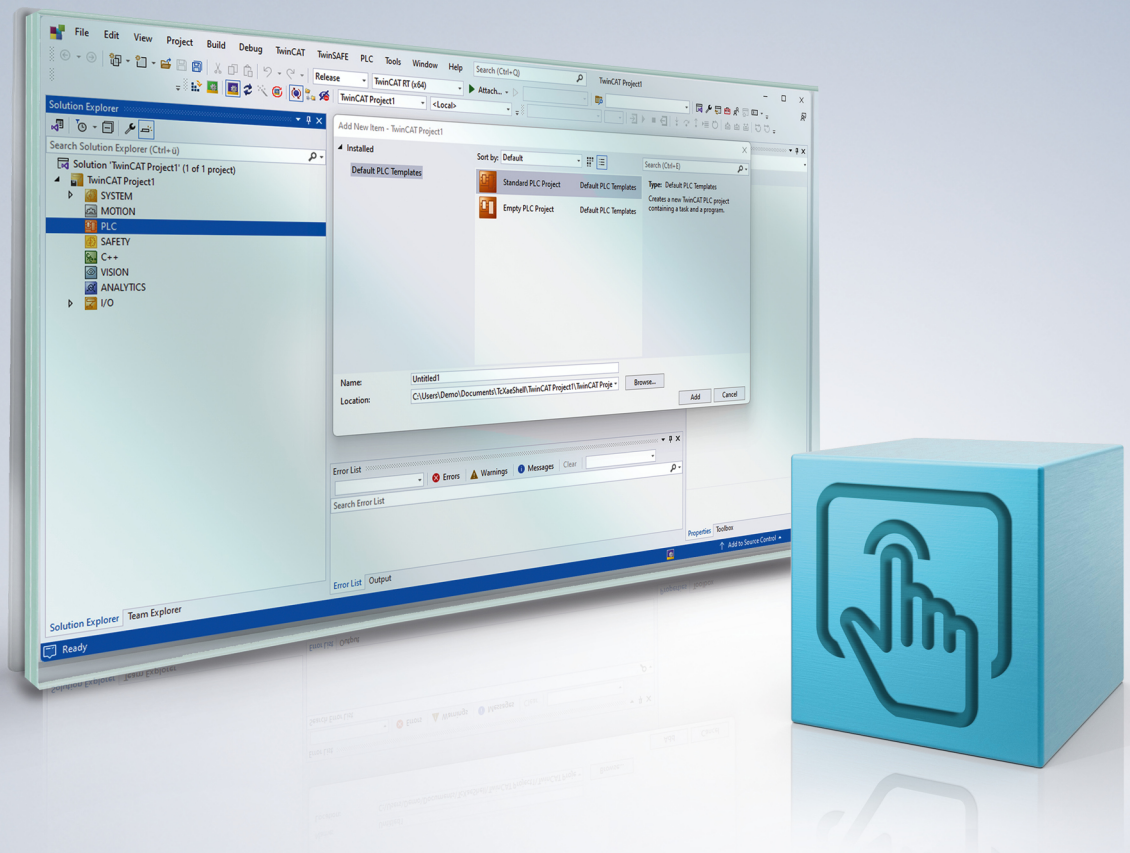
BECKHOFF New Automation Technology

Handbuch | DE

TwinCAT 3

HMI Prozessbibliothek



02.02.2026 | Version: 1.0.0

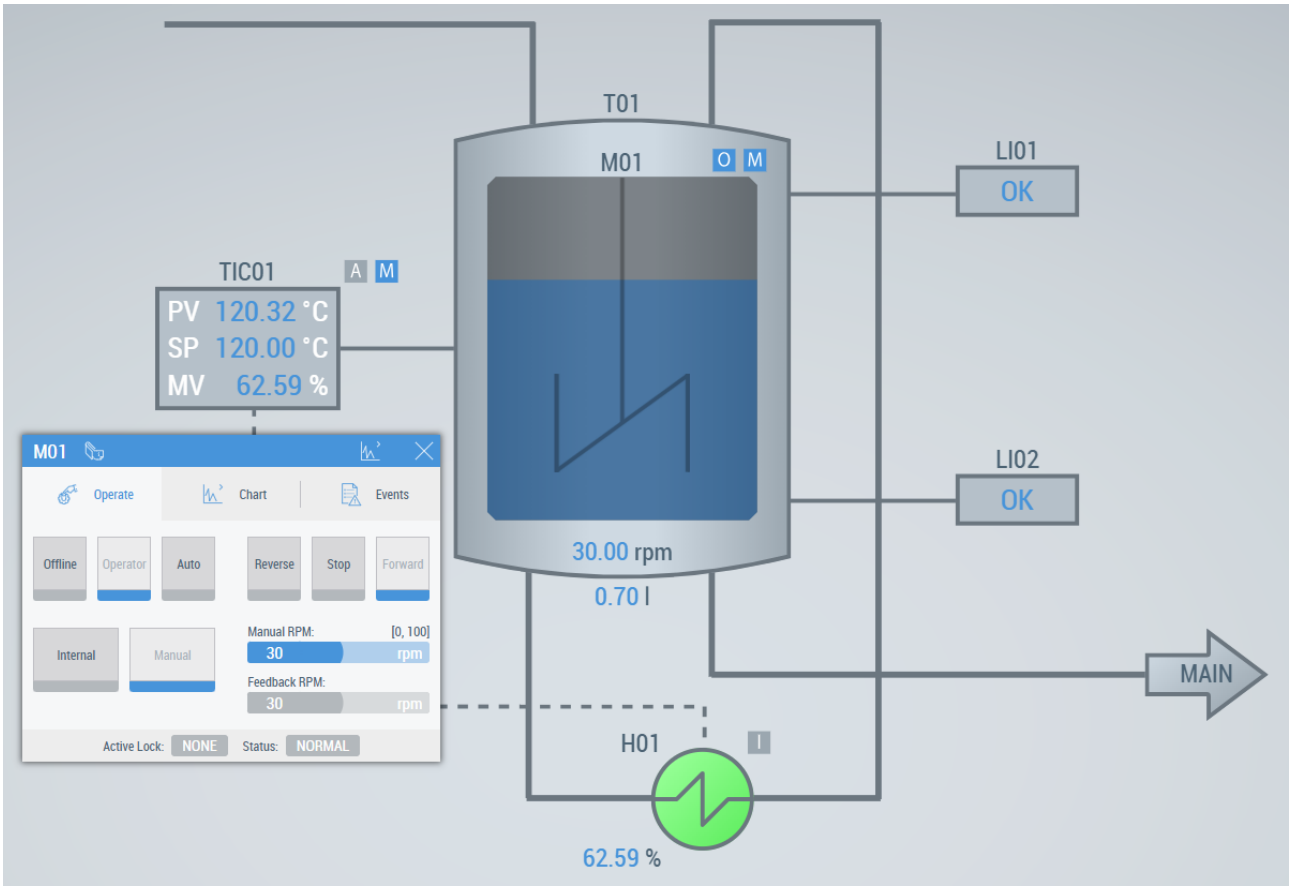
Inhaltsverzeichnis

1 Übersicht

Die TwinCAT HMI Process Library beinhaltet zusätzliche Controls für das TwinCAT HMI (TE/TF2000), die speziell für die Prozessindustrie konstruiert wurden, aber auch für Anwendungen in anderen Bereichen genutzt werden können. Diese Controls verfügen über klare grafische Symbole, die die P&ID-Elemente mit allen wichtigen Informationen veranschaulichen. Die Controls werden mit Faceplates geliefert, die vorkonfiguriert sind und entweder vom Benutzer angepasst oder selbst erstellt werden können.

Die Controls können auf verschiedene Weise mit den SPS-Variablen/Objekten verknüpft werden: jede Eigenschaft kann mit einer separaten SPS-Variable verknüpft werden, oder das gesamte SPS-Objekt (z.B. ein FB) kann mit dem gesamten Control verknüpft werden.

Die TwinCAT HMI Process Library fasst viele vorhandene Funktionalitäten in fertigen Controls zusammen. So können beispielsweise TwinCAT-Scope-Charts ebenso angezeigt werden wie Ereignislisten oder Historisierungsdiagramme.



1.1 Motivation

Controls

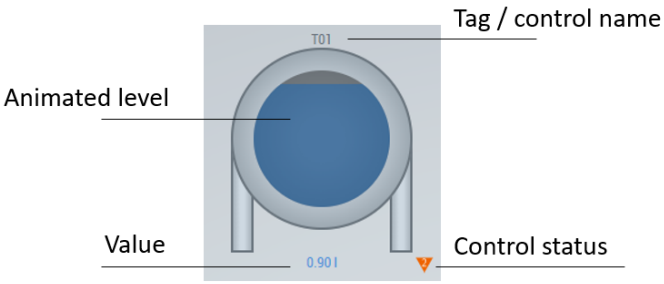
In der HMI Process Library stehen vorgefertigte HMI-Controls zur Verfügung. Sie ermöglichen es den Benutzern, bei der Entwicklung eigener komplexer Controls viel Zeit zu sparen. Die Bibliothek umfasst häufig verwendete Controls auf der Grundlage der Norm DIN EN ISO 10628-2.

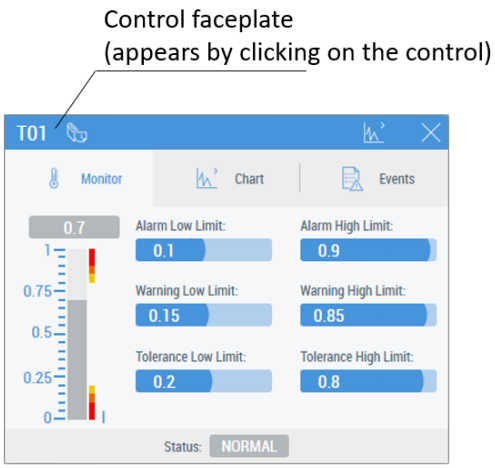
Anfragen zum Hinzufügen neuer Controls

Wenn Sie zusätzliche Controls benötigen, senden Sie bitte eine Anfrage an processindustry@beckhoff.com.

1:1 Mapping

Durch das Mapping eines eigenen SPS-Objekts, z. B. eines FBs, auf das Control kann jede Information automatisch dem entsprechenden Element zugeordnet werden. Zusätzlich zur grafischen Darstellung verfügt das Control über eine Faceplate-Funktionalität in Form eines Pop-ups.

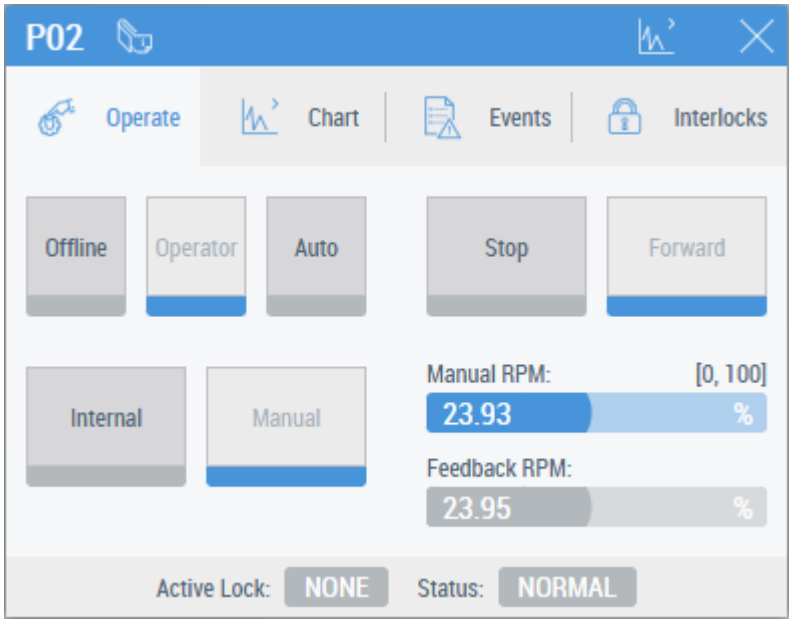


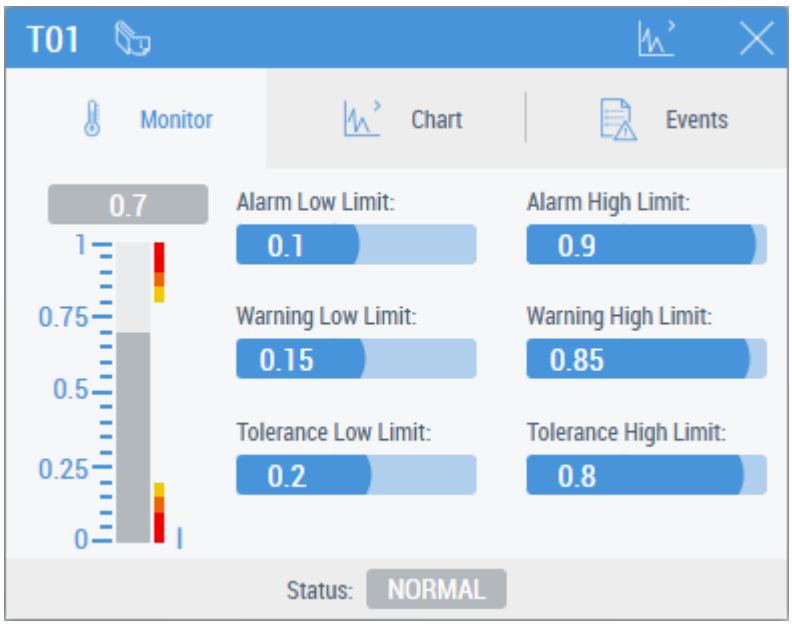


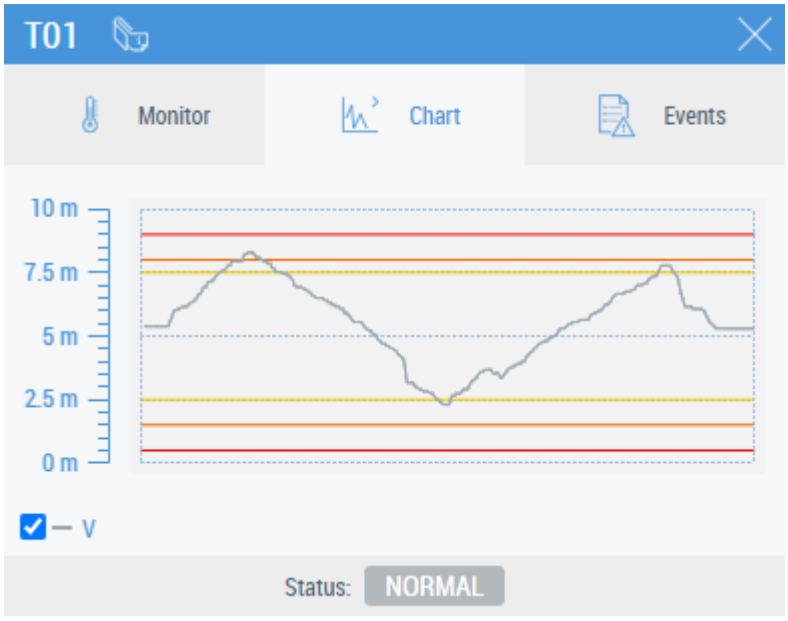
Fertige Faceplates

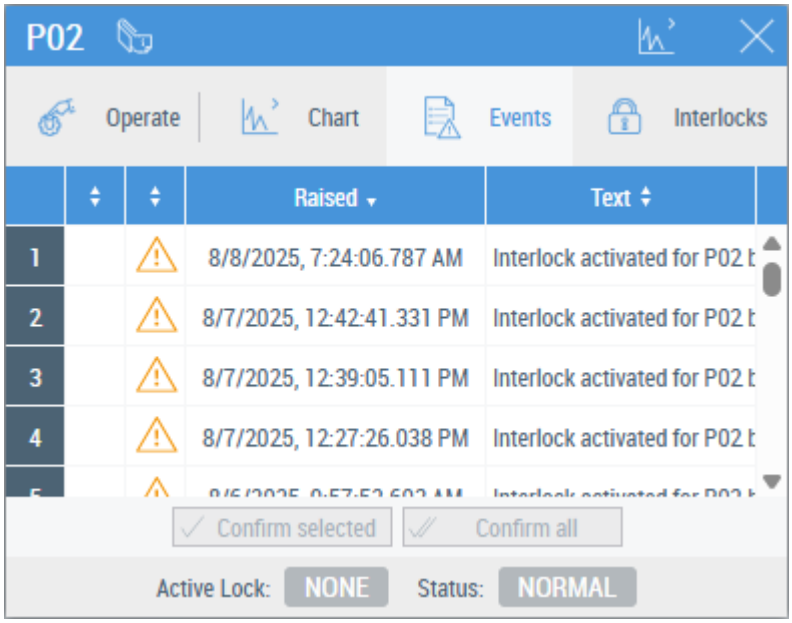
Fertige Faceplates bieten Optionen zum:

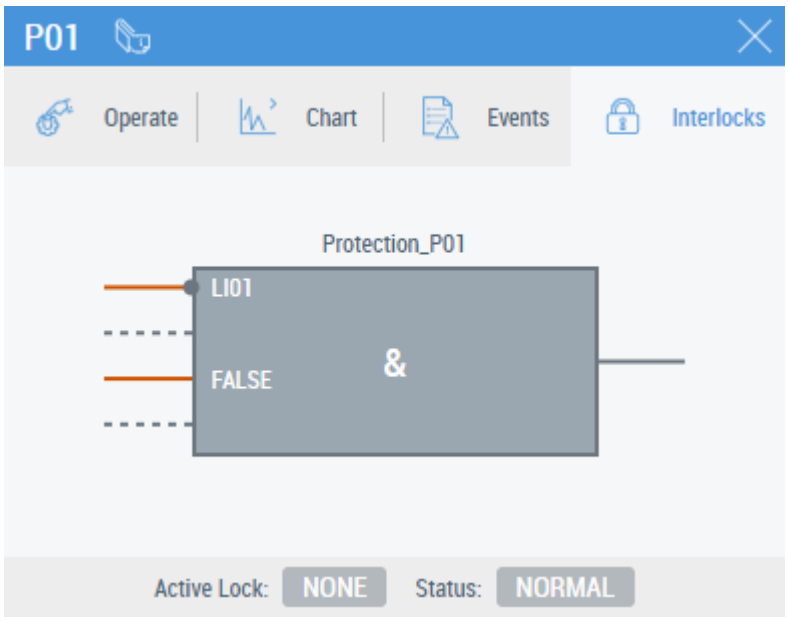
- Anzeigen von aktuellen Daten, Alarmen und Fehlern.
- Senden von Befehlen und Ändern von Sollwerten.
- Konfigurieren von Einstellungen, Fehlerverhalten, Alarmbedingungen.
- Anzeigen von Diagrammen für Daten.
- Anzeigen eines Ereignisprotokolls für das Control.
- Anzeigen des LockView-Controls zur Klärung des Grundes für die aktive Verriegelung.
- Anzeigen des aktuellen Status und der aktiven Verriegelung in der Fußzeile.




T01
Monitor
Chart
Events
0.7
1
0.75
0.5
0.25
0
Alarm Low Limit:
0.1
Alarm High Limit:
0.9
Warning Low Limit:
0.15
Warning High Limit:
0.85
Tolerance Low Limit:
0.2
Tolerance High Limit:
0.8
Status:
NORMAL


T01
Monitor
Chart
Events
10 m
7.5 m
5 m
2.5 m
0 m
V
Status:
NORMAL


P02
Operate
Chart
Events
Interlocks
Raised
Text
1
8/8/2025, 7:24:06.787 AM
Interlock activated for P02 b
2
8/7/2025, 12:42:41.331 PM
Interlock activated for P02 b
3
8/7/2025, 12:39:05.111 PM
Interlock activated for P02 b
4
8/7/2025, 12:27:26.038 PM
Interlock activated for P02 b
Confirm selected
Confirm all
Active Lock:
NONE
Status:
NORMAL



Anpassbare Funktionalität

Neben den vorkonfigurierten Eigenschaften von Controls und Faceplates gibt es eine Vielzahl von Möglichkeiten, die Datenbindung an Controls und den Inhalt von Faceplates individuell anzupassen.

1.2 Szenarien

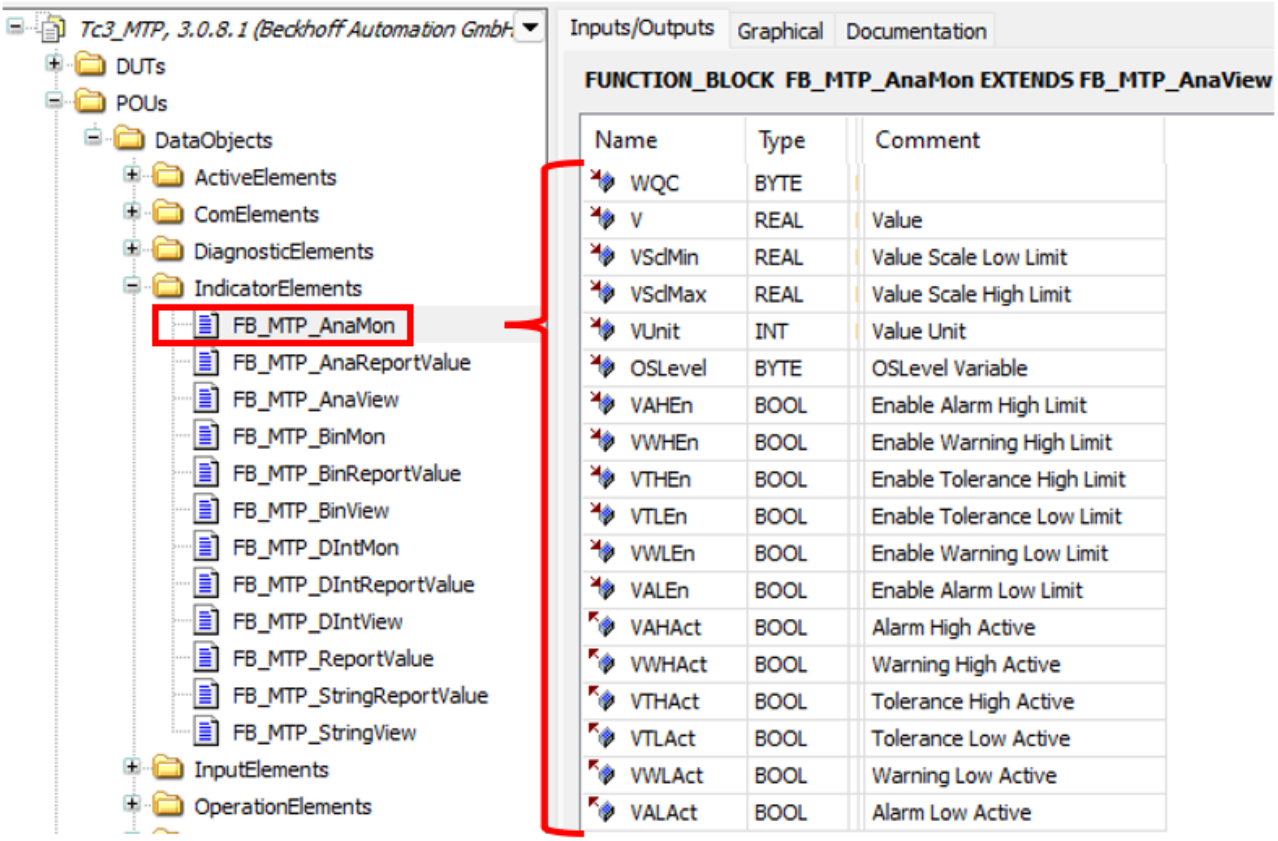
1.2.1 Möglichkeiten des Mappings von SPS-Variablen auf HMI-Controls

Für das Mapping der Controls auf die SPS stehen drei Arten zur Verfügung: MTP-Library-Mapping, Mapping eines benutzerdefinierten Objekts (z. B. eines FB) sowie direktes Variablen-Mapping.

- Das MTP-Library-Mapping ist die komfortabelste und schnellste Methode, da eine standardisierte, sofort einsatzbereite SPS-FB-Instanz (siehe MTP Runtime FBs) über eine einzige **Data Symbol**-Eigenschaft an das Control gebunden wird.
- Mit dem benutzerdefinierten FB-Mapping kann auch ein Einzelmapping über die **Data Symbol**-Eigenschaft implementiert werden. Es unterscheidet sich vom MTP-Library-Mapping dadurch, dass der Benutzer das Control an eine Instanz des von ihm entwickelten FBs bindet, die er mit Attributpragmas angereichert hat. Der Vorteil gegenüber dem direkten Variablen-Mapping ist, dass diese Anreicherung nur einmal in der SPS und nicht für jede Instanz im HMI vorgenommen werden muss.
- Direktes Variablen-Mapping bedeutet, dass eine Eigenschaft des Controls der entsprechenden SPS-Variable zugeordnet werden kann. Diese Methode kann alleine oder in Kombination mit den beiden anderen angewendet werden. Bei kombinierter Verwendung können einzelne Eigenschaften des Controls durch Mapping zu einer separaten SPS-Variable eingestellt oder überschrieben werden.

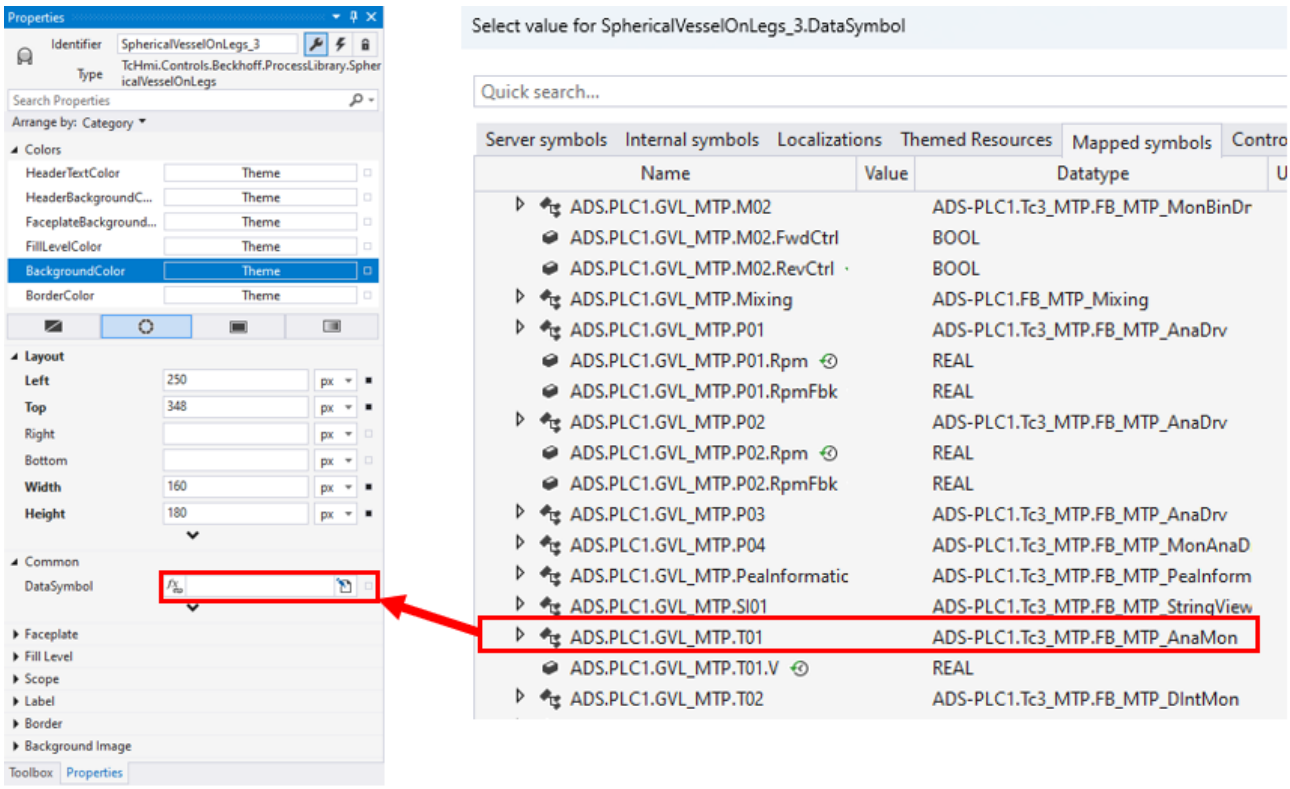
MTP-Library-Mapping

Die Bibliothek TF8400 | TwinCAT 3 MTP Runtime enthält standardisierte Schnittstellen. Im Folgenden ist eine Sequenz dargestellt, die zeigt, wie ein bestimmter standardisierter Typ an das entsprechende Control gebunden ist.



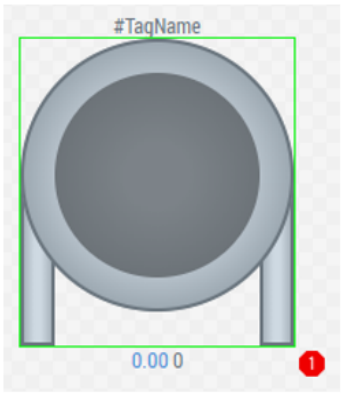
MTP-Library-FBs werden in der SPS instanziiert und stehen für das Mapping im HMI-Projekt zur Verfügung.

```
558  T01: FB_MTP_AnaMon := (
559                  TagName := 'T01',
560                  TagDescription := 'Tank Level',
561                  WQC := 255,
562                  VSclMin := 0,
563                  VSclMax := 1,
564                  VUnit := E_MTP_Unit.Liter,
565                  V := 0,
566                  VAHEn := TRUE,
567                  VAHLim := 1,
568                  VALEn := TRUE,
569                  VALLim := 0,
570                  VWHEn := TRUE,
571                  VWHLim := 0.9,
572                  VWLEn := TRUE,
573                  VWLLim := 0.1,
574                  VTHEn := TRUE,
575                  VTHLim := 0.8,
576                  VTLEn := TRUE,
577                  VTLLim := 0.2
578                  );
```



Das Ergebnis erscheint unmittelbar nach dem Mapping:

a)



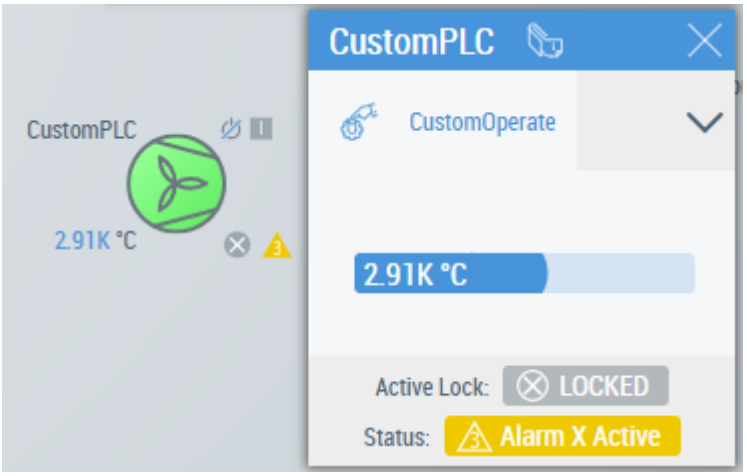
b)

Benutzerdefiniertes FB-Mapping

Die Verwendung des speziellen `'TcHmi.ProcessLibrary'` Pragma-Attributs macht einen benutzerdefinierten FB für das Mapping zum Control über die **Data Symbol**-Eigenschaft verfügbar. Die Pragma-Attribute müssen verwendet werden, um den FB zu erstellen. Dann werden alle Instanzen an das zugeordnete Control weitergeleitet:

- Konfigurationsdaten (definieren einige Eigenschaften und das Verhalten des Controls), die in den Attributen angegeben sind
- aktuelle Werte von Variablen, die mit den Attributen gekennzeichnet sind

```
{region 'TcHmiProcessLibrary.FaceplateParameters'}
    {attribute 'TcHmi.ProcessLibrary.Modal' := 'True'}
    {attribute 'TcHmi.ProcessLibrary.Movable' := 'False'}
    {attribute 'TcHmi.ProcessLibrary.RestoreBounds' := 'False'}
    {attribute 'TcHmi.ProcessLibrary.HideWithControl' := 'True'}
    {attribute 'TcHmi.ProcessLibrary.ReshowWithControl' := 'False'}
{endregion}
{region 'TcHmiProcessLibrary.LabelParameters'} [45 lines]
FUNCTION_BLOCK FB_AttrTest2
VAR_INPUT
    {attribute 'TcHmi.ProcessLibrary' := 'LabelText'}
    sName: STRING := 'CustomPLC2';

    {attribute 'TcHmi.ProcessLibrary' := 'LabelValue'}
    {attribute 'TcHmi.ProcessLibrary.Format' := 'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4'}
    fValue: REAL;

    {attribute 'TcHmi.ProcessLibrary' := 'LabelValueUnit'}
    {attribute 'TcHmi.ProcessLibrary.Format' := 'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter'}
    nUnit: INT := 1001;

    {attribute 'TcHmi.ProcessLibrary' := 'LabelStateMode'}
    nStateMode: INT := 1;

    {attribute 'TcHmi.ProcessLibrary' := 'LabelSourceMode'}
    nSrcMode: INT := 1;

    {attribute 'TcHmi.ProcessLibrary' := 'LabelInterlockStatus'}
    nInterlockStatus: INT := 1;

    {attribute 'TcHmi.ProcessLibrary' := 'LabelAlarmStatus'}
    nAlarmStatus: INT := 1;

    {attribute 'TcHmi.ProcessLibrary' := 'GraphicStatus'}
    nGraphicStatus: INT := 0;

    {attribute 'TcHmi.ProcessLibrary' := 'HeaderText'}
    sHeaderText: STRING := 'New Header';
END_VAR
```

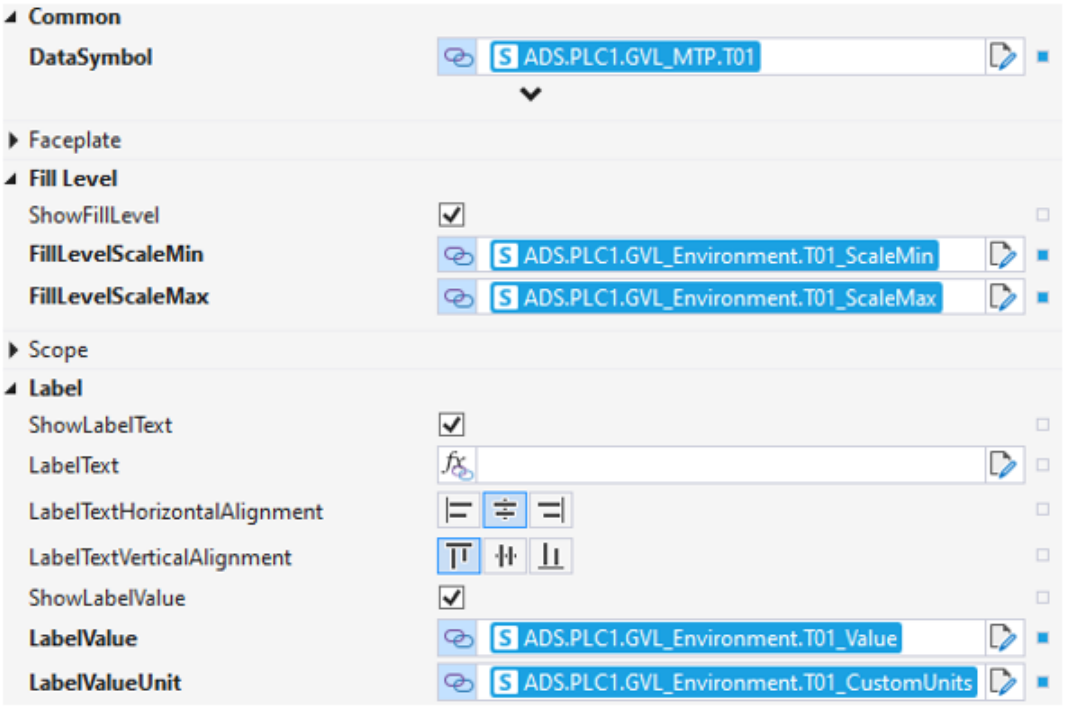


Ausführlichere Informationen zur Verwendung von SPS-Attributen finden Sie in den Kapiteln SPS-Attribut-Funktionalität [▶ 75] und Verwendung der SPS-Attribut-Funktionalität [▶ 21].

Direktes Mapping und Überschreiben durch SPS-Variablen

Dies ist die Standardmethode des Variablen-Mappings in TwinCAT HMI: Jede Eigenschaft des Controls der HMI Process Library, die über das **Eigenschaftsfenster** verfügbar ist, kann auf eine separate Variable eines entsprechenden Typs abgebildet werden.

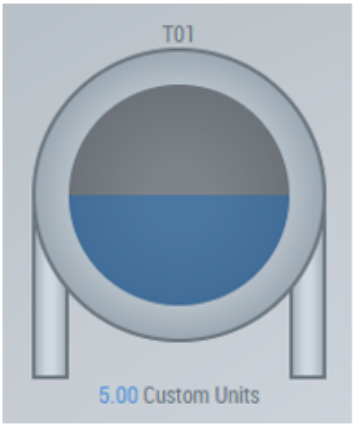
a)



b)

T01_ScaleMax	REAL	10
T01_ScaleMin	REAL	0
T01_Value	REAL	5
T01_CustomUnits	STRING	'Custom Units'

c)



Separates Mapping für jede Control-Instanz

Das Mapping von Variablen zu den Eigenschaften des Controls muss für jede HMI-Control-Instanz separat vorgenommen werden.

Control-Eigenschaften, die bereits über die **Data Symbol**-Eigenschaft auf das MTP oder den benutzerdefinierten FB abgebildet wurden, können durch das direkte Mapping überschrieben werden. Dadurch wird die Verwendung der Controls flexibler.

Priorisierung der gemappten Daten

Über spezifische Eigenschaften eingestellte Daten haben Vorrang vor der **Data Symbol**-Eigenschaft.

1.2.2 Faceplate-Typen

Faceplate-Typen können unterschieden werden, indem sie vom Control verwendet werden.

Es gibt drei Optionen, wie definiert werden kann, welche Faceplate-Typen vom Control verwendet werden: vordefinierte Faceplates, **Tabs**-Eigenschaft und **Faceplate Control**-Eigenschaft. Der Hauptunterschied zwischen den **Tabs**- und **Faceplate Control**-Eigenschaften besteht darin, dass ein **TabNavigation**-Control vordefiniert ist und mit mehreren UserControls in Registerkarten bestückt werden kann, während die **Faceplate Control**-Eigenschaft ein einzelnes dediziertes UserControl erwartet.

Die folgenden Typen sind nach ihrer Priorität geordnet (die niedrigste Priorität zuerst):

- Vordefinierte Faceplates machen die Erstellung eines HMI einfach. Sie verfügen bereits über alle wesentlichen Elemente für die HMI Process Library Controls. Dies funktioniert nur mit TwinCAT MTP Runtime oder benutzerdefinierten FBs, die die exakte Variablenschnittstelle implementieren.
- Mit der **Tabs**-Eigenschaft können benutzerdefinierte Registerkarten für eine vorimplementierte TabNavigation erstellt werden.
- Die **Faceplate Control**-Eigenschaft ermöglicht es, den gesamten Faceplate-Inhalt mit einem benutzerdefinierten UserControl zu überschreiben. Der Hauptunterschied zur Verwendung der **Tabs**-Eigenschaft besteht darin, dass es in diesem Fall keine Registerkarten auf dem Faceplate gibt.

Verwendung von SPS-Attributen zum Einstellen der Tabs- und Faceplate Control-Eigenschaften

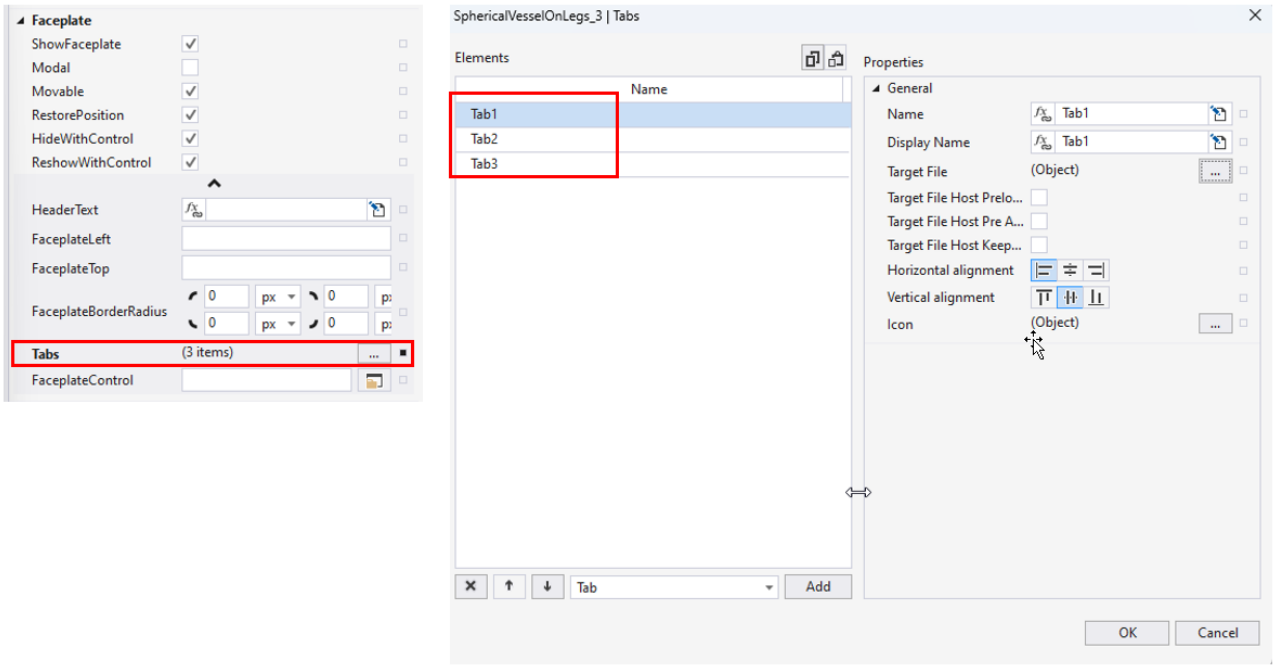
Sie können spezielle SPS-Pragma-Attribute verwenden, um die **Tabs**- und **Faceplate Control**-Eigenschaften einzustellen.

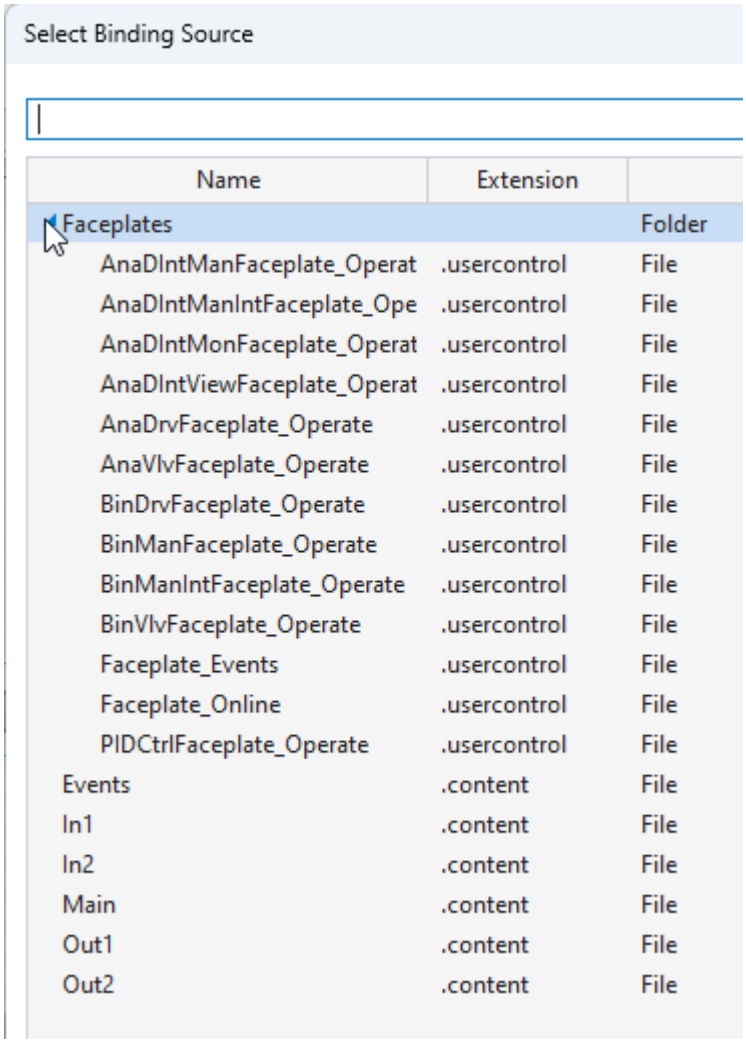
Vordefinierte Faceplates

Vordefinierte Faceplates sind für alle Controls verfügbar und werden automatisch verwendet, wenn die **Data Symbol**-Eigenschaft mit der FB-Instanz der MTP Library verknüpft ist. Mit der **Interlocks Tab**-Eigenschaft des Controls kann eine **Interlocks**-Registerkarte zu den vordefinierten Registerkarten hinzugefügt werden, ohne dass die vordefinierten Registerkarten neu erstellt werden müssen.

Tabs-Eigenschaft

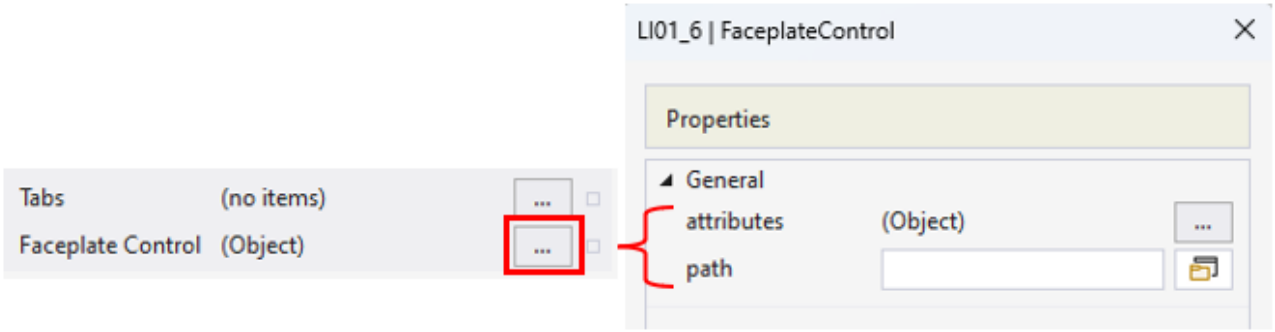
Mit der **Tabs**-Eigenschaft können verfügbare oder vom Benutzer erstellte Faceplates und Inhaltselemente als Registerkarten angezeigt werden.





Faceplate Control-Eigenschaft

Die **Faceplate Control**-Eigenschaft ermöglicht es, den Pfad zum gewünschten UserControl und dessen Parameter festzulegen.



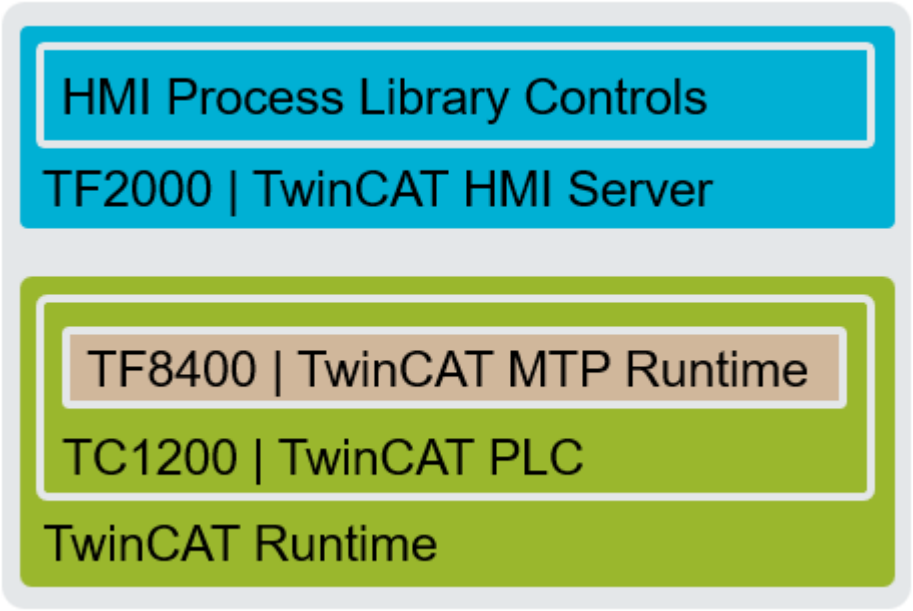
1.2.3 Systemarchitekturen

Da die HMI Process Library auf TwinCAT HMI basiert, sind alle Szenarien für TwinCAT HMI auch für die HMI Process Library zugänglich.

Werden die verschiedenen Möglichkeiten des Mappings von SPS-Variablen und die Flexibilität durch die Verwendung von Faceplates berücksichtigt, so ergeben sich aus Sicht der Systemarchitektur zwei Hauptszenarien für den Einsatz der HMI Process Library: entweder mit oder ohne Verwendung der TwinCAT MTP Runtime.

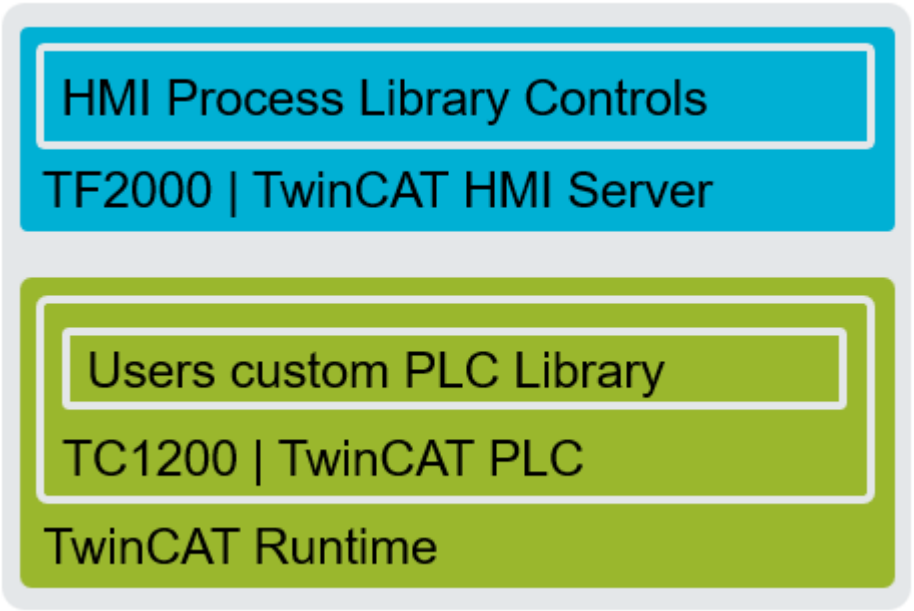
Mit TwinCAT MTP Runtime

Die HMI Process Library bietet Controls, die mit den Funktionsbausteinen aus der TwinCAT MTP Runtime kompatibel sind und der MTP-Spezifikation entsprechen. Dadurch kann das Control auf nur ein SPS-Symbol abgebildet werden.



Ohne TwinCAT PLC MTP Library

Die Verwendung der HMI Process Library ohne die PLC MTP Library gibt dem Anwender die Möglichkeit, fertige Controls mit eigenen FBs zu verwenden.



1.3 Benutzeranforderungen

Für Nutzer dieser Bibliothek werden folgende Grundkenntnisse vorausgesetzt:

- TwinCAT System (siehe Dokumentation Grundlagen)
- TwinCAT HMI
- TwinCAT Scope (abhängig davon, ob ein Scope-Projekt eingebunden werden soll)

- TwinCAT EventLogger
- TwinCAT MTP (je nachdem, ob es mit den MTP-Komponenten des Bibliotheksstandards verwendet wird)
- TwinCAT OPC UA (abhängig davon, ob die Kommunikation mit POL/DCS genutzt werden soll)

1.4 Systemanforderungen

Die HMI Process Library benötigt die TwinCAT HMI Version 1.14.

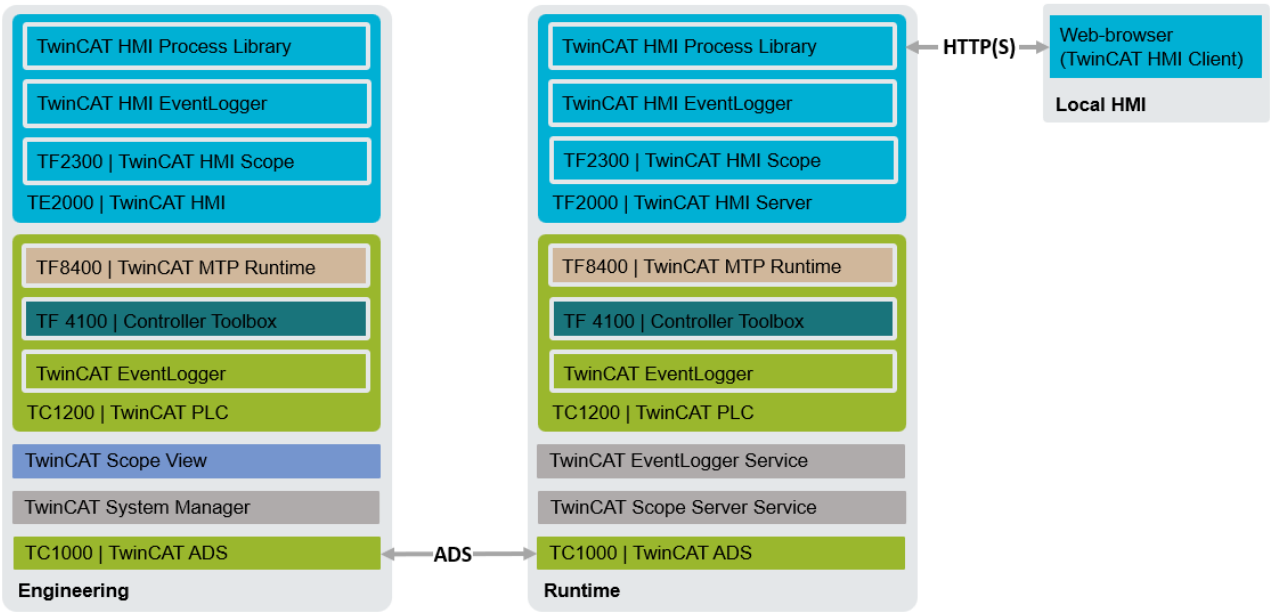
Alle anderen Anforderungen sind die gleichen wie für TwinCAT HMI. Beachten Sie die Dokumentation zum TE2000 TwinCAT 3 HMI Engineering.

Microsoft.TypeScript.MSBuild NuGet Paket

Beziehen Sie Updates für dieses Paket aus dem Beckhoff Offline Repository (nicht von nuget.org).

1.5 Architektur

In der folgenden Abbildung ist die TwinCAT HMI Process Library im Verhältnis zu den anderen TwinCAT-Komponenten dargestellt. Diese Informationen sind umfassender als im Kapitel Systemarchitekturen [▶ 14]. Es sind nicht alle TwinCAT-Komponenten enthalten, aber es vermittelt einen Eindruck von der Funktionsweise und den Interaktionen der TwinCAT HMI Process Library.



Die TwinCAT HMI Process Library ist eine Erweiterung von TwinCAT HMI. TwinCAT HMI Engineering und TwinCAT HMI Server müssen jeweils verwendet werden, um Projekte zu entwickeln und auszuführen, die die Bibliothek verwenden.

Die TwinCAT HMI Process Library erhält Ereignisse vom TwinCAT EventLogger. Sie benötigt deshalb diese Komponenten:

- den TwinCAT HMI EventLogger zur Anzeige von Ereignissen im HMI und zum Abrufen von Ereignissen aus dem EventLogger
- die TwinCAT EventLogger-Bibliothek (optional) auf SPS-Ebene zum Generieren und Abrufen von Ereignissen aus dem EventLogger
- den TwinCAT EventLogger Service auf einer Runtime zur Behandlung von Ereignissen auf OS-Ebene

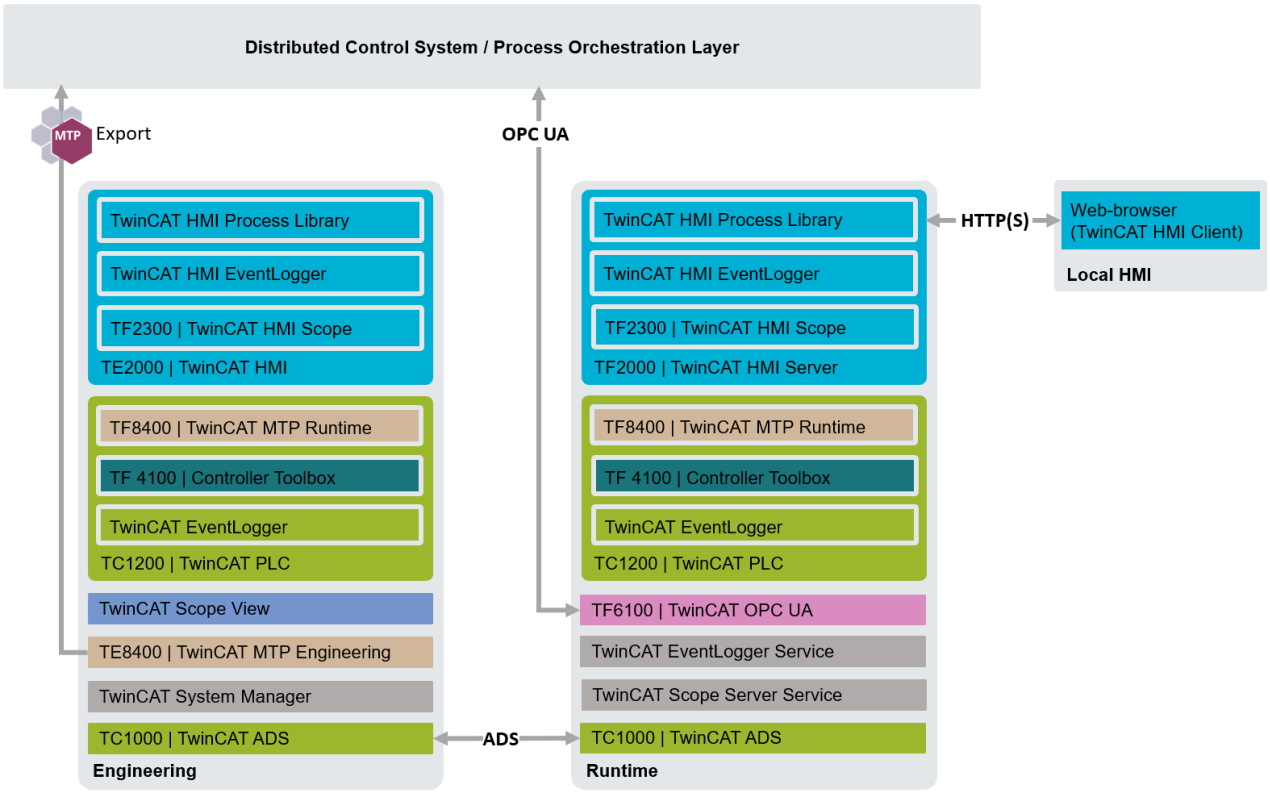
Die TwinCAT HMI Process Library kann mit der TwinCAT-Scope-Funktionalität Diagramme zeichnen. Dazu benötigt sie die folgenden Optionen:

- TwinCAT HMI Scope

- TwinCAT Scope View auf der Engineering-Seite zum Erstellen und Konfigurieren von Diagrammen
- TwinCAT Scope Server Service zum Handling der Scope-Daten auf OS-Ebene

TwinCAT MTP Runtime ist optional. Sie bietet standardisierte FBs für technologische Anlagen.

Mit der TwinCAT HMI Process Library lassen sich lokale HMIs für Systeme erstellen, die von POL/DCS gesteuert werden. Die folgende Abbildung zeigt die Architektur eines solchen Systems:



Weitere Informationen finden Sie in den Dokumentationen TE8400 | TwinCAT 3 MTP Engineering und TF8400 | TwinCAT 3 MTP Runtime.

1.6 Geschäftsmodell

Die TwinCAT HMI Process Library benötigt keine spezielle Lizenz, aber sie benötigt mindestens eine TF2000-Lizenz. Andere Lizenzen, die in der folgenden Tabelle aufgeführt sind, bieten zusätzliche Funktionalität für Lösungen mit der TwinCAT HMI Process Library.

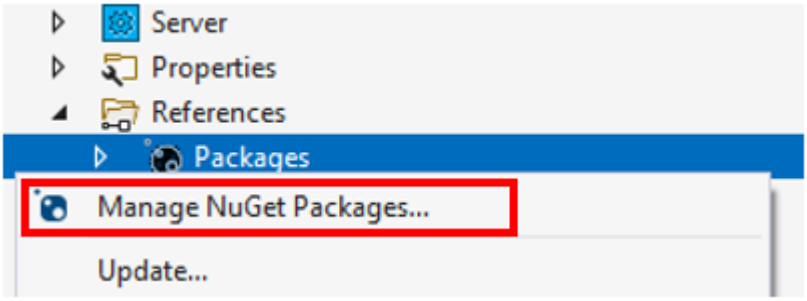
Lizenz	Zweck	Zieltyp
TF2000 | TwinCAT 3 HMI Server	Führt HMI Process Library in TwinCAT HMI aus	Runtime
TF2300 TwinCAT 3 HMI Scope	Repräsentiert Scope-Charts innerhalb von TwinCAT HMI	Runtime
TC1200 | TwinCAT 3 PLC	Führt ein SPS-Programm aus, das Daten für TwinCAT HMI bereitstellt	Runtime
TF8400 |TwinCAT 3 MTP Runtime	Implementierung von richtlinienkonformen MTP-Schnittstellen	Runtime
TF4100 | TwinCAT 3 Controller Toolbox	Implementierung des PID-Reglers	Runtime
TE8400 | TwinCAT 3 MTP Engineering	Entwicklung und Konfiguration von Teilen des MTP-Systems	Engineering
TF6100 | TwinCAT 3 OPC UA	Bietet Interaktion mit POL/DCS über OPC UA	Runtime

2 Installation

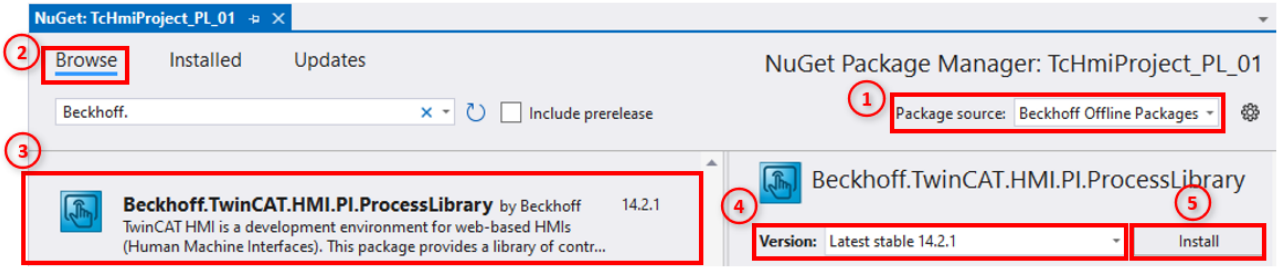
Die Verwendung der TwinCAT HMI Process Library erfordert die Installation von TwinCAT HMI-Komponenten. Siehe die entsprechenden Anweisungen in der Dokumentation für TE2000.

Die Schritte zur Installation eines Pakets für die HMI Process Library werden im Folgenden dargestellt.

1. Für die Installation der HMI Process Library sollte der TwinCAT Package Manager verwendet werden. Er ist als separates Paket **TwinCAT.HMI.PI.ProcessLibrary** oder innerhalb des **TE8400 | TwinCAT 3 MTP Engineering** Workloads erhältlich.
2. **Wählen** Sie im Kontextmenü des **Packages** Knotens die **Manage NuGet Packages** Option aus:

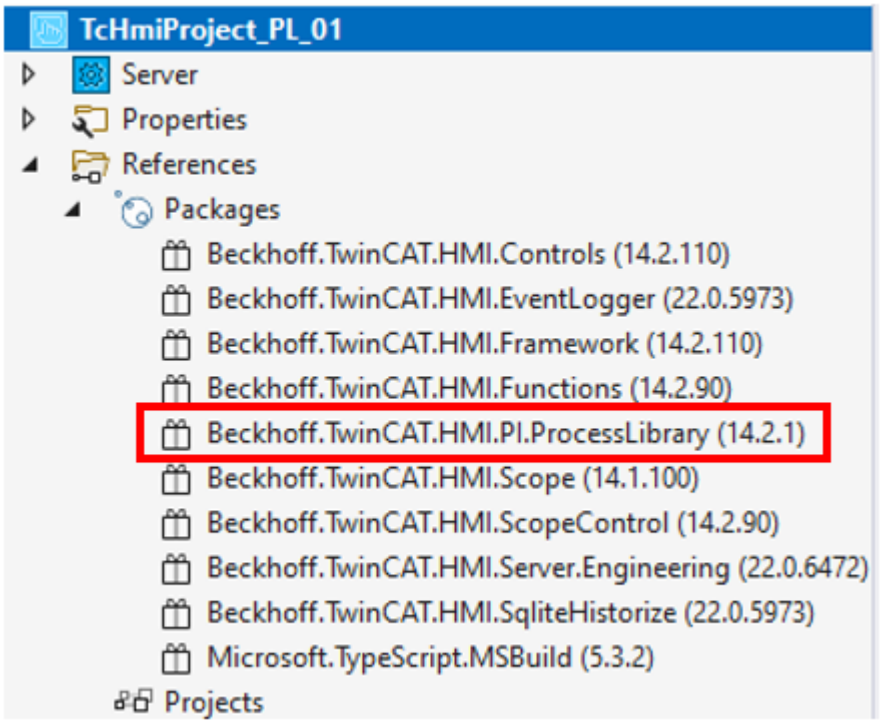


3. Wählen Sie dann die **Package source: Beckhoff Offline Packages** (1) → **Browse**-Registerkarte (2) → **Beckhoff.TwinCAT.HMI.PI.ProcessLibrary** (3) → **Version** auswählen (4) → **Install**-Schaltfläche (5):



⇨ Wenn das HMI Process Library Paket installiert wird, werden auch die davon abhängigen Pakete installiert.

⇨ Das Paket sollte nach der Installation erscheinen:

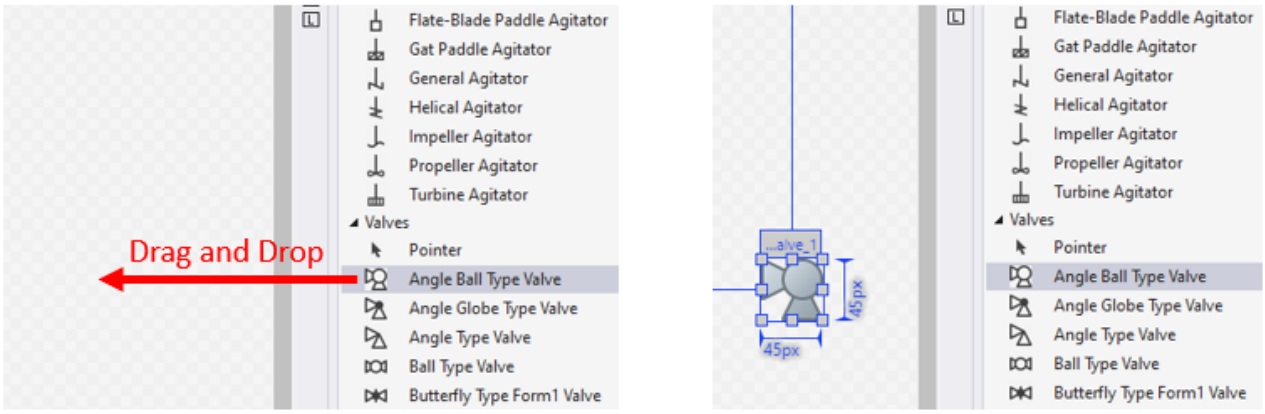


3 QuickStart

Dieses QuickStart-Kapitel enthält grundlegende Schritte, die Ihnen zeigen, wie Sie die Library-Controls zum HMI-Projekt hinzufügen und sie mithilfe der **Data Symbol**-Eigenschaft auf SPS-Daten (z. B. MTP und kundenspezifische FB) abbilden. Darüber hinaus werden weitere grundlegende Basisinformationen bereitgestellt.

3.1 Hinzufügen eines Controls zum Projekt

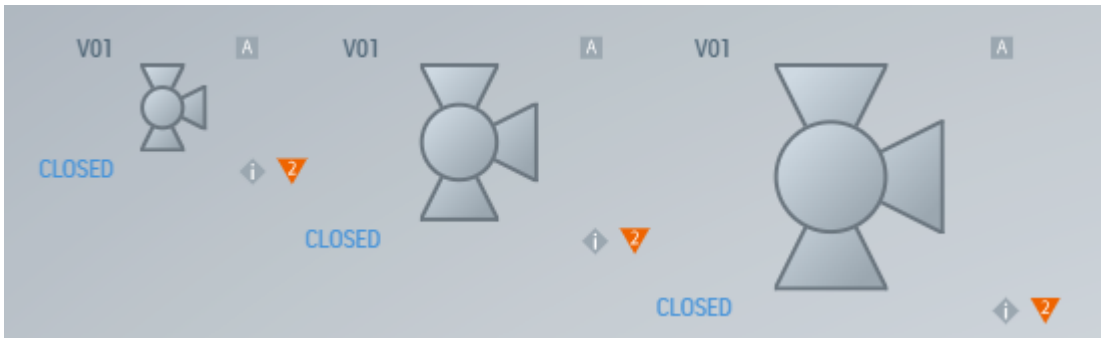
Das Hinzufügen eines Controls aus der HMI Process Library zum Projekt erfolgt auf die gleiche Weise (Drag-and-Drop aus dem **Toolbox**-Panel) wie bei anderen TwinCAT HMI Controls:



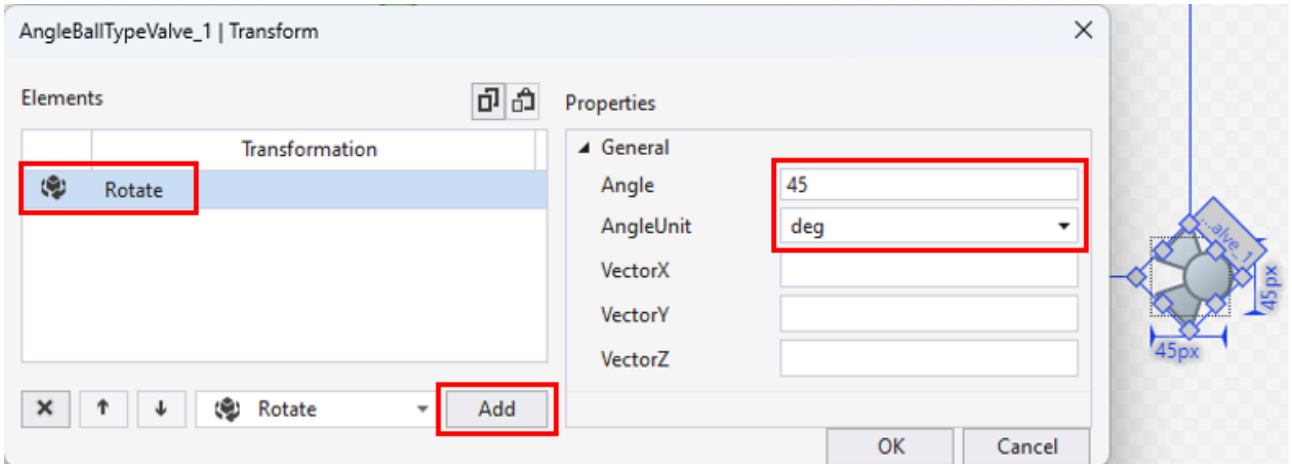
3.2 Bearbeiten der Control-Eigenschaften

Die Eigenschaften, Ereignisse und Berechtigungen sind für die Controls der HMI Process Library in einem **Eigenschaften**-Panel verfügbar. Die Beschreibung der Eigenschaften ist im Kapitel Komponenten und Funktionen [▶ 26] zu finden.

Die Position und Größe des Controls kann über die grafische Schnittstelle oder die entsprechenden Eigenschaften geändert werden.



Mit der **Transform**-Eigenschaft kann das Control gedreht werden:



3.3 Szenarien für die Interaktion mit SPS-Programmen

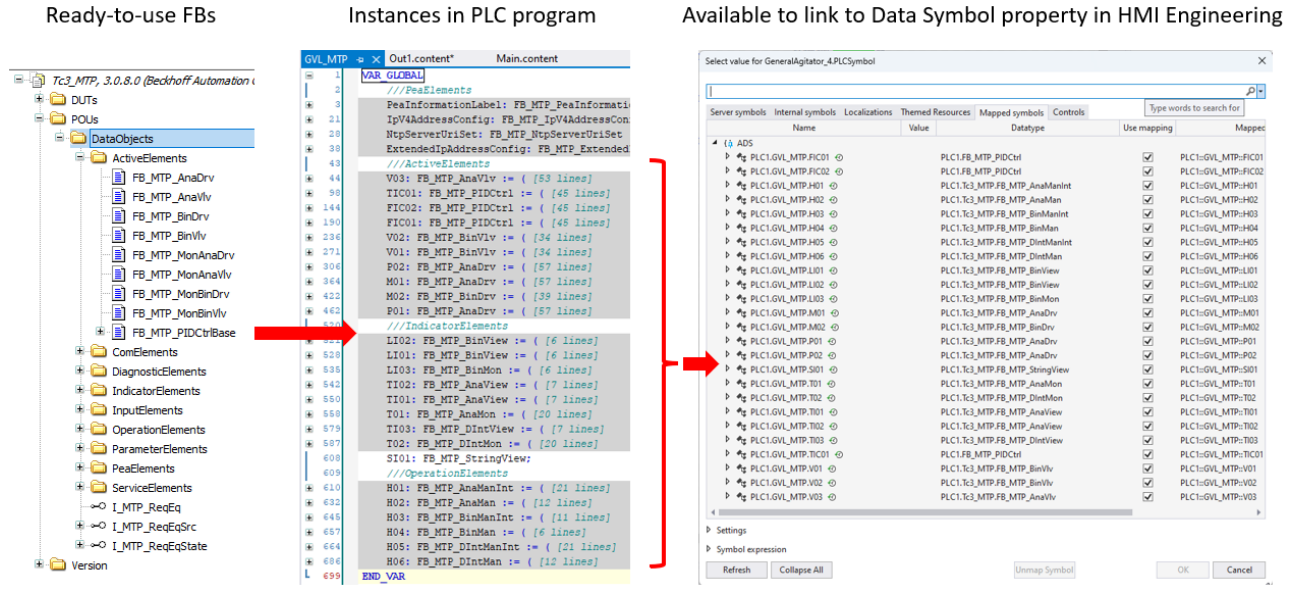
Die wichtigste Eigenschaft, mit der das Verhalten des Controls festgelegt wird, ist das **Data Symbol**. Die Bindung eines FB an die **Data Symbol**-Eigenschaft mit einem geeigneten Satz von Variablen kann ausreichen, um das Verhalten des Controls vollständig zu spezifizieren. Gleichzeitig können aber auch andere Eigenschaften eingestellt werden, um das Verhalten anzupassen. Die Eigenschaften haben Vorrang vor den Variablen der **Data Symbol**-Eigenschaft.

Es gibt zwei Möglichkeiten, einen SPS-Funktionsbaustein an die **Data Symbol**-Eigenschaft zu binden:

- anhand der SPS-Attribut-Funktionalität
- anhand der TwinCAT MTP Runtime

3.3.1 Verwendung der TwinCAT MTP Runtime

Die TwinCAT MTP Runtime (*Tc3_MTP*) bietet eine Reihe von einsatzbereiten FBs, die der MTP-Spezifikation entsprechen. Erstellen Sie eine FB-Instanz für ein bestimmtes DataAssembly in einem SPS-Programm und initialisieren Sie seine Variablen entsprechend der Eigenschaften der HMI Process Library. Nachdem dies geschehen ist und das SPS-Projekt erfolgreich kompiliert wurde, wird dieser FB für die Verknüpfung mit der **Data Symbol**-Eigenschaft verfügbar sein:



Die TwinCAT MTP Runtime implementiert Strukturen von Variablen nach der MTP-Spezifikation und stellt auch Dienste zur Verfügung, die diesem Standard entsprechen. Das bedeutet, dass die FBs Algorithmen zur Steuerung der Geräte enthalten. So entsteht ein komplettes System, von der Geräte-Steuerungsebene bis zum HMI.

3.3.2 Verwendung der SPS-Attribut-Funktionalität

Kundenspezifische SPS-FBs können für die Bindung an die **Data Symbol**-Eigenschaft des Controls der TwinCAT HMI Process Library erstellt werden. Die Verwendung der SPS-Funktionalität in einem kundenspezifischen FB ermöglicht den Zugriff auf die Eigenschaften des Controls. Eine Beschreibung der möglichen Attribute findet sich im Kapitel SPS-Attribut-Funktionalität [▶ 75].

3.3.2.1 Verwendung der SPS-Konfigurationsattribute

Die übliche Form für die Verwendung von SPS-Konfigurationsattributen ist:

```
{attribute 'TcHmi.ProcessLibrary.<Attribute Name>' := '[Value/Values separated by comma]'}
```

Wo werden Konfigurationsattribute verwendet

Verwenden Sie Konfigurationsattribute in der Deklaration. Je nach Typ vor dem Schlüsselwort FUNCTION_BLOCK oder innerhalb des Bereichs VAR_ unmittelbar vor der Deklaration der Variablen, die mit einem Element des Controls verknüpft werden soll.

Beispiele für die Verwendung von SPS-Konfigurationsattributen

Das Pragma kann verwendet werden, um die Attribute zu gruppieren, {region}...{endregion}.

TcHmi.ProcessLibrary.FaceplateControl-Attribut mit dem Region-Pragma:

```
{region 'TcHmiProcessLibrary.FaceplateControl'}
    {attribute 'TcHmi.ProcessLibrary.FaceplateControl.TargetFile':= 'CustomFaceplates/
fbAttrTest_Operate.usercontrol'}
    {attribute 'TcHmi.ProcessLibrary.FaceplateControl.Parameter' := 'DataSymbol:= THISEXP^'}
{endregion}
```

Konfiguration eines Faceplates mit dem Region-Pragma:

```
{region 'TcHmiProcessLibrary.FaceplateParameters'}
    {attribute 'TcHmi.ProcessLibrary. ShowFaceplate' := 'True'}
    {attribute 'TcHmi.ProcessLibrary.Modal' := 'False'}
    {attribute 'TcHmi.ProcessLibrary.Movable' := 'True'}
    {attribute 'TcHmi.ProcessLibrary.RestoreBounds' := 'False'}
    {attribute 'TcHmi.ProcessLibrary.HideWithControl' := 'True'}
    {attribute 'TcHmi.ProcessLibrary.ReshowWithControl' := 'False'}
{endregion}
```

Konfigurieren einer Faceplate-Registerkarte:

```
{attribute 'TcHmi.ProcessLibrary.Tab1.Name' := 'CustomOperate'}
{attribute 'TcHmi.ProcessLibrary.Tab1.TargetFile' := 'CustomFaceplates/
fbAttrTest_Operate.usercontrol,true,true,false'}
{attribute 'TcHmi.ProcessLibrary.Tab1.Parameter' := 'DataSymbol:= THISEXP^'}
{attribute 'TcHmi.ProcessLibrary.Tab1.Alignment' := 'Center,Center'}
{attribute 'TcHmi.ProcessLibrary.Tab1.Icon' := 'PLPATH^/Images/Operate.svg,32,32,px,px'}
```

Konfigurieren einer Faceplate-Registerkarte mit dem einsatzbereiten **Faceplate_Chart.usercontrol**:

```
{attribute 'TcHmi.ProcessLibrary.Tab2.Name' := 'Chart'}
{attribute 'TcHmi.ProcessLibrary.Tab2.TargetFile' := 'Faceplates/
Faceplate_Chart.usercontrol,true,true,false'}
{attribute 'TcHmi.ProcessLibrary.Tab2.Parameter' := 'DataSymbolPath1:=
THIS^.fValue,SclMin:=0,SclMax:=10000,Unit:=STAG^THIS^::nUnitETAG^'}
{attribute 'TcHmi.ProcessLibrary.Tab2.Alignment' := 'Center,Center'}
{attribute 'TcHmi.ProcessLibrary.Tab2.Icon' := 'PLPATH^/Images/Chart.svg,32,32,px,px'}
```

Erstellen einer Variablen für die Historisierung von Charts

Das "Chart"-Attribut sollte vor der Deklaration der Variablen (im Bereich VAR_) hinzugefügt werden. Dazu wird ein Chart zur Historisierung gezeigt:

```
{attribute 'TcHmi.ProcessLibrary.Chart'}
fValue: REAL;
```

Konfigurieren einer Faceplate-Registerkarte mit fertigen Faceplate:

```
{attribute 'TcHmi.ProcessLibrary.Tab3.Name' := 'CustomOperate2'}
{attribute 'TcHmi.ProcessLibrary.Tab3.TargetFile' := 'Faceplates/
BinVlvFaceplate_Operate.usercontrol,true,true,false'}
```

```
{attribute 'TcHmi.ProcessLibrary.Tab3.Parameter' := 'DataSymbol:=STAG^ PLC1.GVL_MTP.V01ETAG^'}
{attribute 'TcHmi.ProcessLibrary.Tab3.Alignment' := 'Center,Center'}
{attribute 'TcHmi.ProcessLibrary.Tab3.Icon' := 'Imports/Images/EX-logo.svg,32,32,px,px'}
```

Konfigurieren des Erscheinungsbildes eines Labels:

```
{region 'TcHmiProcessLibrary.LabelParameters'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextHorizontalAlignment' := 'Right'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextVerticalAlignment' := 'Bottom'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.Left' := '-10'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.LeftUnit' := 'px'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.Top' := '-50'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.TopUnit' := '%'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.Right' := '-10'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.RightUnit' := 'px'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.Bottom' := '-10'}
    {attribute 'TcHmi.ProcessLibrary.LabelTextPadding.BottomUnit' := 'px'}
{endregion}
```

3.3.2.2 SPS-Datenattribute

Deklarieren Sie Variablen innerhalb eines FBs mit dem SPS-Datenattribut `"TcHmi.ProcessLibrary"`, damit sie automatisch mit den Eigenschaften des Controls verknüpft werden. Die vollständige Vorlage lautet:

```
{attribute 'TcHmi.ProcessLibrary' := '[Attribute Name]'}
```

Positionierung der Datenattribute

Verwenden Sie Datenattribute im **VAR_**-Bereich unmittelbar vor der Deklaration der Variablen, die mit einem Element des Controls verknüpft werden soll.

Beispiele für die Verwendung von SPS-Datenattributen

Verknüpfung einer `sName` Variablen mit der `LabelText` Eigenschaft:

```
{attribute 'TcHmi.ProcessLibrary' := 'LabelText'}
sName: STRING := 'CustomPLC';
```

Verknüpfung einer `nStateMode` Variablen mit der `LabelStateMode` Eigenschaft:

```
{attribute 'TcHmi.ProcessLibrary' := 'LabelStateMode'}
nStateMode: INT := 1;
```

Verwendung von SPS-Konfigurationsattributen für Funktionen mit SPS-Datenattributen

Über die Deklaration in einem SPS-Code können Formatierungsfunktionen auf die Eigenschaften angewendet werden. Das Attribut-Pragma `"TcHmi.ProcessLibrary.Format"` sollte verwendet werden:

```
{attribute 'TcHmi.ProcessLibrary.Format' := '[FormattingFunctionName]'}
```

FormattingFunctionName ist der Name einer Formatierungsfunktion, die im Kapitel Funktionen [▶ 72] (Kategorie: Formatierung) aufgeführt ist. Das Attribut mit der Funktion sollte dem SPS-Datenattribut folgen.

Im Folgenden finden Sie einige Beispiele dafür, wie Variablen mit den Attribut-Pragmas deklariert werden können:

```
{attribute 'TcHmi.ProcessLibrary' := 'LabelValue'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4'}
fValue: REAL;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueUnit'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter'}
nUnit: INT := 1001;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValue'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.BinaryValueFormatter,EMERG,NORM'}
_V: BOOL;
```

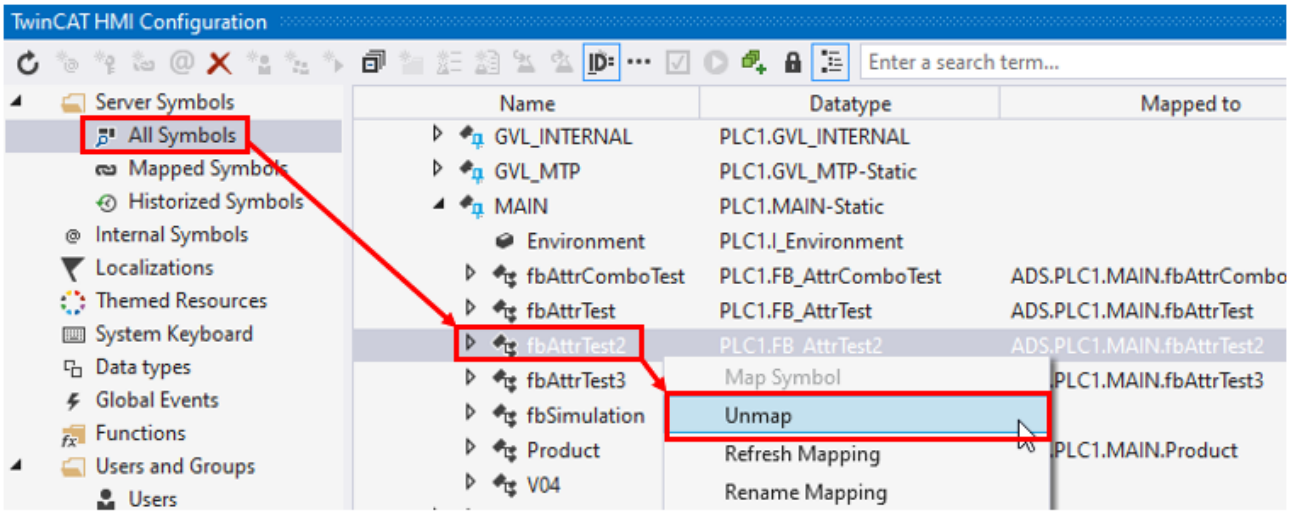
Verwendung von SPS-Attributen, wenn ein Control mehrere Werte anzeigen muss

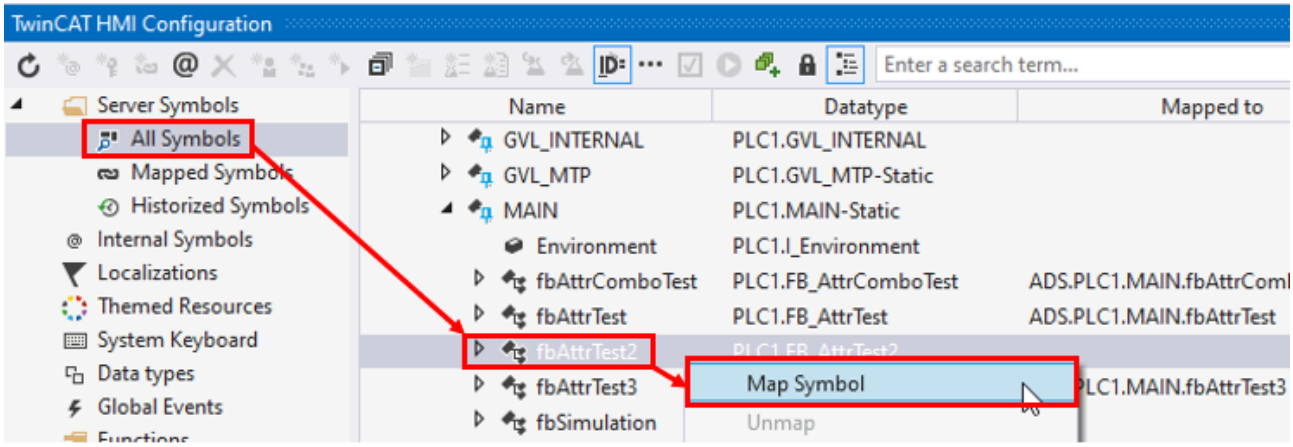
Manchmal müssen mehrere Werte für ein Control-Element angezeigt werden, z.B. bei der PID-Regelung. Um dies zu implementieren, können die Attribute LabelValueAdd1 und LabelValueAdd2 zusammen mit LabelValueUnitAdd1 und LabelValueUnitAdd2 verwendet werden, um Units anzuzeigen. Hier finden Sie Beispiele für die Verwendung der Attribute:

```
{attribute 'TcHmi.ProcessLibrary' := 'LabelValue'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4'}
_SP: REAL;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueAdd1'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4'}
_PV: REAL;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueAdd2'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4'}
_MV: REAL;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueUnit'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter'}
nSPUnit: INT;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueUnitAdd1'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter'}
nPVUnit: INT := 1010;

{attribute 'TcHmi.ProcessLibrary' := 'LabelValueUnitAdd2'}
{attribute 'TcHmi.ProcessLibrary.Format' :=
'TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter'}
nMVUnit: INT := 1342;
```

3.3.2.3 Anwendung der SPS-Attributsfunktionalität Schritt für Schritt

1. Hinzufügen/Bearbeiten von Attributen im SPS-Programm.
2. Erstellen Sie ein SPS-Programm und laden Sie es in die SPS-Runtime.
3. **Wichtig**: Wenn Instanzen des FBs mit hinzugefügten/geänderten Attributen bereits dem TwinCAT HMI zugeordnet sind, führen Sie die Prozedur Unmap/Map für sie im **TwinCAT HMI Configuration**-Fenster aus:

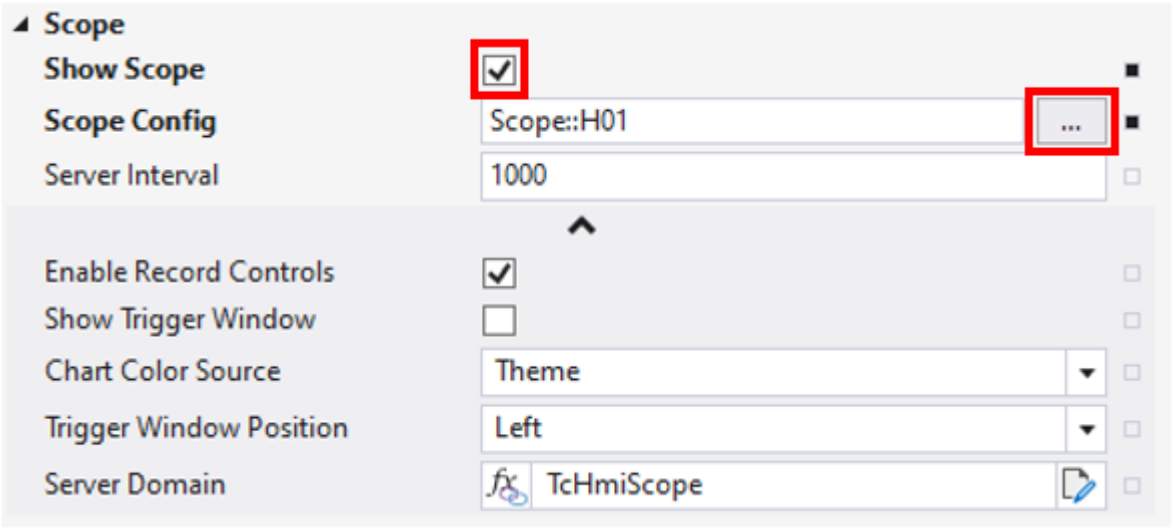




4. Laden Sie die HMI-Seite im Webbrowser neu.

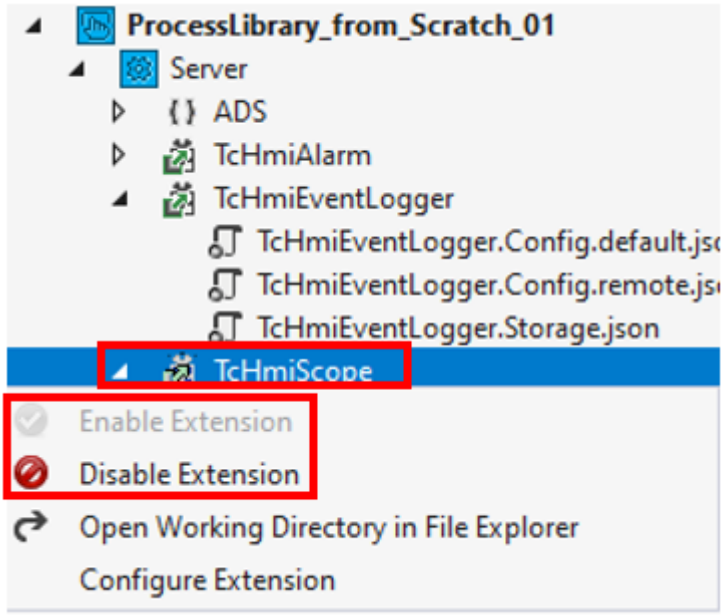
3.4 Scope-Konfiguration

Erstellen Sie ein Scope-Projekt und installieren Sie das **Beckhoff.TwinCAT.HMI.Scope** NuGet-Paket, um Scope-Charts anzuzeigen. Es ist notwendig, die **Show Scope**-Eigenschaft zu aktivieren und eine Konfiguration aus dem Scope-Projekt unter der **Scope Config**-Eigenschaft auszuwählen:



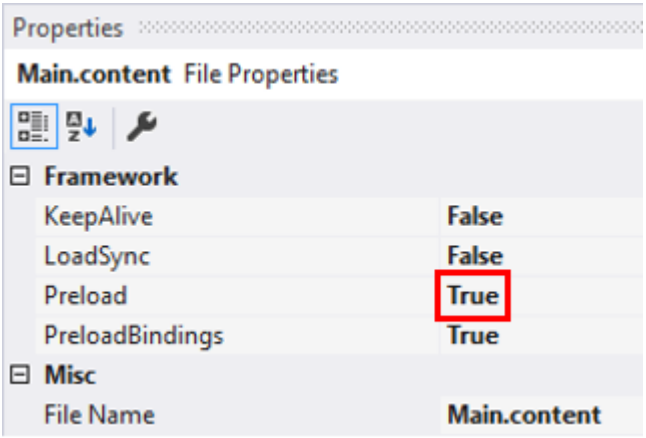
Deaktivieren-Aktivieren der TcHmiScope-Erweiterung

Manchmal ist es erforderlich, die **TcHmiScope**-Erweiterung zu deaktivieren und dann zu aktivieren, nachdem die **Scope Config**-Eigenschaft geändert wurde, um das Scope-Chart im HMI zu sehen.



3.5 Vorladen einer HMI-Seite

Um den Wechsel zwischen den HMI-Seiten reibungsloser und schneller zu gestalten, empfiehlt es sich, die **Preload**-Einstellung für die Inhaltsdateien (**.content*) des Projekts zu aktivieren:

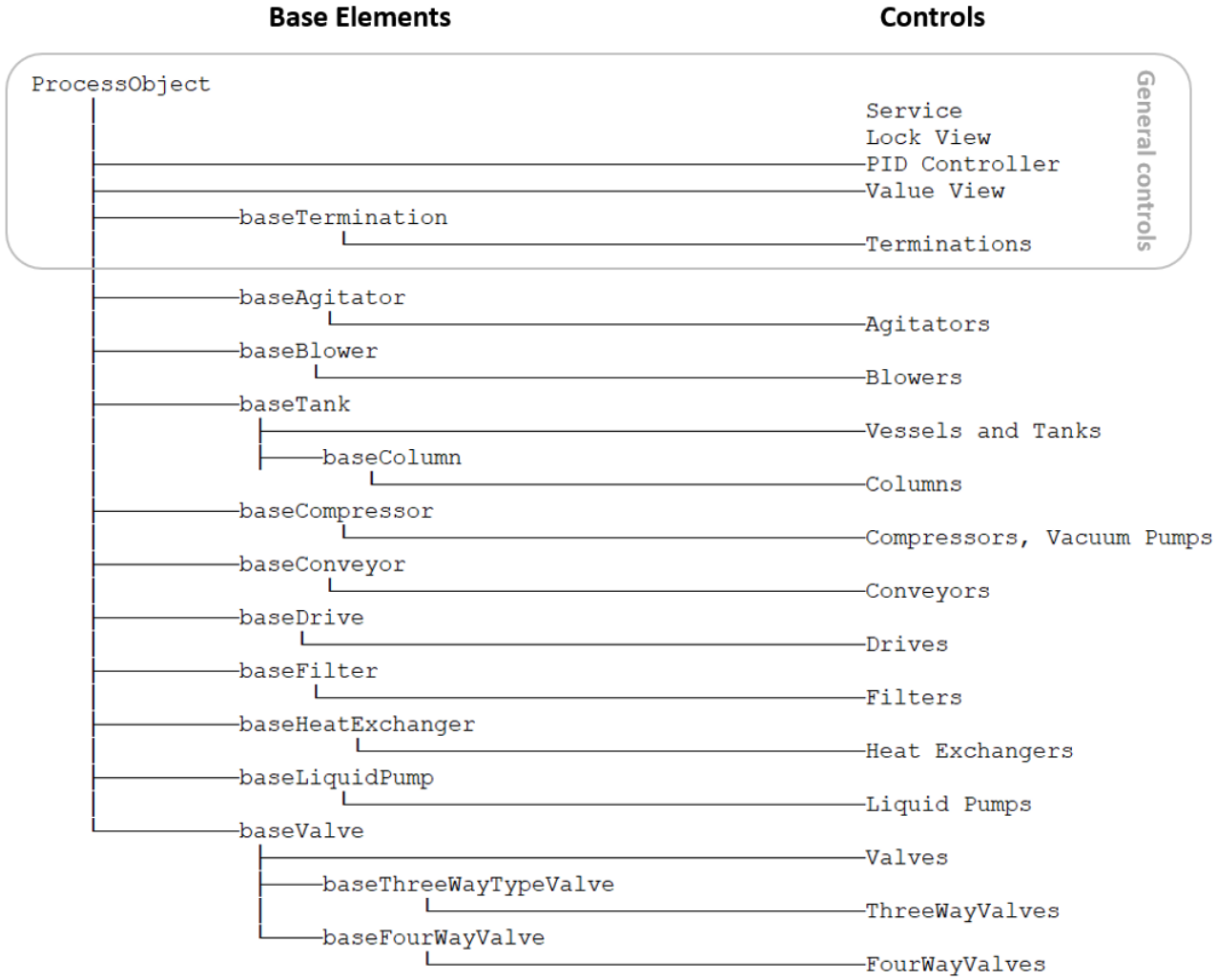


Dies führt zu einem erhöhten Speicherverbrauch auf der Client-Seite. Details finden Sie in der Dokumentation zu TwinCAT HMI (TE2000).

4 Komponenten und Funktionen

Die TwinCAT HMI Process Library besteht aus HMI-Controls und -Funktionen, z. B. Formatierungsfunktionen, die helfen, Werte in einem komfortableren Format darzustellen.

Die meisten HMI-Controls haben eine hierarchische Vererbungsstruktur (nur die Service- und Lock-View-Controls sind getrennt):



Die Vererbung des ProcessObject-Elements

Das ProcessObject-Element ist ein übergeordnetes Element der meisten anderen Controls. Das bedeutet, dass diese Controls ProcessObject-Eigenschaften, -Funktionen und -Ereignisse erben, aber einige von ihnen können in den ererbten Controls nicht verfügbar sein.

Die folgenden Informationen sind so angeordnet, dass die Eigenschaften, Funktionen und Ereignisse der übergeordneten Elemente zuerst im Kapitel ProcessObject [▶ 28] beschrieben werden. Darüber hinaus können diese Mitglieder des ProcessObjects auf die vererbenden Controls angewandt werden und werden nicht für jedes vererbende Control gesondert betrachtet (Kapitel Controls [▶ 41]). Wenn ein Element des übergeordneten Controls für das vererbende Control geändert wird, werden entsprechende Informationen bereitgestellt. Für jedes Control mit seinen besonderen Eigenschaften, Funktionen und Ereignissen werden entsprechende Beschreibungen derer gegeben.

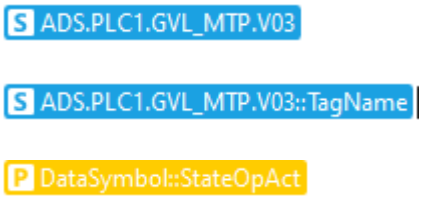
Verwendung von ProcessObject- und Basis-Elementen als Controls

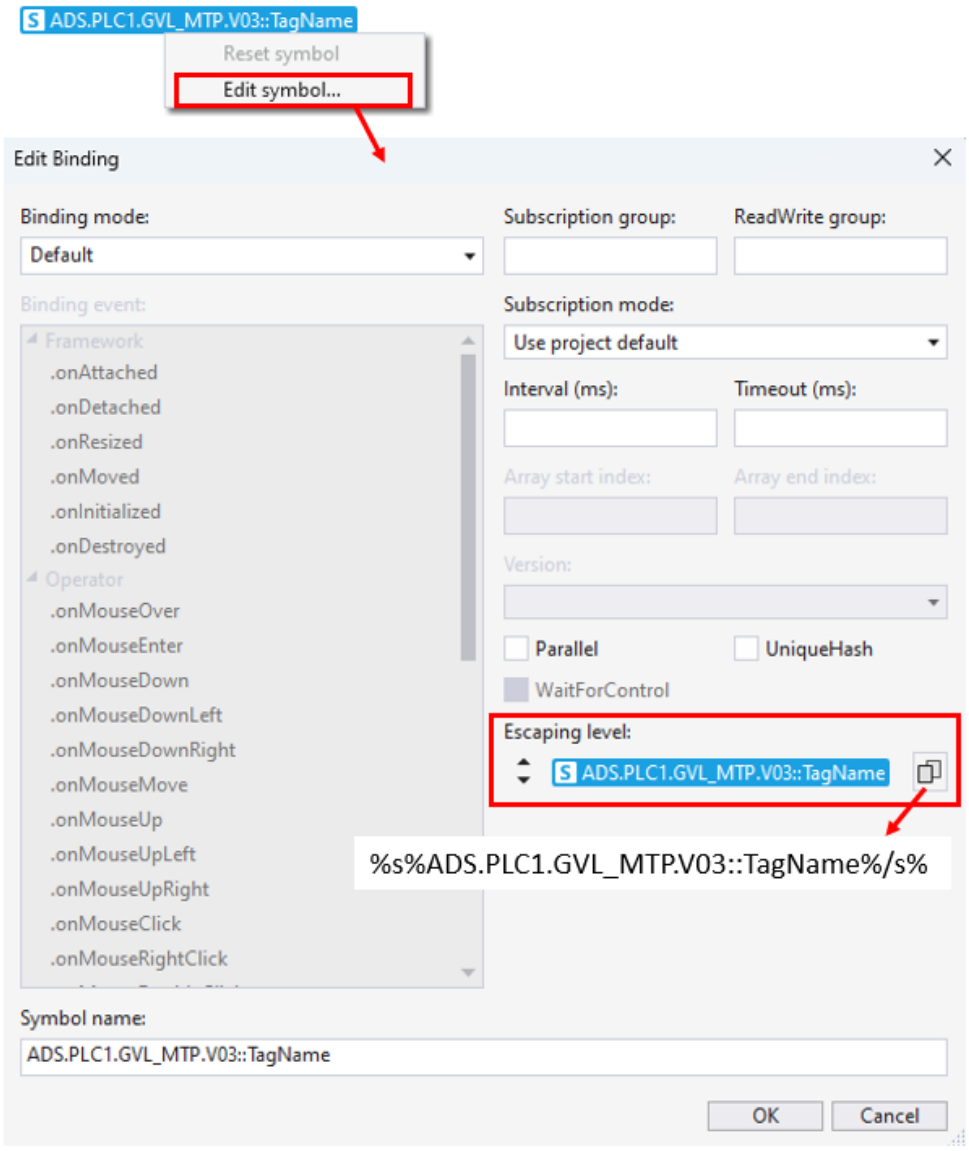
Das ProcessObject kann als Control instanziiert werden, hat aber standardmäßig keine grafische Darstellung. Es erhält sein Faceplate auf der Grundlage der **Data Symbol**-Eigenschaft.

Die Basis-Elemente können nicht als Controls verwendet werden.

4.1 Begriffe

TwinCAT HMI Symbole können in verschiedenen Formen dargestellt werden. Einige Erläuterungen, die in der nachstehenden Tabelle aufgeführt sind, verdeutlichen die Verwendung der Begriffe "Symbol", "SymbolExpression" und "SymbolPath" in diesem Dokument:

Begriff	Erläuterung
Symbol	In TwinCAT HMI ist ein Symbol ein adressierbarer Datenpunkt, der vom HMI-Server angezeigt wird. In der Regel handelt es sich dabei um eine SPS-Variable, die über das TMC exportiert wird, es kann sich aber auch um ein dynamisches oder internes Symbol handeln. Die Symbole können visuell dargestellt werden: • in grafischer Form:  • als SymbolExpression (siehe unten)
SymbolExpression	Eine Darstellung des Symbols als Zeichenkette, die in spezielle Tags (%s%, %pp% usw., je nach Art des Symbols) umgebrochen ist. Beispielsweise: • `%s%ADS.PLC1.GVL_MTP.V03%/s%` • `%s%ADS.PLC1.GVL_MTP.V03::TagName%/s%` • `%pp%DataSymbol::StateOpAct%/pp%` SymbolExpression ist eine Textform der grafischen Darstellung, die im Kontextmenü des Symbols zu finden ist:

Begriff	Erläuterung
	 In einigen Fällen muss SymbolExpression anstelle der grafischen Darstellung verwendet werden.

SymbolPath	Das Symbol wird z. B. als Pfad dargestellt (ohne Umbruch in spezielle Tags):
	`ADS.PLC1.GVL_MTP.V03`
	`ADS.PLC1.GVL_MTP.V03::TagName`
	Manchmal muss der SymbolPath verwendet werden, um das Symbol zu adressieren.

4.2 ProcessObject

Da das ProcessObject-Element von der TcHmi.Controls.System.TcHmiControl Klasse abgeleitet ist, können alle geerbten Eigenschaften in der Dokumentation zu dieser Klasse eingesehen werden. Im Folgenden werden alle Eigenschaften, Funktionen und Ereignisse beschrieben, die für das ProcessObject spezifisch sind und den Benutzern zur Verfügung stehen.

4.2.1 Eigenschaften

Die Eigenschaften der Controls sind in einem **Properties Window** verfügbar.

Priorisierung der Eigenschaften

Standardmäßig werden Daten aus der **Data Symbol**-Eigenschaft verwendet, um das Control und sein Verhalten darzustellen. Wenn jedoch andere Eigenschaften für das Control festgelegt sind, werden diese nach den relevanten Daten der **Data Symbol**-Eigenschaft priorisiert.

4.2.1.1 Kategorie: Common

In der folgenden Tabelle sind die Eigenschaften der Common-Kategorie aufgeführt.

Eigenschaft	Beschreibung
Data Symbol	Ein SPS-FB (nach MTP-Spezifikation oder benutzerdefiniert), der an das Control gebunden werden soll.
Data	Für das Objekt zu verwendende Daten. Diese Eigenschaft wird nicht verwendet, wenn die **Data Symbol**-Eigenschaft konfiguriert ist. Sie ermöglicht es, eine Struktur von Eigenschaften zu konfigurieren, die der Struktur von Standard-MTP-DataObjects entspricht, und diese Eigenschaften können dann separat eingestellt oder gebunden werden.
Graphic Status	Variable, die den Status der Grafik definiert, d. h. Default, Aktiv, Warnung, Fehler. Wenn sich das Verhalten des Controls aus dem **Graphic Status** von dem des ProcessObject unterscheidet, finden Sie die entsprechenden Werte in der Beschreibung des Controls. Entspricht das Verhalten dem des ProcessObject, wird die **Graphic Status**-Eigenschaft in der Control-Beschreibung nicht aufgeführt.
Graphic Control	Der Pfad zur benutzerdefinierten grafischen Darstellung (ein UserControl mit möglichen Parametern, die ausgewertet werden können) des Controls anstelle derjenigen, die der **Data Symbol**-Eigenschaft entnommen wird.
Custom Units	Variable, die eine Übereinstimmung zwischen Integer-Werten und Namen von nicht standardisierten technischen Einheiten definiert. Wenn sie eine Übereinstimmung mit der Nummer aus der Norm enthält, wird sie verwendet und hat Vorrang vor der Standardnummer.

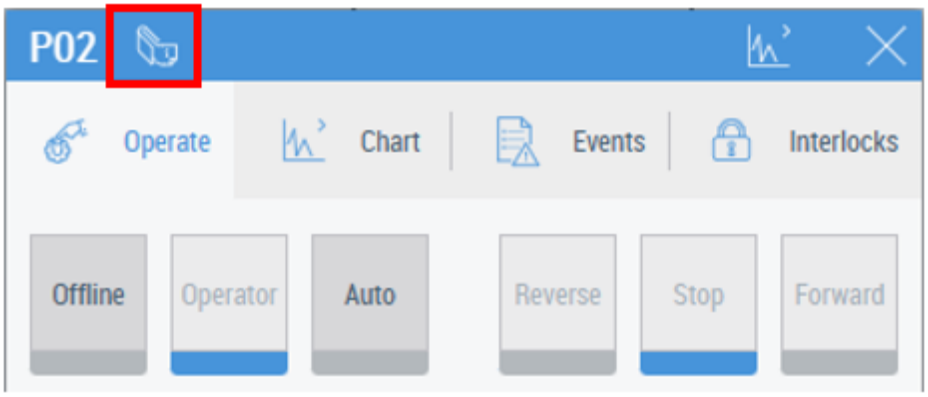
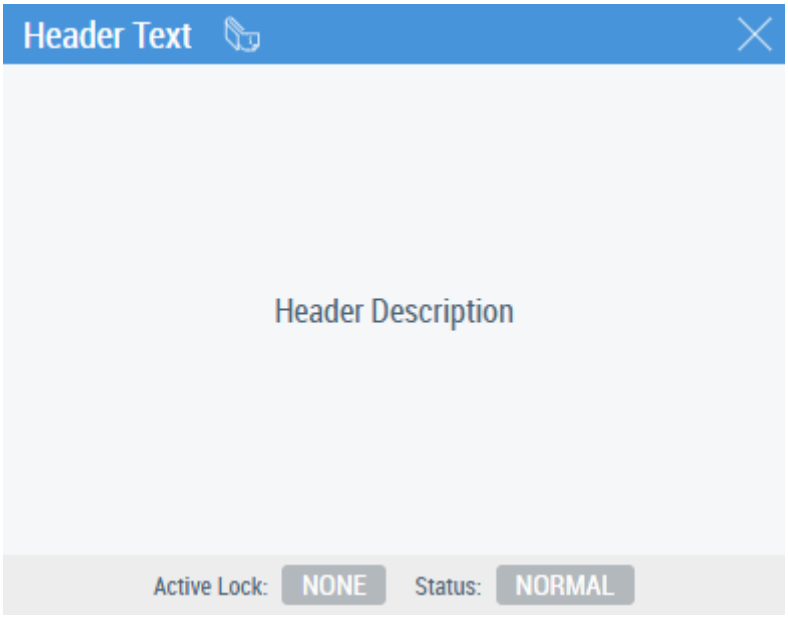
Zusätzlich zum **Graphic Status** wird das Verhalten für ProcessObject anhand eines Antriebsbeispiels gezeigt:

Graphic Status Wert	1	2	3	Jeder andere
Bedeutung	Aktiv	Warnung	Fehler	Default
Bild				

4.2.1.2 Kategorie: Faceplate

In der folgenden Tabelle sind die Eigenschaften der Faceplate-Kategorie aufgeführt

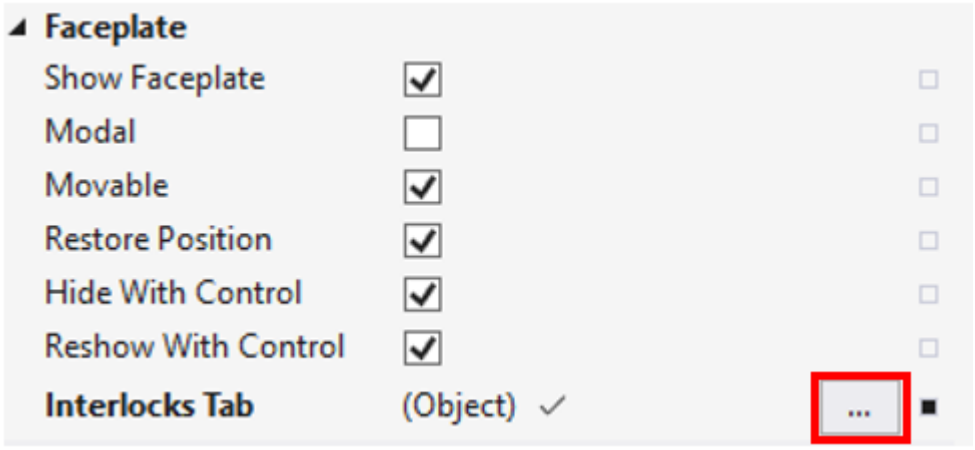
Eigenschaft	Beschreibung
Show Faceplate	Faceplate aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Modal	Legt fest, ob das Popup-Overlay modal ist. Ein modales Popup verdunkelt den Hintergrund und ist nicht verschiebbar. Wenn Sie auf den abgedunkelten Hintergrund klicken, wird das modale Popup-Fenster geschlossen.
Movable	Legt fest, ob das Popup-Overlay beweglich ist.
Restore Position	Legt fest, ob die letzte Begrenzung nach dem Verschieben/der Größenänderung des Popups beim nächsten Öffnen wiederhergestellt wird.
Hide With Control	Blendet das Faceplate automatisch aus, wenn das Control ausgeblendet ist (z. B. beim Umschalten auf eine andere Ansicht). Die Eigenschaften der Seite **Preload** oder zumindest **KeepAlive** *.content müssen auf TRUE gesetzt werden, damit das Faceplate nicht ausgeblendet wird, wenn **Hide With Control** FALSE ist.

Eigenschaft	Beschreibung
Reshow With Control	Zeigt das Faceplate automatisch an, wenn das Control wieder eingeblendet wird und das Faceplate vor dem Ausblenden des Controls angezeigt wurde. Die Eigenschaften der Seite **Preload** oder zumindest **KeepAlive** *.content müssen auf TRUE gesetzt werden, damit das Faceplate wieder erscheint, wenn **Reshow With Control** TRUE ist.
Interlocks Tab	Pfad zu einem Interlock-UserControl, um die Interlocks-Registerkarte des Faceplates anzuzeigen (wenn leer, wird die Interlocks-Registerkarte nicht angezeigt). Diese Eigenschaft ist nur für vordefinierte Faceplates zutreffend. Ausführlichere Informationen finden Sie im Abschnitt Kategorie: Faceplate [▶ 31].
Header Text	Inhalt des Kopfzeilen-Textfeldes im Faceplate.
Show Header Description	Aktivieren/deaktivieren Sie diese Option, um die **Header Description**-Schaltfläche auf dem Faceplate des Controls anzuzeigen:  Die Schaltfläche schaltet die Präsentation der **Header Description** ein und aus.
Header Description	Enthält eine Beschreibung des Controls. Wenn sie über die entsprechende Schaltfläche eingeschaltet ist, wird sie auf dem Faceplate anstelle der anderen Registerkarten angezeigt: 
Show Reset Button	Wenn aktiv, wird die **Reset**-Schaltfläche in der Fußzeile der Meldung angezeigt, unabhängig vom Notfallstatus des Controls.
Faceplate Left	Das Faceplate befindet sich zunächst links (Default-Position ist die linke Position des Controls).

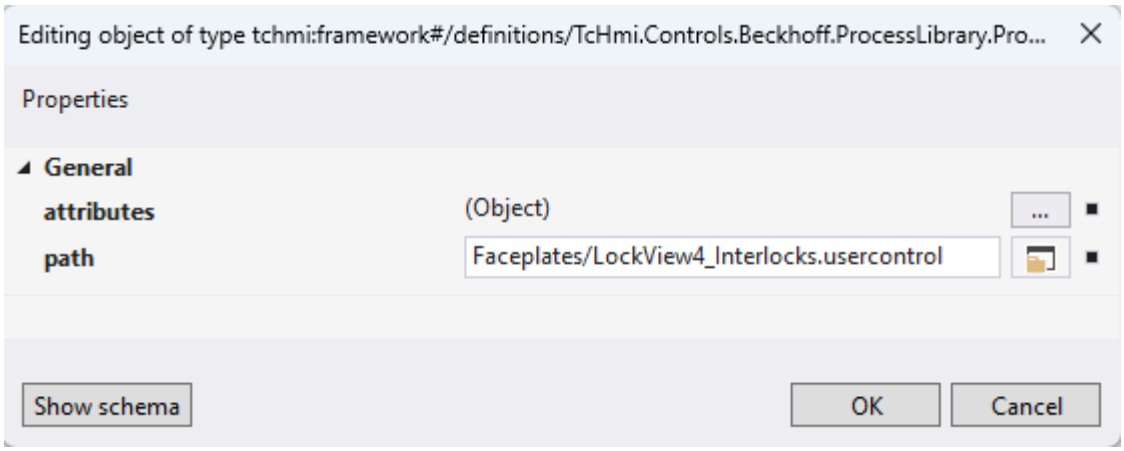
Eigenschaft	Beschreibung
Faceplate Top	Das Faceplate befindet sich zunächst oben (Default-Position ist die obere Position des Controls).
Faceplate Border Radius	Radius des Randes des Faceplates.
Tabs	Eine Liste der Registerkarten, die auf dem Faceplate angezeigt werden sollen. Ausführlichere Informationen finden Sie im Abschnitt Registerkarten-Eigenschaft [▶ 32].
Faceplate Control	Pfad zu einem benutzerdefinierten Faceplate anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft stammt. Überschreibt den gesamten Faceplate-Inhalt, was auch bedeutet, dass die Registerkarten nicht angezeigt werden. Ausführlichere Informationen finden Sie im Abschnitt Faceplate Control-Eigenschaft [▶ 34].

Interlocks Tab-Eigenschaft

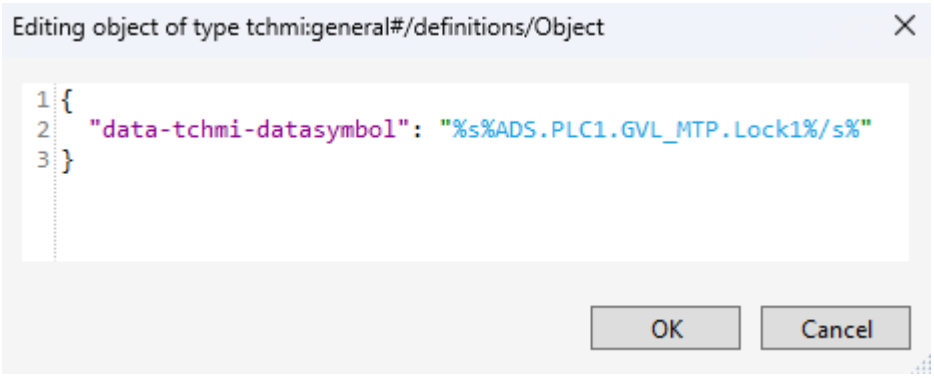
Verwenden Sie die "**...**"-Schaltfläche in den Control-Eigenschaften, um das Fenster für die Interlocks Tab [▶ 66]-Konfiguration zu öffnen.



Zwei Eigenschaften sollten im Konfigurationsfenster konfiguriert werden: **attributes** und **path**:



Die **attributes**-Eigenschaft sollte eine SymbolExpression mit einer SPS-Variable enthalten, die eine Instanz des LockView-FBs aus der Tc3_MTP Library ist: FB_MTP_LockView4, FB_MTP_LockView8, FB_MTP_LockView16 (abhängig von der Anzahl der erforderlichen Verriegelungsgründe). Das Beispiel eines solchen Datensatzes für die SPS-Variable PLC1.GVL_MTP.Lock1 (`Lock1:FB_MTP_LockView4`) ist unten dargestellt:

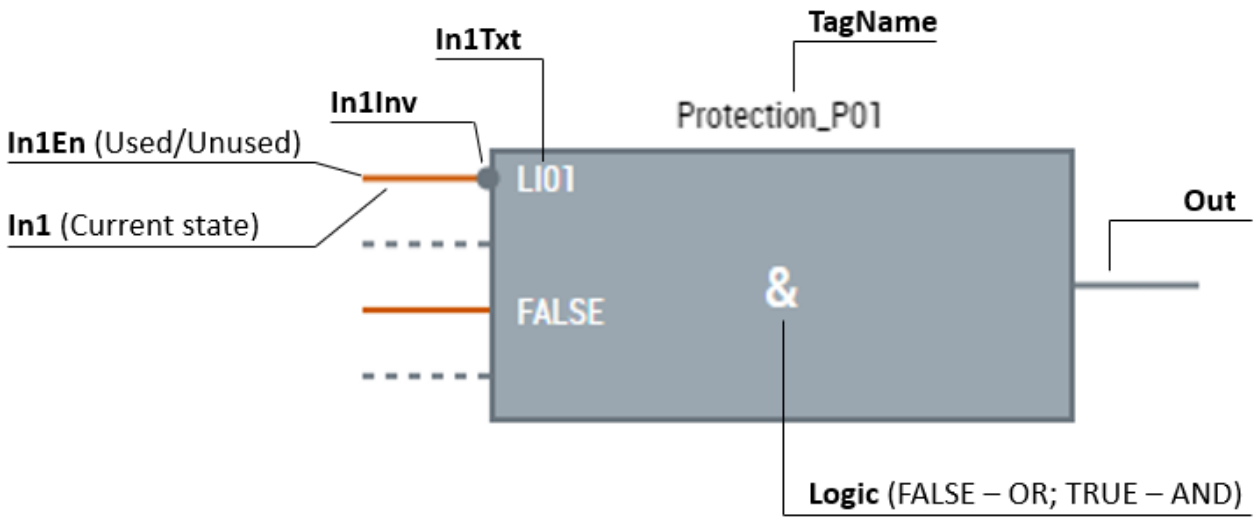


Die **path**-Eigenschaft sollte das LockView-UserControl mit der entsprechenden Anzahl von Eingängen enthalten:

- *Faceplates/LockView4_Interlocks.usercontrol*
- *Faceplates/LockView8_Interlocks.usercontrol*
- *Faceplates/LockView16_Interlocks.usercontrol*

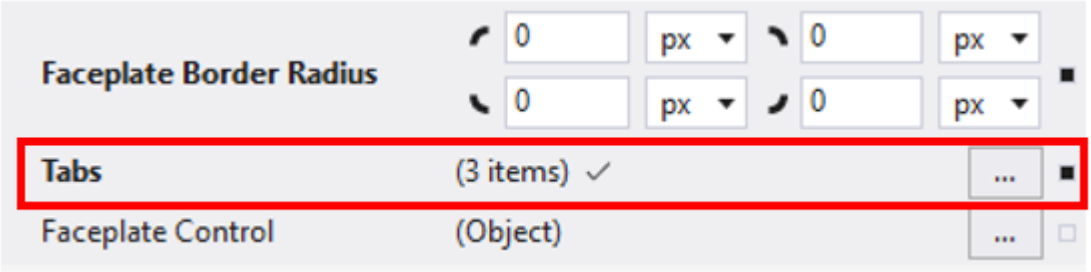
Eigenschaften und aktuelle Zustände des Interlock-Control sollten in den LockView FB-Instanzen eines SPS-Programms angegeben werden. Ein Beispiel für die Übereinstimmung zwischen den FB-Daten und den Kontrollzuständen ist unten dargestellt (die beteiligten FB Variablen sind rot umrandet):

Lock1	FB_MTP_LockView4	
TagName	STRING(255)	'Protection_P01'
TagDescription	STRING(255)	'Shows Interlock...
WQC	BYTE	0
Logic	BOOL	TRUE
In1En	BOOL	TRUE
In1	BOOL	TRUE
In1QC	BYTE	0
In1Inv	BOOL	TRUE
In1Txt	STRING(255)	'LI01'
Out	BOOL	FALSE

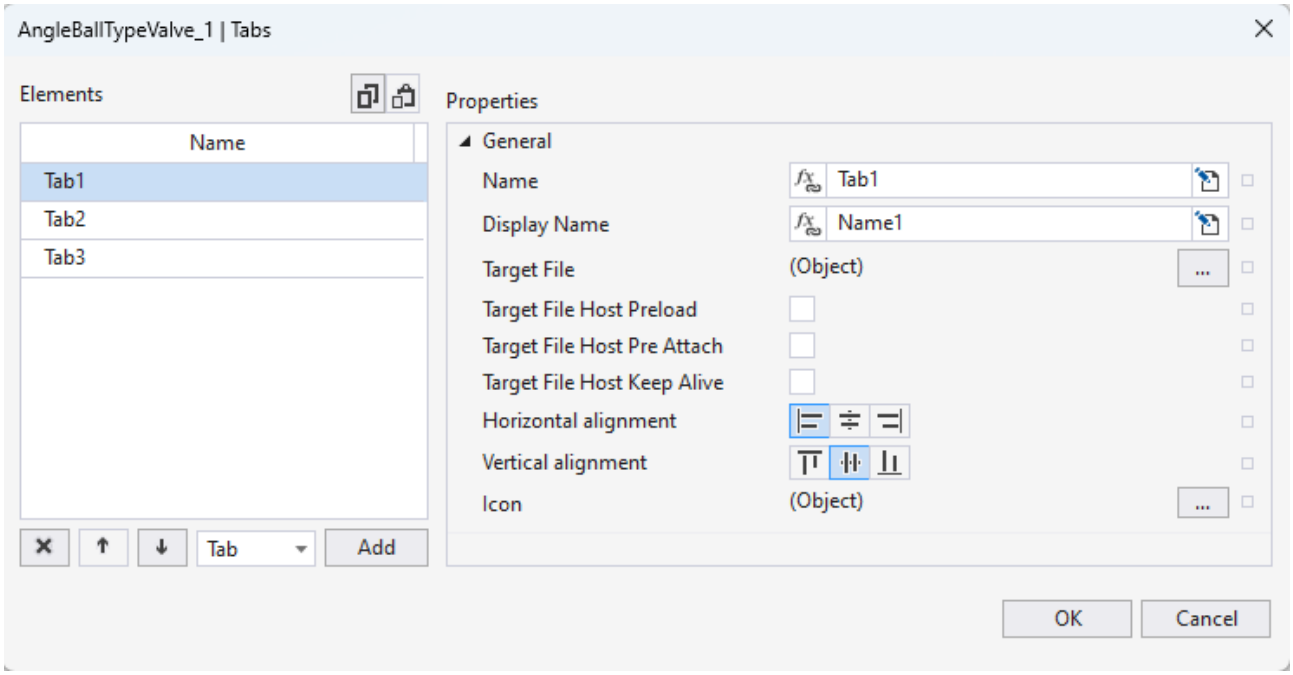


Registerkarten-Eigenschaft

Das Fenster zur Konfiguration der Registerkarten ist über die "..."-Schaltfläche erreichbar.

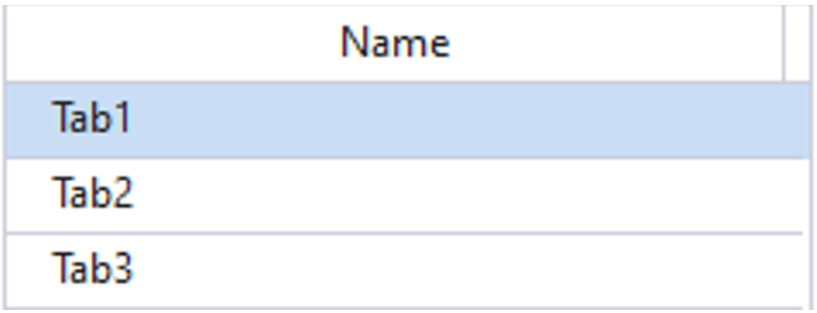
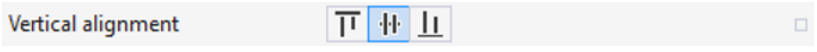


Nach Drücken der "..."-Schaltfläche öffnet sich das Fenster für die Faceplate-Einstellungen:



Die folgende Tabelle zeigt die Beschreibung der Elemente des Fensters für die Faceplate-Einstellungen:

Elemente	Beschreibung
	Die ausgewählte Registerkarte löschen
	Die Reihenfolge der Registerkarten ändern (höher, niedriger)
	Objekttyp (nur Registerkarte)
	Eine neue Registerkarte hinzufügen

Elemente	Beschreibung
	Liste der aktuellen Registerkarten. Tab1 ist aktiv (seine Eigenschaften sind zur Bearbeitung verfügbar).
	Aktuelle Registerkarten kopieren/ einfügen.
	Interner Name der Registerkarte
	Dargestellter Name der Registerkarte im HMI
	Wählen Sie das *.usercontrol- oder *.content-Element als Inhalt für die Registerkarte
	Aktiviert das Vorladen des Inhalts der Registerkarte ***)**
	Aktiviert die Voranbindung des Inhalts der Registerkarte ***)**
	Aktiviert das Keep Alive des Inhalts der Registerkarte ***)**
	Horizontale Ausrichtung für den angezeigten Namen der Registerkarte
	Vertikale Ausrichtung für den angezeigten Namen der Registerkarte
	Icon für die Registerkarte

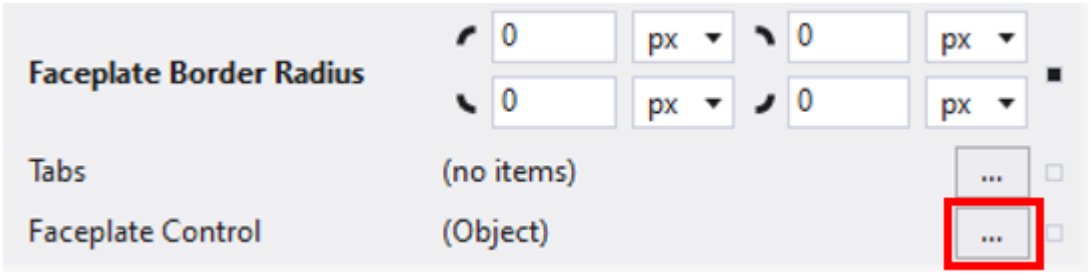
***)** - Ausführlichere Informationen finden Sie im Abschnitt "Lebenszyklus des Controls" im TE2000 TwinCAT 3 HMI-Handbuch

Beispiele für die Konfiguration der **Registerkarten**-Eigenschaft für:

- einsatzbereite Operate- und Monitor-Registerkarten [▶ 53];
- einsatzbereite Chart-Registerkarte [▶ 62];
- einsatzbereite Events-Registerkarte [▶ 65].

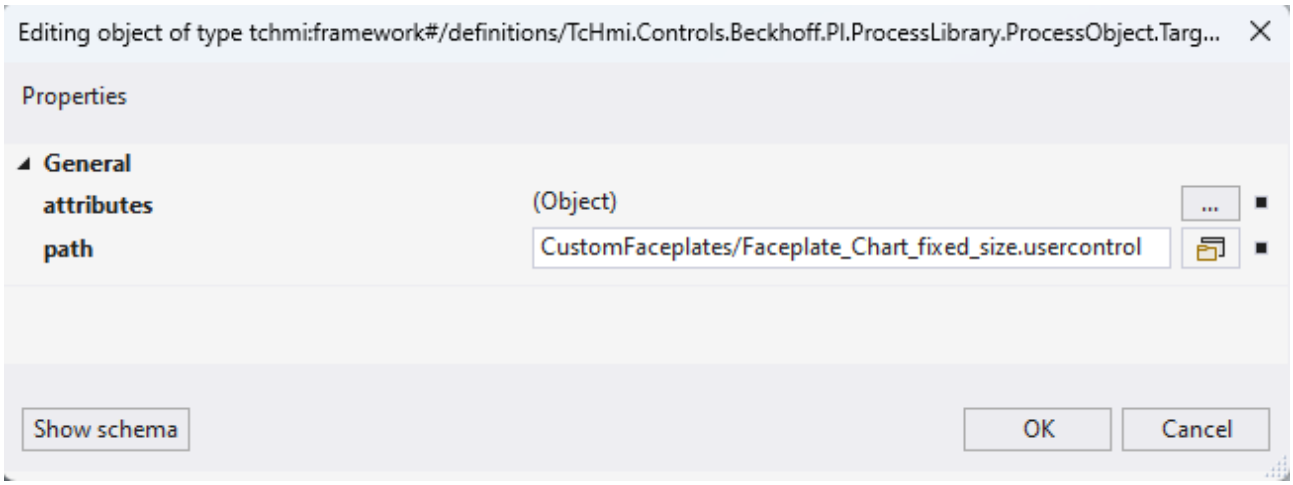
Faceplate Control-Eigenschaft

Verwenden Sie die "**...**"-Schaltfläche in den Control-Eigenschaften, um das Fenster zur Konfiguration der **Faceplate Control**-Eigenschaften zu öffnen.

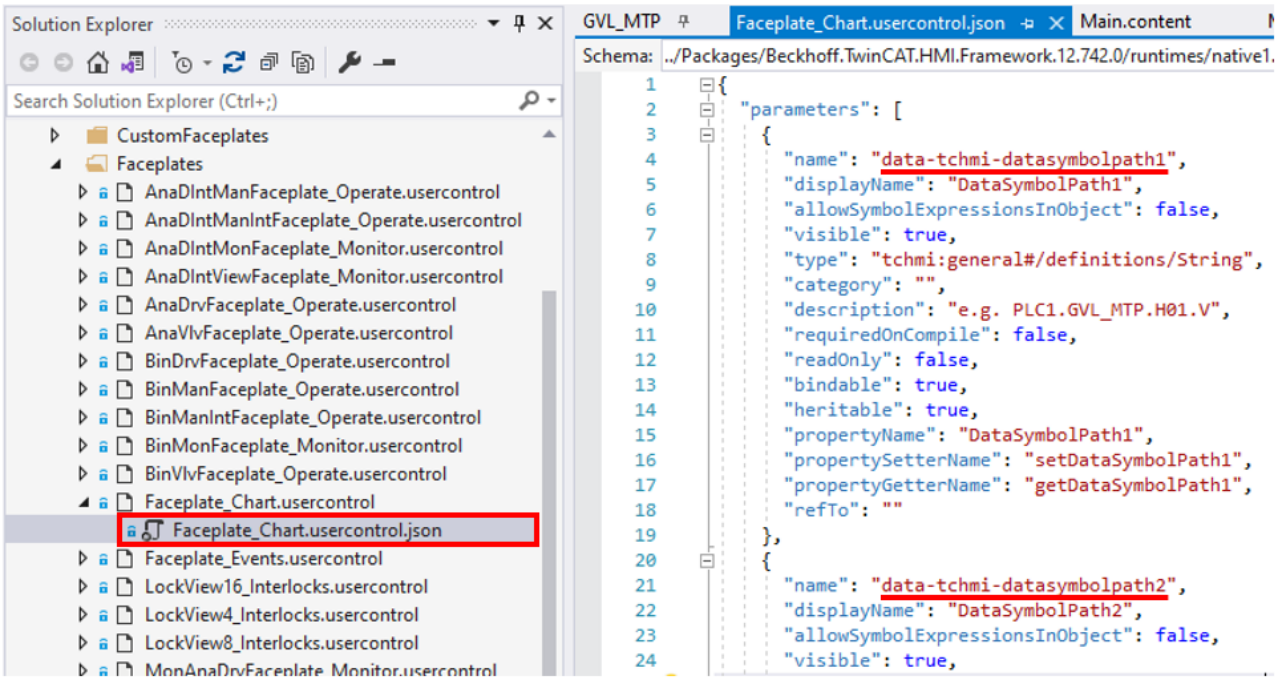


Im Konfigurationsfenster:

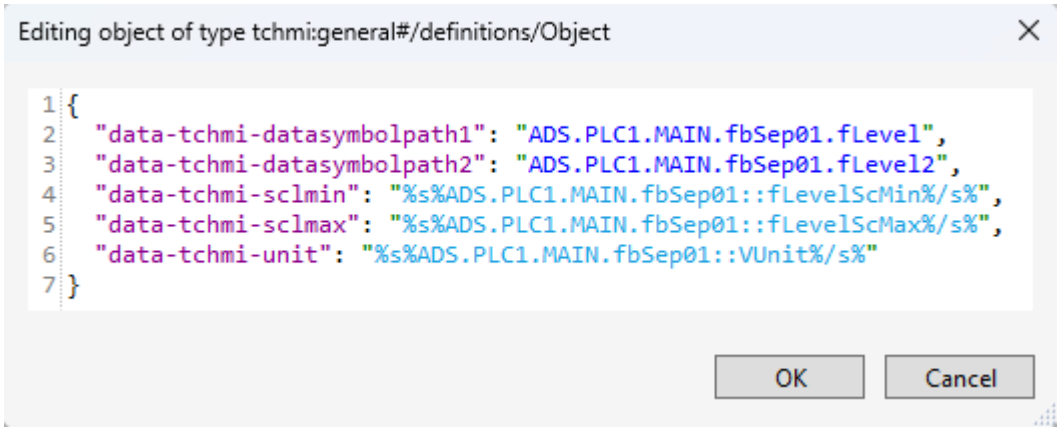
- enthält die **attributes**-Eigenschaft ein JS-Objekt mit Faceplate UserControl-Parametern
- Verweist die **path**-Eigenschaft auf das UserControl, das für das Faceplate verwendet werden soll



Die Namen der Faceplate-Parameter, die für die **attributes**-Eigenschaft benötigt werden, können der **.usercontrol.json*-Datei des Faceplates entnommen werden (die Datei sollte über das Kontextmenü **View Code** im Textmodus geöffnet werden).



Im Folgenden wird ein Beispiel für die Konfiguration der **attributes**-Eigenschaft der **Faceplate Control**-Eigenschaft gezeigt, um *Faceplate_Chart.usercontrol* für das angepasste Faceplate zu verwenden:

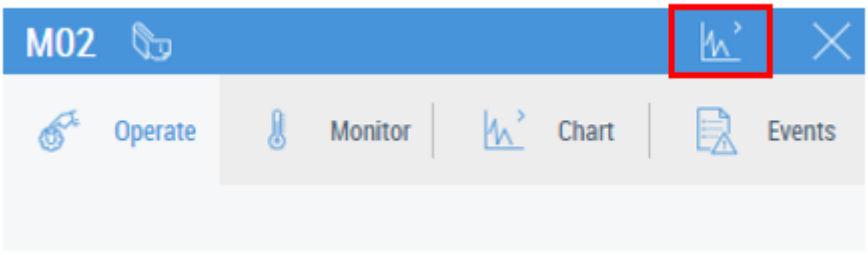


Sehen Sie dazu auch

- Kategorie: Faceplate [▶ 31]
- Kategorie: Faceplate [▶ 32]
- Kategorie: Faceplate [▶ 34]
- Chart-Registerkarte [▶ 60]
- Events-Registerkarte [▶ 64]
- Operate-Registerkarte [▶ 51]
- Monitor-Registerkarte [▶ 54]

4.2.1.3 Kategorie: Scope

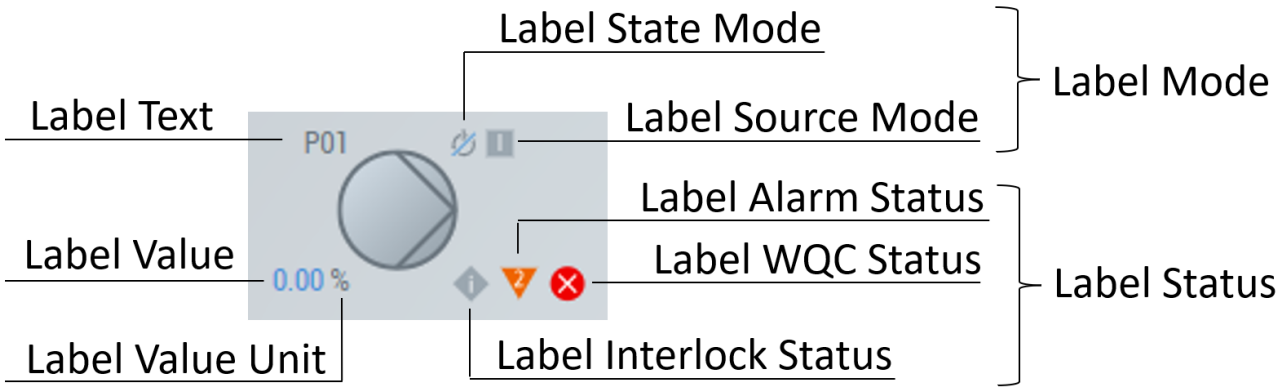
In der folgenden Tabelle sind die Eigenschaften der Scope-Kategorie aufgeführt:

Eigenschaft	Beschreibung
Show Scope	Aktivieren/Deaktivieren der Anzeige der **Scope**-Schaltfläche auf dem Control Faceplate,  die die Scope-Ansicht für die gewählte Konfiguration öffnet (1 - aktivieren; 0 - deaktivieren). Ausführliche Informationen zur Scope Konfiguration finden Sie im Kapitel Scope-Konfiguration [▶ 24]. Wenn die **Show Faceplate**-Eigenschaft deaktiviert ist, wird der Bereich direkt geöffnet, wenn Sie auf das Control klicken.
Scope Config	Die Definition von Scope config mit config name und config chart name.
Server Interval	Zeitintervall in Millisekunden für die Datenerfassung vom Scope-Server.
Enable Record Controls	Schaltet um abhängig davon, ob die Datensatz-Controls angezeigt werden sollen oder nicht.
Show Trigger Window	Schaltet um abhängig davon, ob das Trigger-Fenster angezeigt werden soll oder nicht.
Chart Color Source	Schaltet um abhängig davon, ob die ScopeConfig- oder die Theme-Farben verwendet werden sollen oder nicht.
Trigger Window Position	Gibt an, wo das Trigger-Fenster angezeigt wird. Oben, Unten, Rechts oder Links.

Eigenschaft	Beschreibung
Server Domain	Die Domäne der Scope-Erweiterung auf dem Server. Standardwert ist "TcHmiScope".

4.2.1.4 Kategorie: Label

Ein Beispiel für das Control-Label ist unten abgebildet.



Die folgende Tabelle zeigt die Eigenschaften der Label-Kategorie

Eigenschaft	Beschreibung
Show Label Text	Labeltext aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Label Text	Definiert einen optionalen Labeltext anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft übernommen wird.
Label Text Horizontal Alignment	Definiert die horizontale Ausrichtung des Labeltextes.
Label Text Vertical Alignment	Definiert die vertikale Ausrichtung des Labeltextes.
Show Label Value	Labelwert aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Label Value	Definiert einen optionalen Labelwert anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft übernommen wird.
Label Value Unit	Definiert eine optionale Einheit für den Labelwert anstelle derjenigen, die von der **Data Symbol**-Eigenschaft übernommen wird.
Label Value Format	Eine Formatierungsfunktion, um den Labelwert zu formatieren (kein HTML erlaubt).
Label Value Unit Format	Eine Formatierungsfunktion, um die Einheit des Labelwerts zu formatieren (kein HTML erlaubt).
Label Value Horizontal Alignment	Definiert die horizontale Ausrichtung des Labelwerts.
Label Value Vertical Alignment	Definiert die vertikale Ausrichtung des Labelwerts.
Show Label Mode	Labelmodus aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Label State Mode	Definiert einen optionalen State-Modus des Labels anstelle desjenigen, der von der **Data Symbol**-Eigenschaft übernommen wird. State-Modi [▶ 38].
Label Source Mode	Definiert einen optionalen Sourcemodus des Labels anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft stammt. Source-Modi [▶ 38].
Label Mode Horizontal Alignment	Definiert die horizontale Ausrichtung des Labelmodus.
Label Mode Vertical Alignment	Definiert die vertikale Ausrichtung des Labelmodus.
Show Label Status	Labelstatus aktivieren/deaktivieren (1 - aktiviert; 0 - deaktiviert).
Label Interlock Status	Definiert einen optionalen Interlockstatus des Labels, der aus der **Data Symbol**-Eigenschaft übernommen wird. Interlock-Stati [▶ 38].
Label Interlock Message	Meldung für die aktive Verriegelung in der Fußzeile des Faceplates.
Label Alarm Status	Definiert einen optionalen Alarmstatus des Labels anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft stammt. Alarm-Stati [▶ 38].

Eigenschaft	Beschreibung
Label Alarm Message	Meldung für das Status-Element in der Fußzeile des Faceplates.
Label WQC Status	Definiert einen optionalen WQC-Status des Labels anstelle desjenigen, der aus der **Data Symbol**-Eigenschaft stammt. WQC-Stati [▶ 39].
Label WQC Message	Meldung für das WQC-Element in der Fußzeile des Faceplates.
Label Status Horizontal Alignment	Definiert die horizontale Ausrichtung des Labelstatus.
Label Status Vertical Alignment	Definiert die vertikale Ausrichtung des Labelstatus.
Label Text Padding	Der Abstand des Labeltextes zur ursprünglichen Position.
Label Value Padding	Der Abstand des Labelwerts zur ursprünglichen Position.
Label Mode Padding	Der Abstand des Labelmodus zur ursprünglichen Position.
Label Status Padding	Der Abstand des Labelstatus zur ursprünglichen Position.

Zusätzlich zum State-Modus des Labels (Erklärungen werden für das Verhalten gemäß dem MTP-Standard gegeben, aber ein Benutzer kann das Verhalten durch die Implementierung benutzerdefinierter FBs ändern):

Num	Icon	Bedeutung	Erläuterung
1		Offline	Das Control ist blockiert. Es werden keine Schaltwünsche oder Wertübertragungen berücksichtigt.
2		Bediener (Operator)	Das manuelle Control ist aktiviert. Manuelle (Bediener-)Schaltanfragen werden beachtet.
3		Automatik	Das automatische Control ist aktiviert. Interne (automatische) Schaltanfragen werden beachtet.
Weitere	-	keine	

Zusätzlich zum Sourcemodus für das Label (Erklärungen werden für das Verhalten gemäß dem MTP-Standard gegeben, aber ein Benutzer kann das Verhalten durch Implementierung benutzerdefinierter FB ändern):

Num	Icon	Bedeutung	Erläuterung
1		Intern	Das Control wird durch interne Parameter aktiviert. Interne Control-Anforderungen werden durch automatische Indikatoren nach außen hin sichtbar gemacht.
2		Manual	Das Control wird durch Parameter, die von einem Bediener eingestellt werden, aktiviert.
Weitere	-	keine	

Zusätzlich zum Interlockstatus des Labels (Erklärungen werden für das Verhalten gemäß dem MTP-Standard gegeben, aber ein Benutzer kann das Verhalten durch die Implementierung benutzerdefinierter FB ändern):

Num	Icon	Bedeutung	Erläuterung
1		Erlaubnis	Implementiert eine Einschaltsperre. Dies verhindert die Aktivierung der DataAssembly mit gesperrtem Status. Eine bereits aktivierte DataAssembly mit gesperrtem Status wird jedoch nicht in die sichere Position gebracht.
2		Interlock	Implementiert eine Sperre, die sowohl das Einschalten verhindert als auch eine DataAssembly in die sichere Position bringt. Ein aktives Interlock muss nicht zurückgesetzt werden.
3		Schutz	Implementiert eine Sperre wie ein Interlock, jedoch mit dem Unterschied, dass die Sicherung zurückgesetzt werden muss.
Weitere	-	keine	

Zusätzlich zum Alarmstatus des Labels (Erklärungen werden für das Verhalten gemäß dem MTP-Standard gegeben, aber ein Benutzer kann das Verhalten durch die Implementierung benutzerdefinierter FB ändern):

Num	Icon	Bedeutung	Erläuterung
1		Toleranz	Toleranzgrenze überschritten.

Num	Icon	Bedeutung	Erläuterung
2		Warnung	Warngrenze überschritten.
3		Alarm	Alarmgrenze überschritten.
Weitere	-	keine	

Zusätzlich zum WQC-Status des Labels (Statusnamen werden gemäß den NAMUR-Empfehlungen (NE 184) vergeben, aber ein Benutzer kann das Verhalten durch die Implementierung benutzerdefinierter FB ändern):

Num	Icon	Status
0		Fehler
96		Funktionsprüfung
164		Wartungsanfrage
104		Außerhalb der Spezifikation
28		Außer Betrieb
Weitere	-	keine

4.2.2 Funktionen

Das ProcessObject fügt einige Funktionen hinzu, die den Benutzern in gleicher Weise wie andere Funktionen zur Verfügung stehen.

4.2.2.1 Kategorie: Common

Die Funktionen der Common-Kategorie sind für die Arbeit mit dem Faceplate-Pop-up vorgesehen. Sie sind in der nachstehenden Tabelle aufgeführt:

Funktion	Beschreibung
open	Öffnet das Faceplate.
close	Schließt das Faceplate.
resetBounds	Setzt die Größe und Position des Pop-ups zurück und löscht diese Daten aus dem lokalen Speicher.

4.2.2.2 Kategorie: Scope

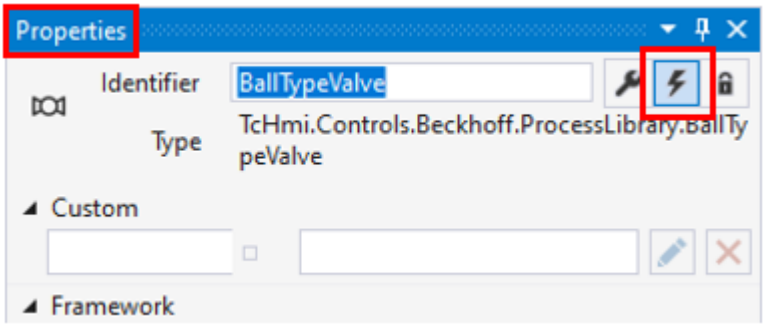
Funktionen aus der Scope-Kategorie helfen bei der Arbeit mit der Scope View-Funktionalität. Sie sind in der nachstehenden Tabelle aufgeführt:

Funktion	Beschreibung
openScope	Öffnet den Scope.
closeScope	Schließt den Scope.
startRecord	Startet die Aufzeichnung auf dem TwinCAT 3 Scope Server.
stopRecord	Beendet die Aufzeichnung auf dem TwinCAT 3 Scope Server.
startDisplay	Startet das Abrufen der Live-Daten aus der Scope-Aufzeichnung.
stopDisplay	Stoppt das Abrufen der Live-Daten aus der Scope-Aufzeichnung.
zoomToDefault	Zoomt auf die Standard-Anzeigebreite.
zoomOutMax	Zoomt auf die derzeit höchste Anzeigebreite.
goTo	Geht zur neuen Startzeit von valueNew. Parameter:

Funktion	Beschreibung
	• valueNew: die neue Startzeit. • Typ: *String* • Format: d,HH:mm:ss,fff:uuu (keine Leerzeichen), wobei: • d – Tage; • HH – Stunden; • mm – Minuten; • ss – Sekunden; • fff – Millisekunden; • uuu – Mikrosekunden.
setDisplayWidth	Legt einen valueNew für die Anzeigebreite fest. Parameter: • valueNew: die neue Anzeigebreite. • Typ: *String* • Format: d,HH:mm:ss,fff:uuu (keine Leerzeichen), wobei: • d – Tage; • HH – Stunden; • mm – Minuten; • ss – Sekunden; • fff – Millisekunden; • uuu – Mikrosekunden.
scroll	Scrollt vorwärts (oder rückwärts). Parameter: • valueNew: die Scroll-Richtung • Typ: *TcHmi.Controls.Beckhoff.TcHmiScopeControl.ScrollDirection*.
scrollBig	Großes Scrollen: (eine ganze Seite) vorwärts (oder rückwärts) für das Scope-Control. Parameter: • valueNew: die Scroll-Richtung • Typ: *TcHmi.Controls.Beckhoff.TcHmiScopeControl.ScrollDirection*.
undo	Macht die letzte Interaktion rückgängig.
redo	Wiederholt die letzte Interaktion.
setOverviewMode	Schaltet die Übersicht um.
setMouseMode	Setzt den Mousemodus für Interaktionen. Parameter: • valueNew: der neue mouseMode • Typ: *TcHmi.Controls.Beckhoff.TcHmiScopeControl.MouseMode*.
getMouseMode	Ruft den Mausmodus ab

4.2.3 Ereignisse

Ereignisse der Controls sind verfügbar, wenn die **Show Events**-Schaltfläche des **Properties**-Fensters gedrückt wird. Das ProcessObject fügt einige Ereignisse (siehe Tabelle unten) in einer Control-Kategorie hinzu.



Die folgende Tabelle zeigt die ProcessObject-Ereignisse

Ereignis	Beschreibung
.onResetPressed	Das onResetPressed-Ereignis wird nach einem (Maus-)Klick oder (Touch) Tipp-Ereignis ausgelöst.
.onOpened	Das onOpened-Ereignis wird ausgelöst, nachdem das Faceplate geöffnet wurde.
.onClosed	Das onClosed-Ereignis wird ausgelöst, nachdem das Faceplate geschlossen wurde.
.onScopeOpened	Das onScopeOpened-Ereignis wird ausgelöst, nachdem der Scope geöffnet wurde.
.onScopeClosed	Das onScopeClosed-Ereignis wird ausgelöst, nachdem der Scope geschlossen wurde.
.onResetStateChanged	Das onResetStateChanged-Ereignis wird ausgelöst, wenn sich der Status der Reset-Schaltfläche geändert hat.
.onResetStatePressed	Das onResetStatePressed-Ereignis wird ausgelöst, wenn sich der Status der Reset-Schaltfläche auf true geändert hat.
.onResetStateReleased	Das onResetStateReleased-Ereignis wird ausgelöst, wenn der Status der Reset-Schaltfläche auf false gewechselt hat.
.onValueChanged	Das onValueChanged-Ereignis wird ausgelöst, wenn sich der LabelValue des Controls geändert hat.

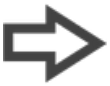
4.3 Controls

Die in der Beschreibung genannten Registriernummern der grafischen Symbole der Controls sind der Norm DIN EN ISO 10628-2 entnommen.

4.3.1 Allgemeine Controls

In der folgenden Tabelle sind allgemeine Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Service		Control, das mit einem Service verknüpft werden kann.
Lock View		Lock View Anzeige mit bis zu 16 Eingängen. Die Funktionalität ist im Kapitel Interlocks Tab [▶ 66] näher beschrieben.
PID Controller		Ein Blick auf die Parameter des PID-Reglers: PV (Prozessvariable), SP (Sollwert), MV (Stellgröße).

Control	Icon	Beschreibung
Value View		Zeigt Prozesswerte in verschiedenen Formaten mit technischen Einheiten an.
Directed Termination		Abschluss des Rohres mit Richtungsangabe. Kann verwendet werden, um zu einer anderen Ansicht zu wechseln, wenn gedrückt.
Termination		Abschluss des Rohres. Kann verwendet werden, um zu einer anderen Ansicht zu wechseln, wenn gedrückt.

4.3.1.1 Lock View

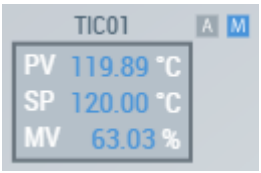
Lock View ist von der *TcHmi.Controls.System.TcHmiControl*-Klasse abgeleitet, daher können alle geerbten Eigenschaften in der Dokumentation dieser Klasse eingesehen werden. Alle spezifischen Lock View-Eigenschaften, die den Benutzern zur Verfügung stehen, sind in der nachstehenden Tabelle beschrieben:

Eigenschaft	Kategorie	Beschreibung
Text Color	Colors	Textfarbe innerhalb des Control-Bausteins.
Active Color	Colors	Füllfarbe des Control-Bausteins, wenn die Sperre aktiv ist.
Lock View Symbol	Common	Eine Lock View PLC FB-Instanz mit 4, 8 oder 16 Eingängen, die an das Control gebunden werden.
Logic	Common	Logische Verknüpfung, die mit den Werten der Control-Eingänge durchgeführt wird (0 - ODER; 1 - UND).
Out	Common	Ausgangswert des Controls (0 - die Sperre ist inaktiv; 1 - die Sperre ist aktiv).
Show Input [N] ***)**	Common	Zeigt den Eingang N an (1 - anzeigen; 0 - ausblenden).
Input [N] Enabled	Common	Aktiviert/deaktiviert den Eingang N (1 - aktivieren; 0 - deaktivieren).
Input [N]	Common	Aktueller Wert am Eingang N (1 - aktivieren; 0 - deaktivieren).
Input [N] Inverted	Common	Invertiert den Wert des Eingangs N (1 - invertieren; 0 - nicht invertieren).
Input [N] Text	Common	Textlabel am Eingang N.
Show Label Text	Label	Labeltext aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Label Text	Label	Definiert einen optionalen Labeltext anstelle des Tag-Namens, der aus dem Lock View-Symbol übernommen wird.

***)** [N] - Anzahl der Eingänge von 4, 8 oder 16

4.3.1.2 PID-Regler

Es gibt keine spezifischen Eigenschaften, Funktionen oder Ereignisse für den PID-Regler, aber für die **Label Value**- und **Label Value Unit**-Eigenschaften, die in die drei Werte (PV, SP und MV) aufzuteilen sind, wird eine spezielle Formatierung verwendet:



Die Werte/Einheiten sollten durch Semikolon getrennt werden:

```
PV value/unit; SP value/unit; MV value/unit
```

4.3.1.3 Werteansicht

Es gibt keine spezifischen Eigenschaften, Funktionen oder Ereignisse für das Value View-Control.

4.3.1.4 Terminations

Das baseTermination-Element erbt von ProcessObject und fügt neue Eigenschaften hinzu, die in der folgenden Tabelle aufgeführt sind. Die Termination- und Directed Termination-Controls wiederum erben Eigenschaften und Verhalten von baseTermination und fügen nichts Neues hinzu.

Die folgende Tabelle zeigt die baseTermination-Eigenschaften:

Eigenschaft	Kategorie	Beschreibung
Target Region	Common	Das Zielbereich-Control, in dem der Inhalt angezeigt wird, wenn das Ereigniszeilen-Control angeklickt oder berührt wird. Typ: *TcHmi.Controls.System.TcHmiRegion.*
Target Content	Common	Pfad zur Ziel-Inhaltsseite, die angezeigt wird, wenn das Ereigniszeilen-Control angeklickt oder berührt wird. Typ: *InhaltPfad.*

4.3.2 Rührwerke

In der folgenden Tabelle sind die Rührwerk-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Anchor Agitator		Ankerrührwerk (ISO C2022).
Cross-Beam Agitator		Querbalkenrührwerk (ISO C2021).
Disk Agitator		Scheibenrührwerk (ISO C2026).
Flate-Blade Paddle Agitator		Rahmenrührwerk (ISO C2019).
Gat Paddle Agitator		Rahmenrührwerk mit Mittelsteg (ISO C2020).
General Agitator		Rührwerk (allgemein) (ISO 2672).
Helical Agitator		Wendelrührwerk (ISO C2023).
Impeller Agitator		Impellerrührwerk (ISO C2024).

Control	Icon	Beschreibung
Propeller Agitator		Propellerrührwerk (ISO C2025).
Turbine Agitator		Turbinenrührwerk (ISO C2027).

Alle zusätzlichen Eigenschaften von Rührwerk-Controls, im Vergleich zu ProcessObject, sind im baseAgitator-Element implementiert. Andere Rührwerk-Controls erben diese Eigenschaften und haben keine zusätzlichen.

4.3.2.1 baseAgitator

Das baseAgitator-Element fügt eine Rotation-Kategorie mit einigen Eigenschaften hinzu. Siehe folgende Tabelle:

Eigenschaft	Beschreibung
Show Rotation	Aktiviert/deaktiviert die Drehung (1 - aktivieren; 0 - deaktivieren)
Rotation Duration	Definiert die Rotationsdauer in Millisekunden.

Animation der Drehung

Wenn **Show Rotation** aktiviert ist, hängen die Bewegung und die Drehzahl der Animation von anderen Eigenschaften ab.

Ob sich das Rührwerk bewegen soll oder nicht, wird durch die **Data Symbol**- oder **Label Value**-Eigenschaften festgelegt. **Label Value** hat Vorrang. Ist der Wert gleich 0, gibt es keine Bewegung. Andernfalls sollte sich das Rührwerk bewegen.

Die Drehzahl der Animation hängt vollständig von der **Rotation Duration**-Eigenschaft ab.

Eigenschaften des Grafikstatus

Die **Graphic Status**-Eigenschaft ist für die Rührwerk-Controls nicht verfügbar.

4.3.3 Gebläse und Lüfter

In der folgenden Tabelle sind die Gebläse- und Lüfter-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
General Blower		Gebläse, Lüfter (allgemein) (ISO X8164).

Das baseBlower-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Andere Gebläse- und Lüfter-Controls fügen baseBlower nichts anderes hinzu.

4.3.4 Kolonnen

In der folgenden Tabelle sind die Kolonnen-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
General Column		Kolonne (allgemein) (ISO X8100).

Control	Icon	Beschreibung
General Tray Column		Bodenkolonne (allgemein) (ISO X8101).

Das baseColumn-Element erbt von baseTank [▶ 49] und fügt diesem gegenüber keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Andere Kolonnen-Controls fügen nichts anderes zu baseColumn hinzu.

4.3.5 Kompressoren und Vakuumpumpen

In der folgenden Tabelle sind die Kompressor- und Vakuumpumpen-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Centrifugal Compressor		Zentrifugalkompressor (ISO X8179).
Diaphragm Compressor		Membrankompressor, Vakuumpumpe mit Membran (ISO X8099).
General Compressor		Kompressor, Vakuumpumpe (allgemein) (ISO 2302).
Jet Compressor		Ejektionskompressor, Strahl-Vakuumpumpe (ISO X8163).
Liquid Ring Compressor		Kompressor, Flüssigkeitsring-Vakuumpumpe (ISO X8162).
Reciprocating Piston Compressor		Kompressor, Hubkolben-Vakuumpumpe (ISO X8097).
Roller Vane Compressor		Kompressor, Rollflügel-Vakuumpumpe (ISO X8104).
Rotary Compressor		Rotationskompressor, Drehkolben-Vakuumpumpe (ISO X8105).
Screw Compressor		Rotationskompressor, Drehkolben-Vakuumpumpe (ISO X8161).
Turbo Compressor		Kompressor, Turbo-Vakuumpumpe (ISO X8102).

Das baseCompressor-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Auch andere Kompressor- und Vakuumpumpen-Controls fügen dem baseCompressor nichts anderes hinzu.

4.3.6 Förderer

In der folgenden Tabelle sind die Förderer-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Screw Conveyor		Schneckenförderer, geschlossen (ISO X8063).

Alle zusätzlichen Eigenschaften der Förderer-Controls, im Vergleich zu ProcessObject, sind im baseConveyor-Element implementiert. Andere Förderer-Controls erben diese Eigenschaften und haben keine zusätzlichen Eigenschaften.

4.3.6.1 baseConveyor

Das baseConveyor-Element fügt eine Rotation-Kategorie mit einigen Eigenschaften hinzu. Siehe folgende Tabelle:

Eigenschaft	Beschreibung
Show Rotation	Aktiviert/deaktiviert die Drehung (1 - aktivieren; 0 - deaktivieren)
Rotation Duration	Definiert die Rotationsdauer in Millisekunden.

Animation der Drehung

Wenn **Show Rotation** aktiviert ist, hängen die Bewegung und die Drehzahl der Animation von anderen Eigenschaften ab.

Ob sich das Förderband bewegen soll oder nicht, wird durch die DataSymbol- oder **Label Value**-Eigenschaften festgelegt. **Label Value** hat Vorrang. Ist der Wert gleich 0, gibt es keine Bewegung. Andernfalls sollte sich der Förderer bewegen.

Die Drehzahl der Animation hängt vollständig von der **Rotation Duration**-Eigenschaft ab.

4.3.7 Antriebe

In der folgenden Tabelle sind die Antrieb-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
AC Generator		Generator, AC (ISO X8154).
AC Motor		Elektro-Motor, AC (ISO X8157).
DC Generator		Generator, DC (ISO X8155).
DC Motor		Elektrischer Motor, DC (ISO X8158).
Gear		Getriebe (ISO X8167).
General Generator		Generator (allgemein) (ISO C0079).
General Motor		Elektromotor (allgemein) (ISO C0082).

Control	Icon	Beschreibung
General Turbine		Turbine (allgemein) (ISO 2571).

Das baseDrive-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Auch andere Antrieb-Controls fügen baseDrive nichts anderes hinzu.

4.3.8 Filter

In der folgenden Tabelle sind die Filter-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Bag Liquid Filter		Flüssigkeitsfilter, Beutel, Kerze oder Kartusche (ISO X8117).
General Liquid Filter		Flüssigkeitsfilter (allgemein) (ISO X8116).

Das baseFilter-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Andere Filter-Controls fügen baseFilter nichts anderes hinzu.

4.3.9 Heizen und Kühlen

In der folgenden Tabelle sind die Heiz- und Kühl-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Electrical Heater for Vessel		Elektrische Heizung für Behälter mit gewölbten Böden und elektrischer Beheizung (ISO X2070). Ist zur Kombination mit Behältern vorgesehen.
General Heat Exchanger		Wärmetauscher (allgemein), Verflüssiger (ISO X8079).
Heating/Cooling Jacket for Vessel		Heiz-/Kühlmantel für Behälter mit gewölbten Böden und Heiz-/Kühlmantel (ISO X2069). Ist zur Kombination mit Behältern vorgesehen.
Heating/Cooling Jacket for Vessel Dished		Heiz-/Kühlmantel (gewölbter Boden) für Behälter mit gewölbtem Boden und Heiz-/Kühlmantel (ISO X2069). Ist zur Kombination mit Behältern vorgesehen.
M-Shape Heat Exchanger		Wärmetauscher (M-Form) (ISO X8017).
Straight Tubes Heat Exchanger		Wärmetauscher mit geraden Rohren (Platten mit festen Rohren) (ISO 2511).

Das baseHeatExchanger-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Die Heiz- und Kühl-Controls fügen baseHeatExchanger nichts anderes hinzu.

4.3.10 Flüssigkeitspumpen

In der folgenden Tabelle sind die Flüssigkeitspumpen-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Centrifugal Pump		Zentrifugalpumpe (ISO 2322).
Diaphragm Pump		Membranpumpe (ISO X8095).
Gear Pump		Zahnradpumpe (ISO X8091).
Jet Pump		Ejektor-Strahlpumpe für Flüssigkeiten (ISO X8096).
Liquid Pump		Flüssigkeitspumpe (allgemein) (ISO 2301).
Progressive Cavity Pump		Exzenterschneckenpumpe (ISO X8093).
Reciprocating Piston Pump		Hubkolbenpumpe (ISO X8094).
Screw Pump		Schneckenpumpe (ISO X8092).

Das baseLiquidPump-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Andere Flüssigkeitspumpen-Controls fügen der baseLiquidPump nichts anderes hinzu.

4.3.11 Behälter und Tanks

In der folgenden Tabelle sind die Behälter- und Tank-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Container		Container, Tank, Zisterne (ISO 2061).
General Tank		Tank, Behälter (ISO 301).
Spherical Vessel		Kugelförmiger Behälter (ISO 2063).
Spherical Vessel On Legs		Kugelförmiger Behälter auf Füßen (ISO X8010).
Tank Dished Ends		Tank, Behälter mit gewölbten Böden (ISO 2062).
Vessel Dished Ends On Legs		Behälter mit gewölbten Böden und Stützfüßen (ISO X8002).
Vessel Dished Roof Conical Bottom		Behälter mit gewölbtem Dach und konischem Boden (ISO X8008).

Alle zusätzlichen Eigenschaften der Behälter- und Tank-Controls im Vergleich zu ProcessObject sind im baseTank-Element implementiert. Behälter- und Tank-Controls erben diese Eigenschaften und haben keine zusätzlichen Eigenschaften.

4.3.11.1 baseTank

Das baseTank-Element fügt zur Colors-Kategorie eine Fill Level-Kategorie mit einigen Eigenschaften und der **Fill Level Color**-Eigenschaft hinzu. Siehe folgende Tabelle:

Eigenschaft	Kategorie	Beschreibung
Fill Level Color	Colors	Die Farbe des angezeigten Füllstands.
Show Fill Level	Fill Level	Füllstand aktivieren/deaktivieren (1 - aktivieren; 0 - deaktivieren).
Fill Level Scale Min	Fill Level	Legt das Minimum der Füllhöhe für die Skalierung der Füllstandanimation fest.
Fill Level Scale Max	Fill Level	Legt das Maximum der Füllhöhe für die Skalierung der Füllstandsanimation fest.

Füllstandsanimation

Wenn **Show Fill Level** aktiviert ist, wird der Füllstand im Behälter entsprechend der DataSymbol- oder **Label Value**- Eigenschaft animiert. Die **Label Value**-Eigenschaft hat Vorrang, wenn sie gesetzt ist. Für die Skalierung werden die entsprechenden Daten aus DataSymbol oder, vorrangig, die **Fill Level Scale Min**- und **Fill Level Scale Max**-Eigenschaften implementiert, aber nur, wenn beide gesetzt sind.

4.3.12 Ventile

In der folgenden Tabelle sind die Ventil-Controls und ihre Beschreibungen aufgeführt.

Control	Icon	Beschreibung
Angle Ball Type Valve		Eck-Kugelventil (ISO X8072).
Angle Globe Type Safety Valve		Sicherheitsventil, federbelastet, Eck-Durchgangsventil (ISO X2125).
Angle Globe Type Valve		Eck-Durchgangsventil (ISO X8069).
Angle Type Valve		Eckventil (allgemein) (ISO 2102).
Ball Type Valve		Kugelventil (ISO X8071).
Butterfly Type Form1 Valve		Absperrklappe (Form 1) (ISO X8075).
Diaphragm Valve		Diaphragm Valve
Membranventil		Regelventil, kontinuierlich betrieben (ISO X8087).

Control	Icon	Beschreibung
Four Way Valve		4-Wegeventil (allgemein).
Gate Type Valve		Absperrschieber (ISO X8074).
General Valve		Ventil (allgemein) (ISO 2101).
Globe Type Safety Valve		Sicherheits-Durchgangsventil, federbelastet (ISO X2124).
Globe Type Valve		Durchgangsventil (ISO X8068).
Needle Type Valve		Nadelventil (ISO X8076).
Three Way Ball Type Valve		3-Wege-Kugelventil (ISO X8073).
Three Way Globe Type Valve		3-Wege-Durchgangsventil (ISO X8070).
Three Way Type Valve		3-Wegeventil (allgemein) (ISO 2103).

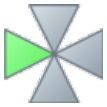
Das baseValve-Element fügt im Vergleich zu ProcessObject keine weiteren Eigenschaften, Funktionen oder Ereignisse hinzu. Die baseThreeWayTypeValve- und baseFourWayValve-Elemente fügen die **Valve Position**-Eigenschaft in der Common-Kategorie hinzu. So erben die Basis-Controls für 3- und 4-Wegeventile dieses Verhalten von ihrem Basiselement, aber dennoch hat die **Graphic Status**-Eigenschaft Vorrang vor der **Valve Position**-Eigenschaft. Die **Valve Position**-Eigenschaft wird für 3- und 4-Wegeventile unterschiedlich gehandhabt.

Die folgende Tabelle zeigt die Übereinstimmung zwischen der **Valve Position**-Eigenschaft und dem Bild des 3-Wegeventil-Controls:

Valve Position Wert	0	1	2	3	4	5	6	7	Jeder andere
Bild									

Die folgende Tabelle zeigt die Übereinstimmung zwischen der **Valve Position**-Eigenschaft und dem Bild des 4-Wegeventil-Controls:

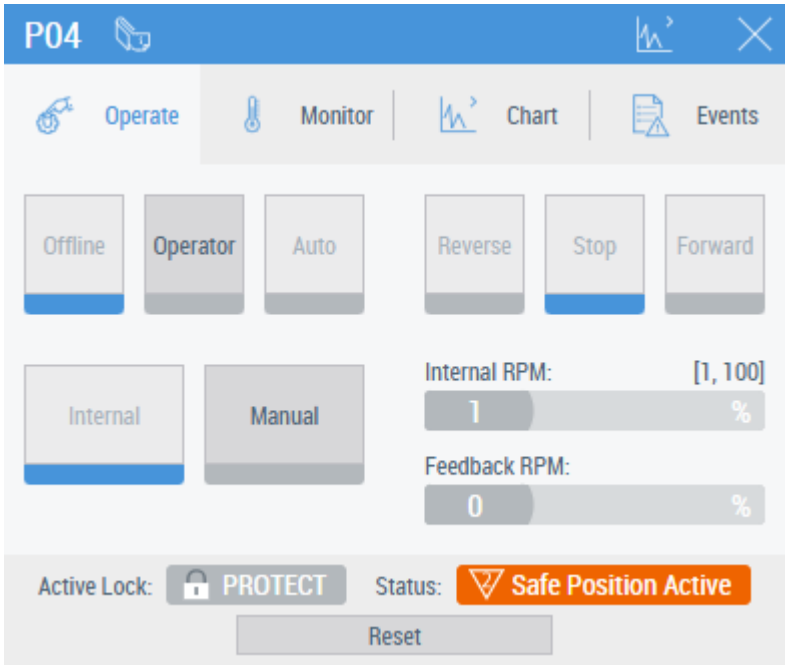
Valve Position Wert	0	1	2	3	4	5	6	7
Bild								

Valve Position Wert	8	9	10	11	12	13	14	15
Bild								

Valve Position Wert	16	17	18	19	20	21	Jeder andere	
Bild								

4.4 Fertige Faceplate-Elemente

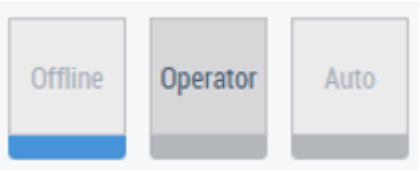
Dies ist ein Beispiel für ein fertiges Faceplate

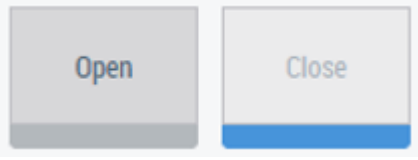
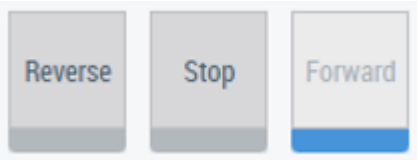
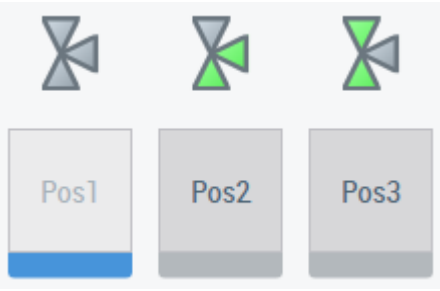
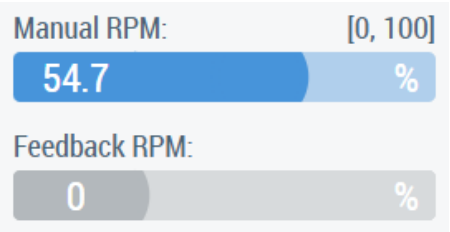


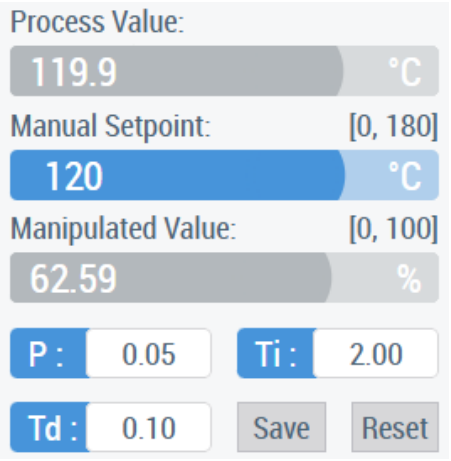
4.4.1 Operate-Registerkarte

Die Menge der Elemente auf der **Operate**-Registerkarte hängt vom MTP-Typ des Symbols ab, das an die **Data Symbol**-Eigenschaft gebunden ist.

Die **Operate**-Registerkarte des fertigen Faceplates enthält sich wiederholende Gruppen von Elementen für verschiedene MTP-Typen. Unter den Schaltflächen befindet sich ein Indikator, der den entsprechenden Akt- oder Fbk-Status anzeigt (grau = 0, blau = 1). In der folgenden Tabelle werden die Elemente beschrieben.

#	Gruppe von Elementen (Beispiel)	Beschreibung
1		**State**-Gruppe (gemäß der MTP-Spezifikation Teil 3). Siehe State-Modi [▶ 38]. Der aktuelle Status der DataAssembly ist OFFLINE. Die **Operator**-Schaltfläche wird als freigegeben angezeigt, andere Tasten sind deaktiviert. Der Grund dafür ist, dass nach der MTP-Spezifikation Teil 3 folgende Transitionen möglich sind: Offline → Operator Operator → Offline

#	Gruppe von Elementen (Beispiel)	Beschreibung
		Operator → Automatik Automatik → Operator
2		**Source**-Gruppe (gemäß MTP-Spezifikation Teil 3). Siehe Source-Modi [▶ 38]. Die aktuelle Quelle der DataAssembly ist MANUAL. Die **Internal**-Schaltfläche ist freigegeben. Die **Manual**-Schaltfläche ist deaktiviert, da die aktuelle Quelle MANUAL ist.
3		Öffnet/schließt die **Switching**-Gruppe (kann ein-/ausgeschaltet werden, etc.). Der aktuelle Status ist GESCHLOSSEN. Die **Open**-Schaltfläche ist freigegeben. Die **Close**-Schaltfläche ist deaktiviert, da das Ventil bereits geschlossen ist. Ob die Gruppe aktiv ist oder nicht, hängt vom aktuellen Status der **State**-Gruppe (#1 in dieser Tabelle) ab: Wenn die **State**-Gruppe nicht vorhanden ist: immer aktiv. Wenn die **State**-Gruppe vorhanden ist: nur aktiv, wenn der Status OPERATOR ist.
4		Rückwärts/Stop/Vorwärts **Switching**-Gruppe. Der aktuelle Status ist FORWARD. Die **Forward**-Schaltfläche ist deaktiviert, die **Stop**- und **Reverse**-Schaltflächen sind freigegeben. Je nach FwdEn/RevEn können die entsprechenden Schaltflächen ausgeblendet werden. Die Gruppe ist nur aktiv, wenn der aktuelle Status der **State**-Gruppe (#1 in dieser Tabelle) OPERATOR ist.
5		Pos1/Pos2/Pos3 **Switching**-Gruppe für TriPos-Ventile. Die grafischen Elemente oben zeigen die Teile an, die für jede Position aktiv sein sollen. Der aktuelle Status ist POSITION 1. Die **Pos1**-Schaltfläche ist deaktiviert, die **Pos2**- und **Pos3**-Schaltflächen sind freigegeben. Die Gruppe ist nur aktiv, wenn der aktuelle Status der **State**-Gruppe (#1 in dieser Tabelle) OPERATOR ist.
6		Die **Value**-Gruppe besteht aus einem Control, mit dem ein Wert manuell eingestellt werden kann, und einem Control, das eine Rückmeldung über den aktuellen Istwert gibt. Der Name, der Maßstab und die technischen Einheiten des Wertes können unterschiedlich sein. Ob die Gruppe den Wert setzen darf oder nicht, hängt vom aktuellen Status der **Source**-Gruppe (#2 in dieser Tabelle) ab: Wenn die **Source**-Gruppe nicht vorhanden ist: immer erlaubt. Wenn die **Source**-Gruppe vorhanden ist: nur aktiv, wenn der Status MANUAL ist. Wenn der Status der **Source**-Gruppe INTERNAL ist, zeigt die Einstellwertkontrolle nur den Wert an, der intern vorgegeben ist.

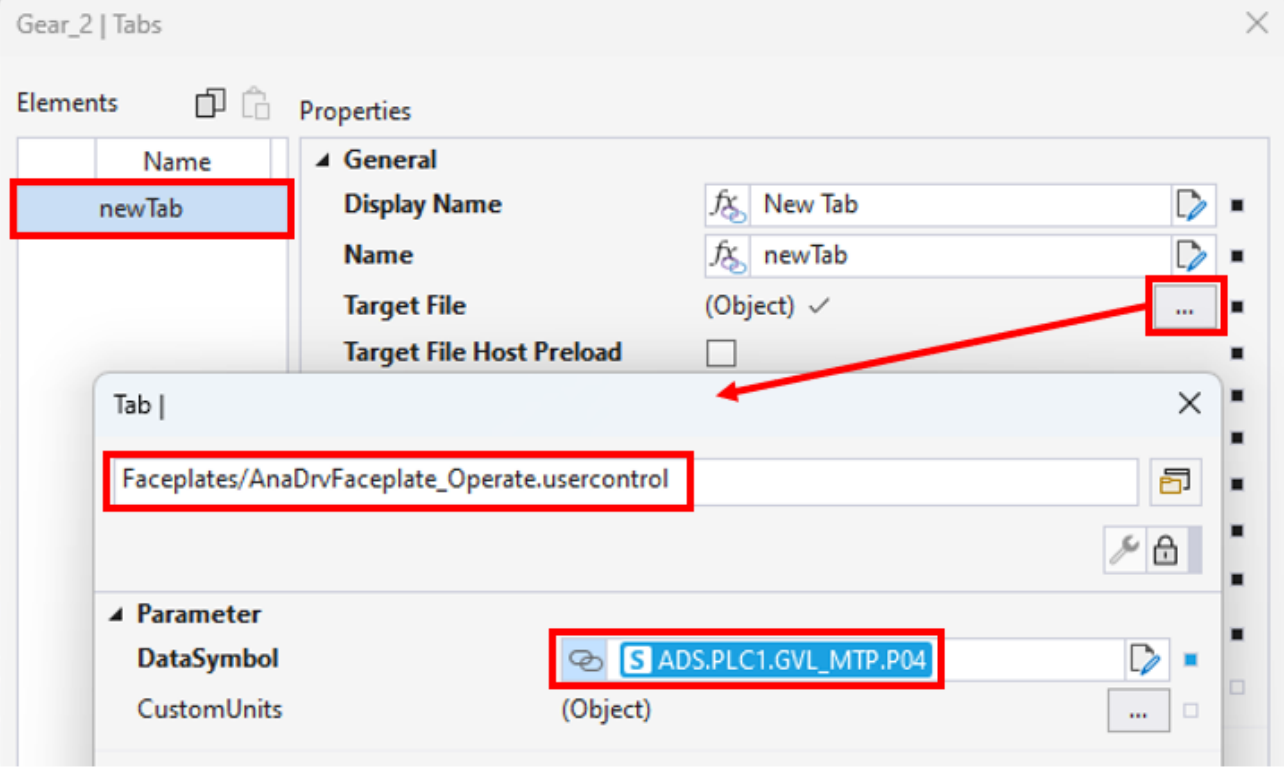
#	Gruppe von Elementen (Beispiel)	Beschreibung
7		Die **PID Controller**-Gruppe besteht aus folgenden Elementen: • **Process Value** ist der aktuelle Messwert eines bestimmten Teils eines Prozesses. Er ist immer schreibgeschützt. • **Setpoint** stellt den gewünschten Wert für den **Process Value** ein. Das Verhalten des **Setpoints** in Abhängigkeit von den Zuständen der **State** und **Source**-Gruppen ist in einer Tabelle [▶ 53] beschrieben. • **Manipulated Value** ist ein Ausgang des PID-Reglers, der den **Process Value** direkt beeinflusst. Das Verhalten des **Manipulated Value** in Abhängigkeit von den Zuständen der **State** und **Source**-Gruppen ist in einer Tabelle [▶ 53] beschrieben. • **P**, **Ti** und **Td** sind proportionale, integrale bzw. abgeleitete Terme. • Die **Save**-Schaltfläche speichert **P**, **Ti** und **Td** im SPS-Programm (nicht im persistenten Speicher) und im Speicher des Webclients, der beim Neuladen der Seite zurückgesetzt wird. Die **Reset**-Schaltfläche gibt die vorherigen Werte für **P**, **Ti** und **Td** wieder, die im Speicher des Webclients gespeichert sind.

In der folgenden Tabelle wird das Verhalten der Elemente der **PID Controller**-Gruppe beschrieben

• Status der **State**-Gruppe, #1 in der Tabelle [▶ 51]	• Status der **Source**-Gruppe, #2 in der Tabelle [▶ 51]	**Setpoint** Verhalten	**Manipulated Value** Verhalten
OFFLINE	INTERNAL	Nur lesen	Veränderbar, ohne wirksam zu werden
OFFLINE	MANUAL	Veränderbar, ohne wirksam zu werden	Veränderbar, ohne wirksam zu werden
OPERATOR	INTERNAL	Nur lesen	Veränderbar, wirksam auf **Manipulated Value**
OPERATOR	MANUAL	Veränderbar, ohne wirksam zu werden	Veränderbar, wirksam auf **Manipulated Value**
AUTOMATIC	INTERNAL	Nur lesen, wirksam auf **Manipulated Value**	Nur lesen, berechnet auf Basis des **Setpoints**
AUTOMATIC	MANUAL	Veränderbar, wirksam auf **Manipulated Value**	Nur lesen, berechnet auf Basis des **Setpoints**

Wiederverwendung der Operate-Registerkarte mit der Registerkarten-Eigenschaft

Fertige Operate-Faceplates können in der Registerkarten-Eigenschaft [▶ 32] ohne Änderungen wiederverwendet werden, wenn sie mit einer FB-Instanz arbeiten, die den gleichen Satz von Variablen (gleiche Namen und gleiche Typen) besitzt wie die für den **DataSymbol**-Parameter des Faceplates definierte. Nachfolgend finden Sie ein Beispiel für die Einstellungen des AnaDrv MTP Typ **DataSymbol**-Parameters für *AnaDrvFaceplate_Operate.usercontrol*:



Sehen Sie dazu auch

- Kategorie: Label [▶ 38]
- Kategorie: Label [▶ 38]

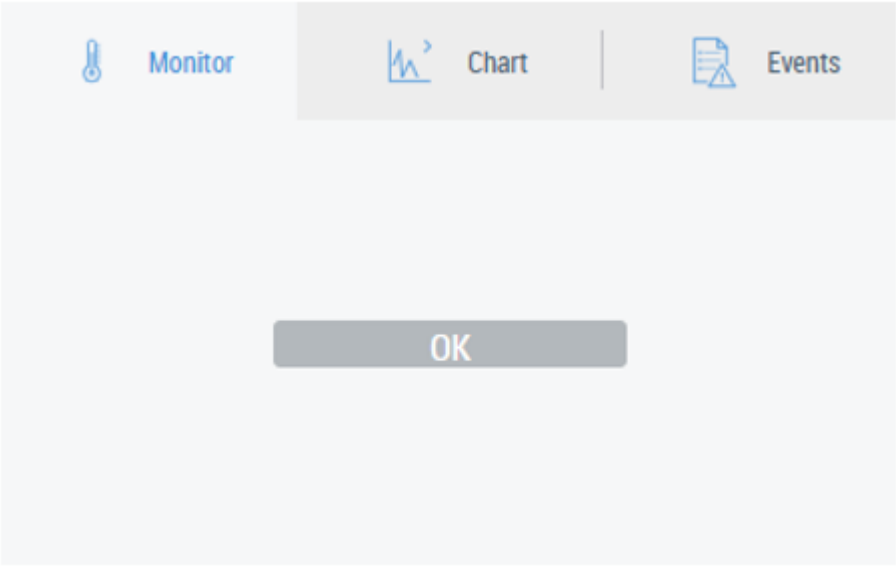
4.4.2 Monitor-Registerkarte

Je nach MTP-Typ des Symbols, das an die **Data Symbol**-Eigenschaft gebunden ist, kann die **Monitor**-Registerkarte automatisch auf dem Faceplate angezeigt werden. Seine Ansicht ist je nach MTP-Typ mit unterschiedlichen UserControls implementiert.

Monitor-Registerkarten können mit der Registerkarten-Eigenschaft [▶ 32] auf die gleiche Weise wiederverwendet werden wie die Operate-Registerkarten [▶ 53].

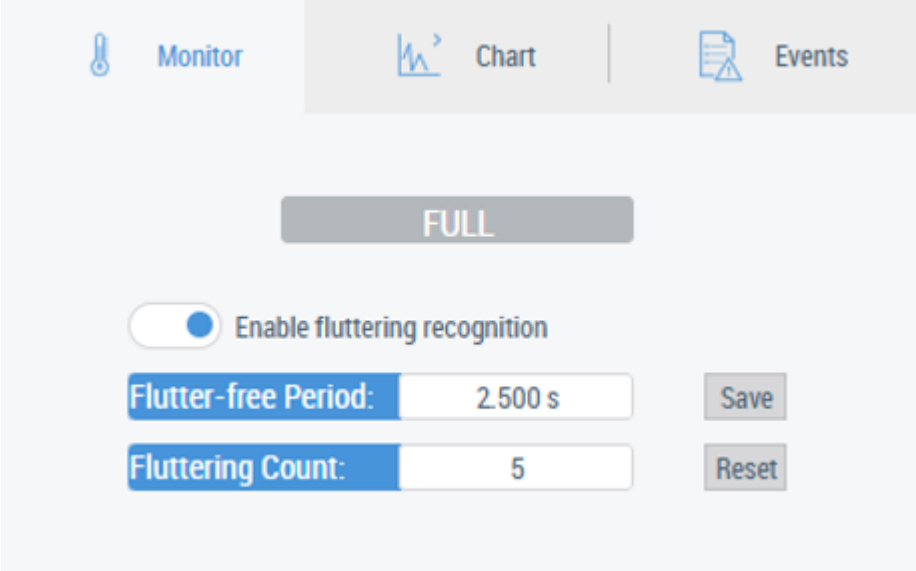
4.4.2.1 BinViewFaceplate

BinViewFaceplate zeigt nur den aktuellen Wert gemäß VState0 und VState1 an:



4.4.2.2 BinMonFaceplate

Ein Beispiel für BinMonFaceplate:

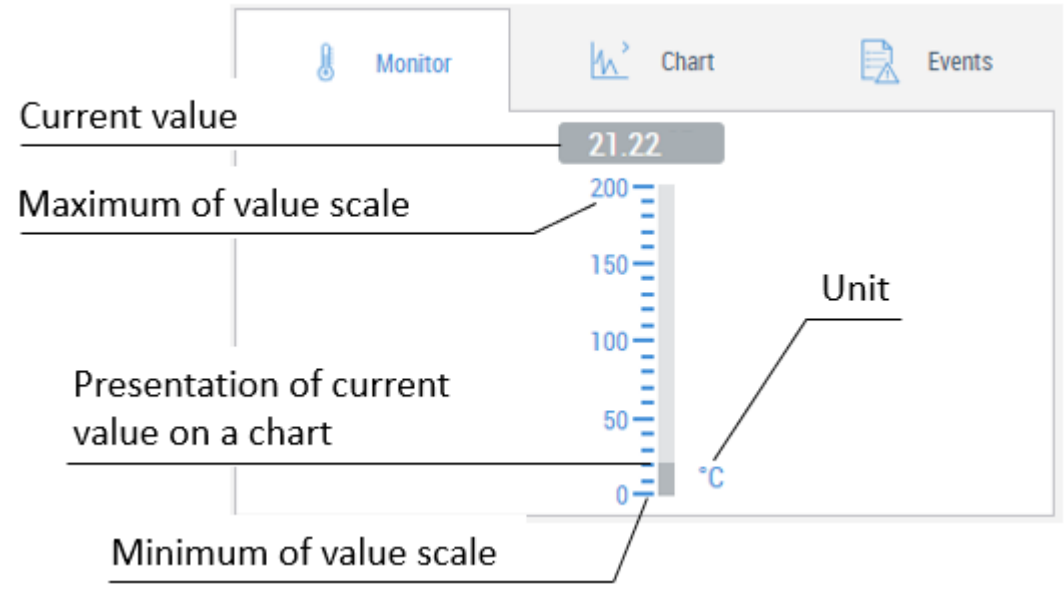


Eine Beschreibung der Elemente finden Sie in der folgenden Tabelle.

#	Element	Beschreibung
1	Value	Verarbeiteter Wert.
2	Enable fluttering recognition	Schaltet das Erkennen des Signalflatterns ein/aus.
3	Flutter-free Period	Dauer eines aktiven Signals, bevor es als flatterfrei erkannt werden kann.
4	Fluttering Count	Anzahl der zulässigen Flattersignale im definierten Zeitraum **Flutter-free Period** (Nr. 3 in dieser Tabelle).
5	Save button	Die **Save**-Schaltfläche speichert die **Flutter-free Period** und den **Fluttering Count** im SPS-Programm (nicht im persistenten Speicher) und im Speicher des Web-Clients, der beim Neuladen der Seite zurückgesetzt wird.
6	Reset button	Die **Reset**-Schaltfläche gibt die vorherigen Werte für die **Flutter-free Period** und den **Fluttering Count** zurück, die im Speicher des Web-Clients gespeichert sind.

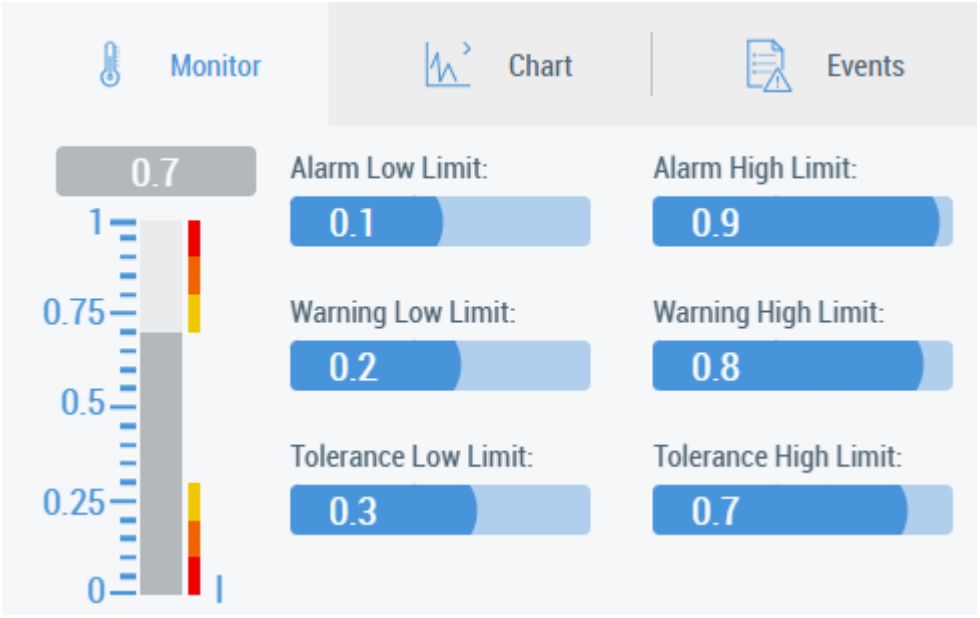
4.4.2.3 AnaViewFaceplate und DIntViewFaceplate

AnaViewFaceplate und DIntViewFaceplate sehen gleich aus, aber DInt zeigt den Integer-Wert an.

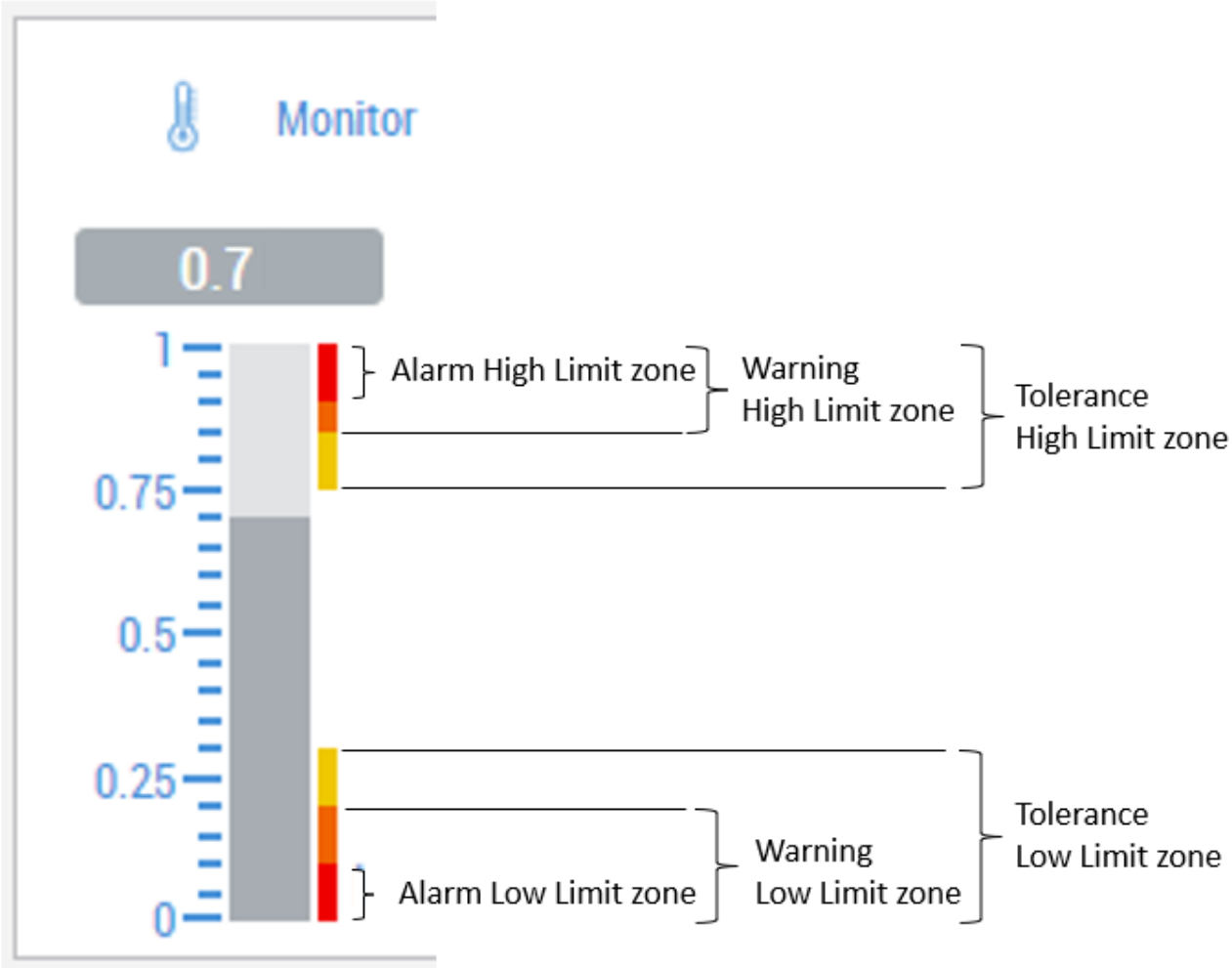


4.4.2.4 AnaMonFaceplate und DIntMonFaceplate

AnaMonFaceplate und DIntMonFaceplate sehen gleich aus, aber DInt zeigt Integer-Werte an und setzt Grenzwerte, die ebenfalls Integer-Werte sind.



Elements für Alarm-, Warn- und Toleranzgrenzen dienen dazu, die Grenzen des Wertes anzuzeigen und zu verändern. Die nachstehende Abbildung zeigt die Zonen auf dem Diagramm, die den Grenzwerteinstellungen entsprechen. Andere Elemente des Diagramms werden für AnaViewFaceplate und DIntViewFaceplate [▶ 55] erklärt.



4.4.2.5 MonBinDrvFaceplate und MonBinVlvFaceplate

MonBinDrvFaceplate und MonBinVlvFaceplate haben den gleichen Inhalt. Ein Beispiel für das Faceplate ist unten abgebildet.

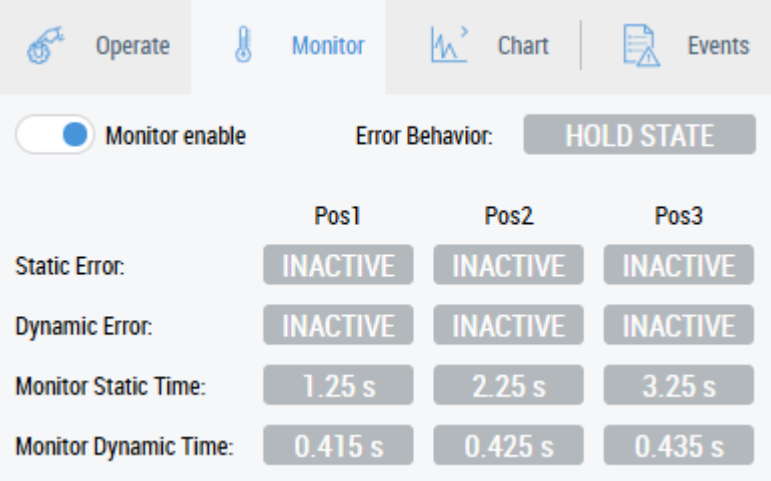
Die folgende Tabelle zeigt die Beschreibung der MonBinDrvFaceplate- und MonBinVlvFaceplate-Elemente:

#	Element	Beschreibung
1	Monitor enable	Monitoring aktivieren
2	Static Error	Anzeige: statischer Fehler
3	Dynamic Error	Anzeige: dynamischer Fehler
4	Monitor Static Time	Zeitparameter (MonStatTi) für den statischen Fehler

#	Element	Beschreibung
5	Monitor Dynamic Time	Zeitparameter (MonDynTi) für den dynamischen Fehler
6	Error Behavior	Verhalten im Falle eines Fehlers (Status halten -HOLD STATE- oder zu SafePos gehen)

4.4.2.6 MonTriPosVlvFaceplate

Eine Ansicht der MonTriPosVlvFaceplate ist unten dargestellt.

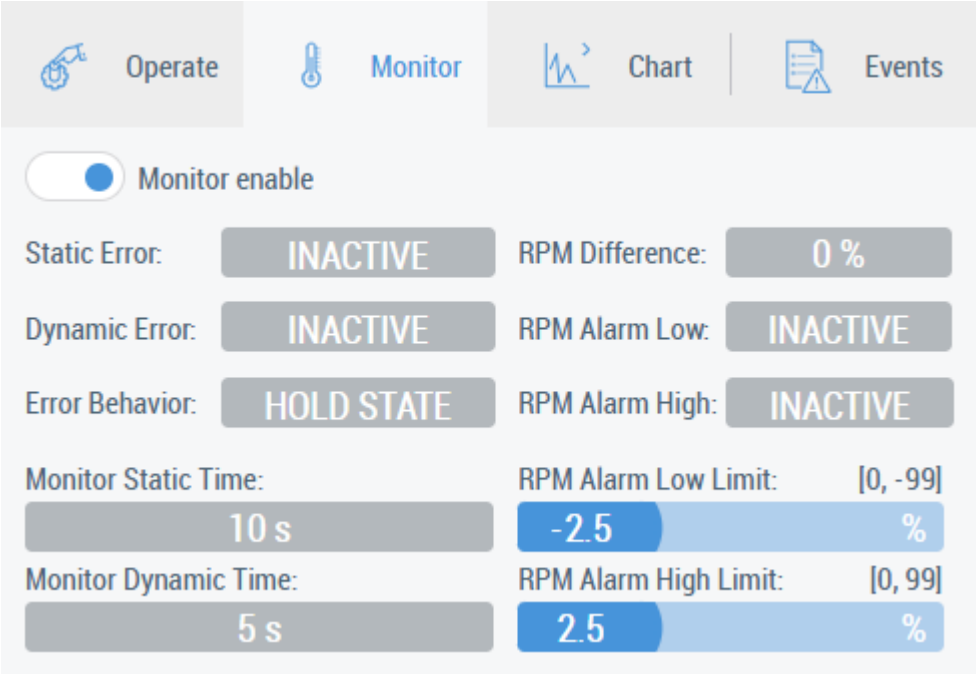


Die folgende Tabelle zeigt die Beschreibung der MonTriPosVlvFaceplate-Elemente:

#	Element	Beschreibung
1	Monitor enable	Monitoring aktivieren
2	Error Behavior	Verhalten im Falle eines Fehlers (Status halten -HOLD STATE- oder zu SafePos gehen)
3	Static Error	Anzeige: statischer Fehler für Pos1...3
4	Dynamic Error	Anzeige: dynamischer Fehler für Pos1...3
5	Monitor Static Time	Zeitparameter (MonStatTi) für Pos1...3 statischer Fehler
6	Monitor Dynamic Time	Zeitparameter (MonDynTi) für Pos1...3 dynamischer Fehler

4.4.2.7 MonAnaDrvFaceplate

Ein Beispiel für MonAnaDrvFaceplate ist unten abgebildet.

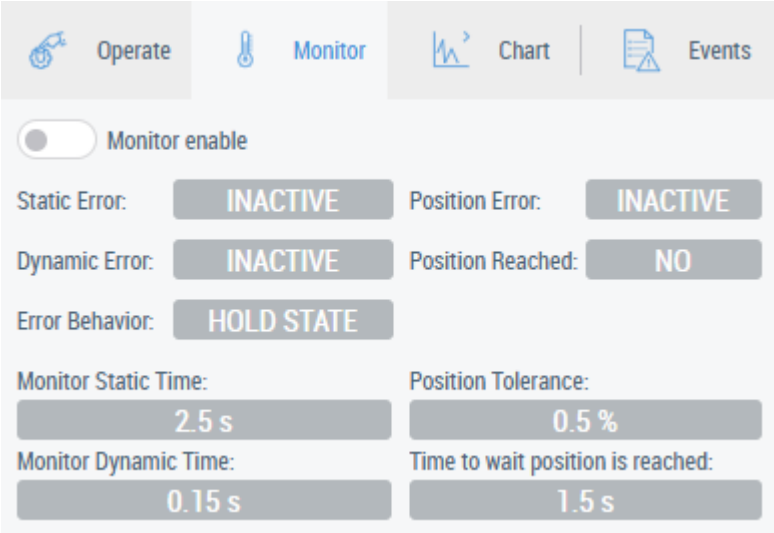


Die folgende Tabelle zeigt die Beschreibung der MonAnaDrvFaceplate-Elemente:

#	Element	Beschreibung
1	Monitor enable	Monitoring aktivieren
2	Static Error	Anzeige: statischer Fehler
3	Dynamic Error	Anzeige: dynamischer Fehler
4	Error Behavior	Verhalten im Falle eines Fehlers (Status halten -HOLD STATE- oder zu SafePos gehen)
5	RPM Difference	Differenz zwischen eingestellter und aktueller Drehzahl (Drehzahlabweichung)
6	RPM Alarm Low	Anzeige für Drehzahlabweichung, die den unteren Grenzwert überschreitet (#11 in dieser Tabelle)
7	RPM Alarm High	Anzeige für Drehzahlabweichung, die den oberen Grenzwert überschreitet (#10 in dieser Tabelle)
8	Monitor Static Time	Zeitparameter (MonStatTi) für den statischen Fehler
9	Monitor Dynamic Time	Zeitparameter (MonDynTi) für den dynamischen Fehler
10	RPM Alarm Low Limit	Unterer Grenzwert für die Drehzahlabweichung **RPM Difference** (#5 in dieser Tabelle)
11	RPM Alarm High Limit	Oberer Grenzwert für die Drehzahlabweichung **RPM Difference** (#5 in dieser Tabelle)

4.4.2.8 MonAnaVlvFaceplate

Ein Beispiel für MonAnaVlvFaceplate ist unten abgebildet.

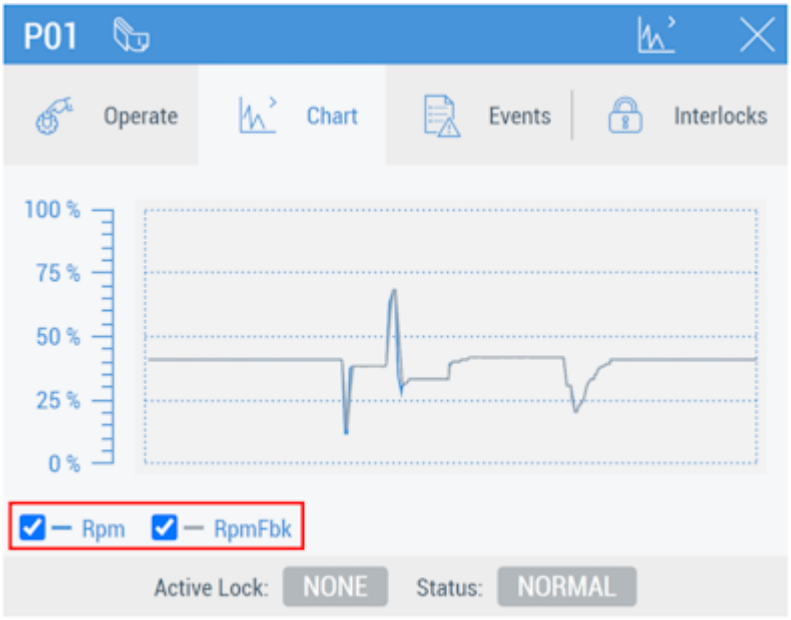


Die folgende Tabelle zeigt die Beschreibung der **MonAnaVlvFaceplate**-Elemente:

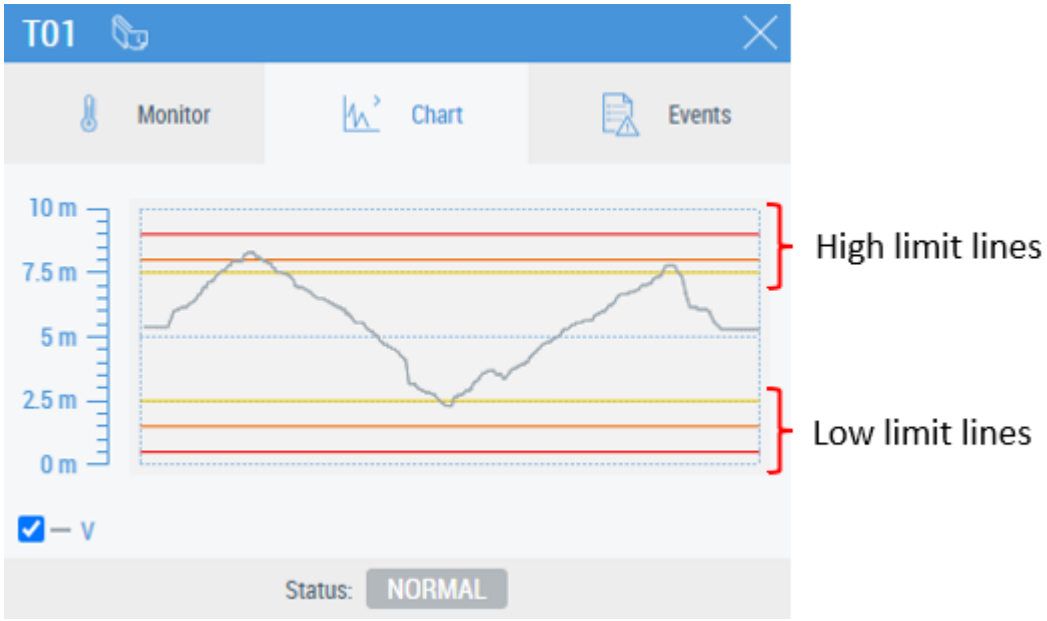
#	Element	Beschreibung
1	Monitor enable	Monitoring aktivieren
2	Static Error	Anzeige: statischer Fehler
3	Dynamic Error	Anzeige: dynamischer Fehler
4	Error Behavior	Verhalten im Falle eines Fehlers (Status halten -HOLD STATE- oder zu SafePos gehen)
5	Position Error	Anzeige, dass die maximale Fahrzeit (#10 in dieser Tabelle) überschritten wurde
6	Position Reached	Anzeige, dass die Zielposition mit der eingestellten **Position Tolerance** (#9 in dieser Tabelle) erreicht wurde
7	Monitor Static Time	Zeitparameter (MonStatTi) für den statischen Fehler
8	Monitor Dynamic Time	Zeitparameter (MonDynTi) für den dynamischen Fehler
9	Position Tolerance	Positionstoleranz-Parameter zur Feststellung, dass die Sollwert-Position erreicht ist
10	Time to wait position is reached	Parameter zur Einstellung der maximalen Fahrzeit (MonPosTi)

4.4.3 Chart-Registerkarte

Der Satz von Diagrammen und Grenzwertlinien auf der Charts-Registerkarte hängt von der Art der DataAssembly ab, die der MTP PLC Library FB präsentiert. Die Diagramme können mit den entsprechenden Controls ausgeblendet werden:



Ein Beispiel für Grenzwertlinien:



Parameter

Die folgende Tabelle zeigt die Beschreibung der *Faceplate_Chart.usercontrol*-Parameter:

Parameter	Beschreibung	Typ	Beispiel
DataSymbolPath1	SymbolPath zu einer Variablen 1	String	ADS.PLC1.GVL_MTP.H01.V
DataSymbolPath2	SymbolPath zu einer Variablen 2	String	ADS.PLC1.GVL_MTP.H01.VFbk
DataSymbolPath3	SymbolPath zu einer Variablen 3	String	ADS.PLC1.GVL_MTP.H01.VInt
SclMin	Mindestwert der Chartskala	REAL	%s%ADS.PLC1.GVL_MTP.V03::PosSclMin%/s%, der beim Binden wie folgt aussieht: 

Parameter	Beschreibung	Typ	Beispiel
SclMax	Höchstwert der Chartskala	REAL	%s%ADS.PLC1.GVL_MTP.V03::PosSclMax%/s%, der beim Binden wie folgt aussieht: 
Unit	Einheiten	Nummer	%s%ADS.PLC1.GVL_MTP.V03::PosUnit%/s%, der beim Binden wie folgt aussieht: 
HHHEn	Die Linie für HiHiHi Limit anzeigen	BOOL	
HHH	Linie für HiHiHi Limit	REAL	90 oder 
HHEn	Die Linie für HiHi Limit anzeigen	BOOL	
HH	Linie für HiHi Limit	REAL	Siehe Beispiele für HHH
HEn	Die Linie für Hi Limit anzeigen	BOOL	
H	Linie für Hi Limit	REAL	Siehe Beispiele für HHH
LEn	Die Linie für Low Limit anzeigen	BOOL	
L	Linie für Low Limit	REAL	Siehe Beispiele für HHH
LLEn	Die Linie für LowLow Limit anzeigen	BOOL	
LL	Linie für LowLow Limit	REAL	Siehe Beispiele für HHH
LLLEn	Die Linie für LowLowLow Limit anzeigen	BOOL	
LLL	Linie für LowLowLow Limit	REAL	Siehe Beispiele für HHH

Variablen, die in DataSymbolPath[N] (N ist eine Zahl, z. B. 1, 2, 3) gesetzt sind, werden beim Konfigurieren des Faceplates im TwinCAT HMI Engineering automatisch historisiert. Dasselbe geschieht bei der Verwendung von SPS-Attributen, aber der Variable sollte das **Chart**-Attribut vorangestellt werden (siehe Beispiel für die Konfiguration der Chart-Registerkarte mit SPS-Attributen [▶ 21]).

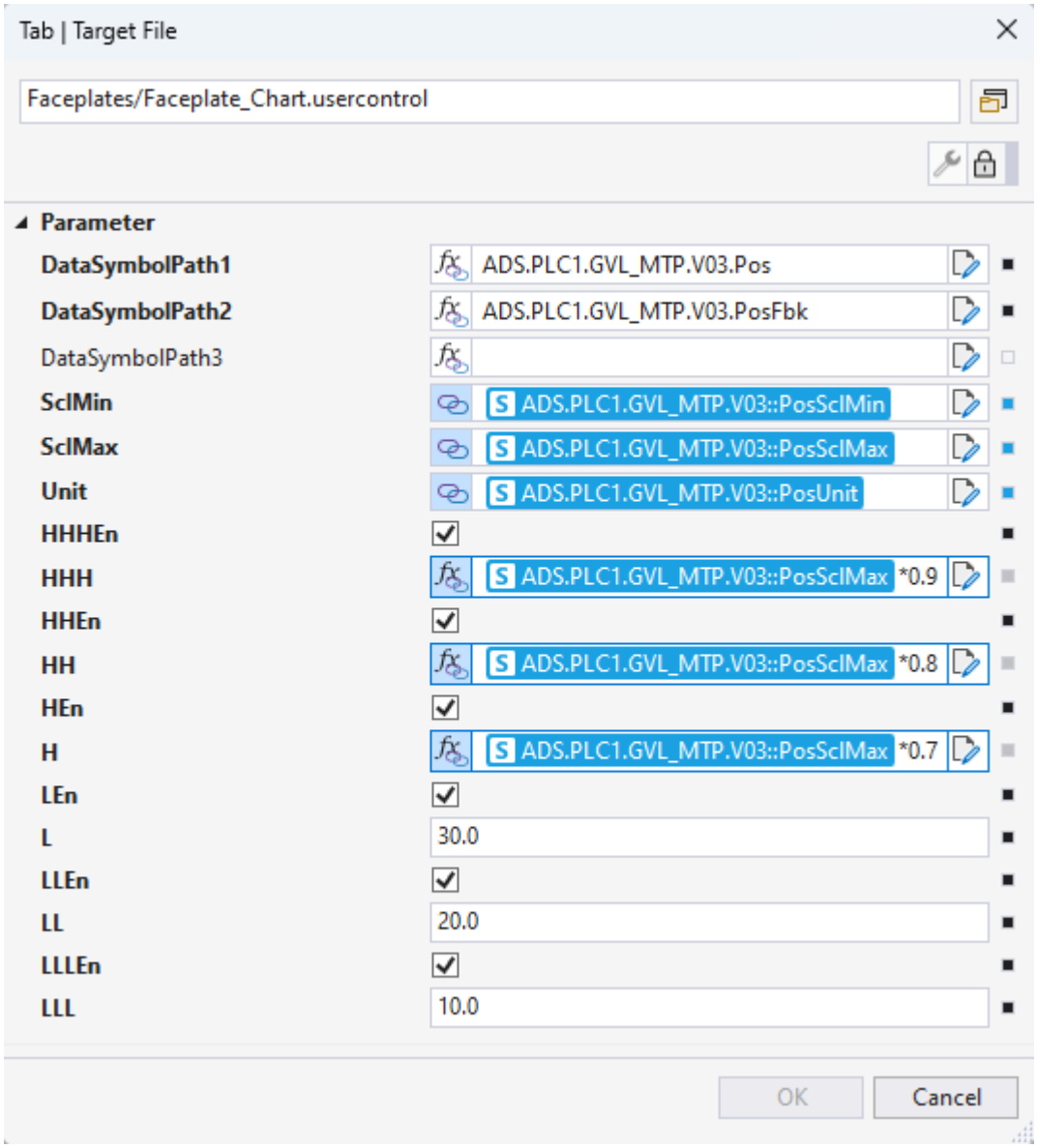
Wiederverwendung der Chart-Registerkarte mit der Registerkarten-Eigenschaft

Faceplate_Chart.usercontrol kann als angepasste Registerkarte [▶ 32] für ein Faceplate verwendet werden. Es können maximal drei Variablen ausgewählt werden, um Diagramme zu zeichnen.

Symbole als historisiert festlegen

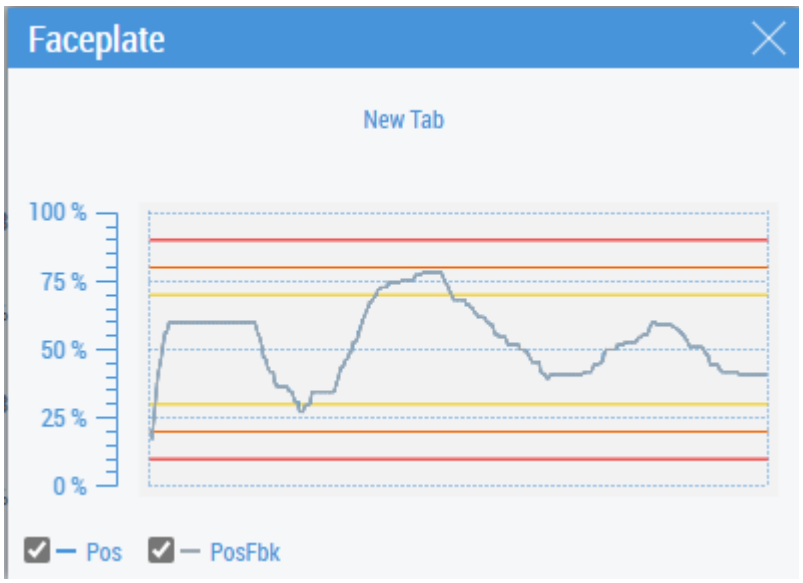
Um das Diagramm anzuzeigen, sollten die Symbole, die an die **DataSymbolPath1 ... DataSymbolPath3**-Parameter gebunden sind, als **Historized** eingestellt werden.

Faceplates/Faceplate_Chart.usercontrol sollte für die **Target File**-Eigenschaft im Fenster zur Registerkarten-Konfiguration gewählt werden. Ein Beispiel für eine Parameterkonfiguration ist unten dargestellt:



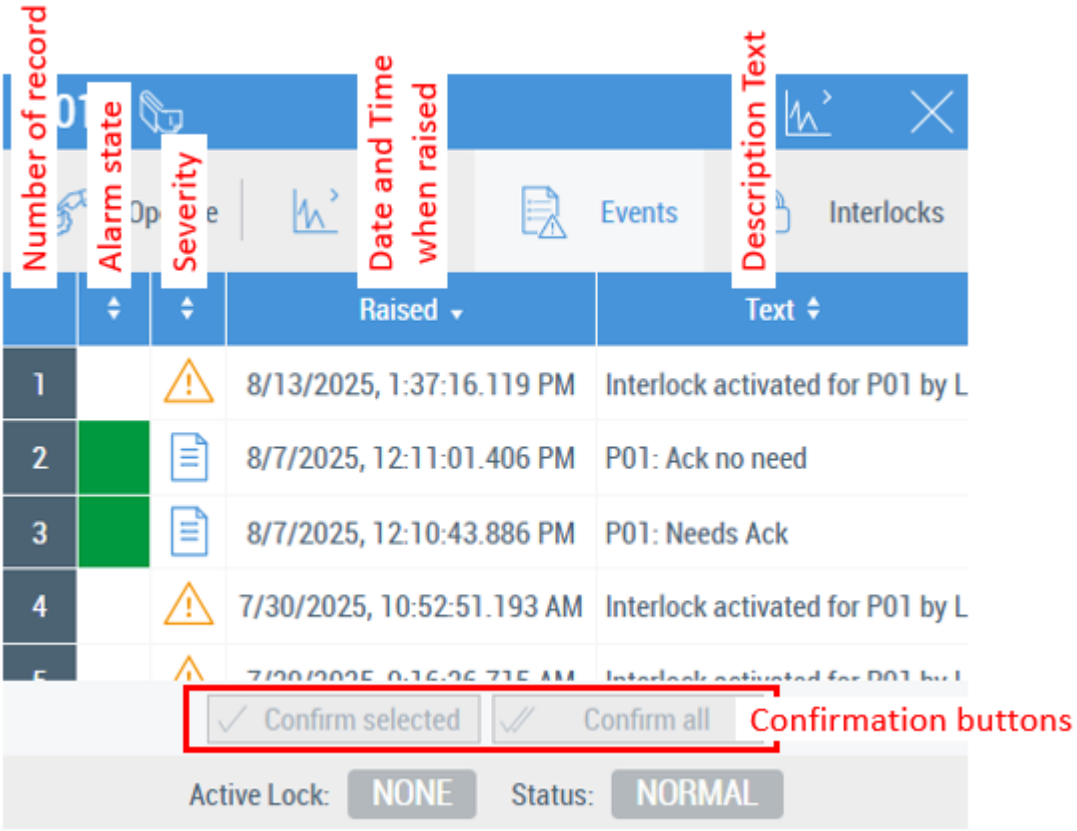
- **DataSymbolPath1**... **DataSymbolPath3** sollte als Pfad zur SPS-Variable eingegeben werden (manuell, nicht aus der Dropdown-Liste ausgewählt).
- **SclMin, SclMax, Unit** und Grenzwerte sollten als Symbole verknüpft oder als JavaScript-Ausdrücke eingegeben werden.

Das Ergebnis der Konfiguration ist unten dargestellt:



4.4.4 Events-Registerkarte

Die Menge der Ereignisse auf der Events-Registerkarte entspricht dem P&ID-Element, das das Control darstellt. Spalten, die mit einem Pfeil/Pfeilen markiert sind, können nach Reihenfolge sortiert werden. Nachstehend finden Sie ein Beispiel für die Events-Registerkarte mit einer Beschreibung der Spalten und Schaltflächen.



In dieser Tabelle werden die Bedeutungen der Farben für die Spalte **Alarm state** erläutert:

Ansicht	Name der Farbe	Ereignis-Typ	Benötigt Quittierung	Bedeutung
	Weiß	Meldung	Nein	Es gibt keine Alarmzustände.

Ansicht	Name der Farbe	Ereignis-Typ	Benötigt Quittierung	Bedeutung
	Rot	Alarm	Ja	Der Alarm ist aktiv und noch nicht bestätigt.
			Nein	Der Alarm ist aktiv.
	Orange	Alarm	Ja	Der Alarm ist aktiv und bereits bestätigt.
	Gelb/grün	Alarm	Ja	Der Alarm ist inaktiv, aber noch nicht bestätigt.
	Grün	Alarm	Ja	Der Alarm ist inaktiv und bestätigt.
			Nein	Der Alarm ist inaktiv.

Parameter

Das *Faceplate_Events.usercontrol* basiert auf dem TwinCAT HMI Event Grid-Control. Es gibt nur einen Parameter, **Filter** (Datentyp ist eventFilter), der gemäß dem TwinCAT HMI Engineering Handbuch konfiguriert werden kann.

Wiederverwendung der Events-Registerkarte mit der Registerkarten-Eigenschaft

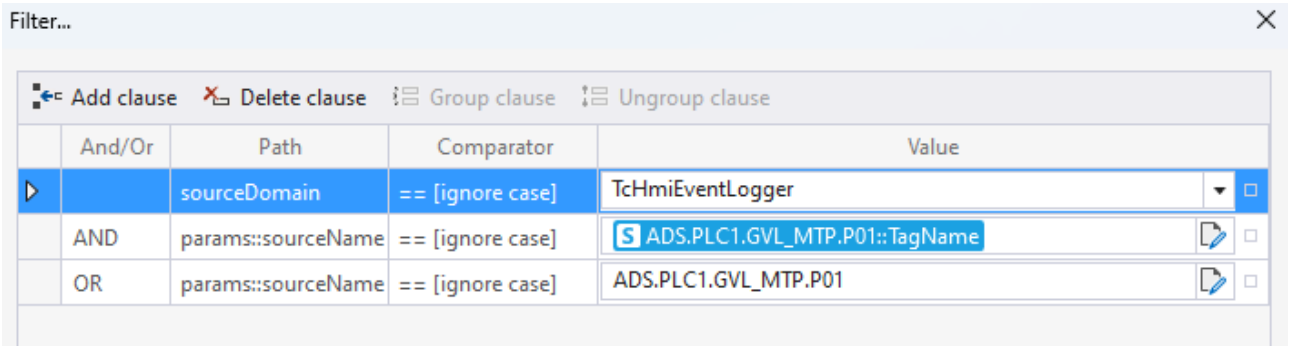
Faceplate_Events.usercontrol kann als angepasste Registerkarte [▶ 32] für ein Faceplate verwendet werden.

Erzeugen von Ereignissen

Erforderliche Ereignisse sollten im SPS-Programm generiert werden (siehe TC3 EventLogger-Handbuch).

Faceplates/Faceplate_Events.usercontrol sollte für die **Target File**-Eigenschaft im Fenster zur Registerkarten-Konfiguration ausgewählt werden. Der **Filter**-Parameter sollte entsprechend dem TwinCAT HMI Engineering Handbuch (Event Grid-Control) konfiguriert werden.

Ein Beispiel für die Konfiguration ist unten dargestellt:

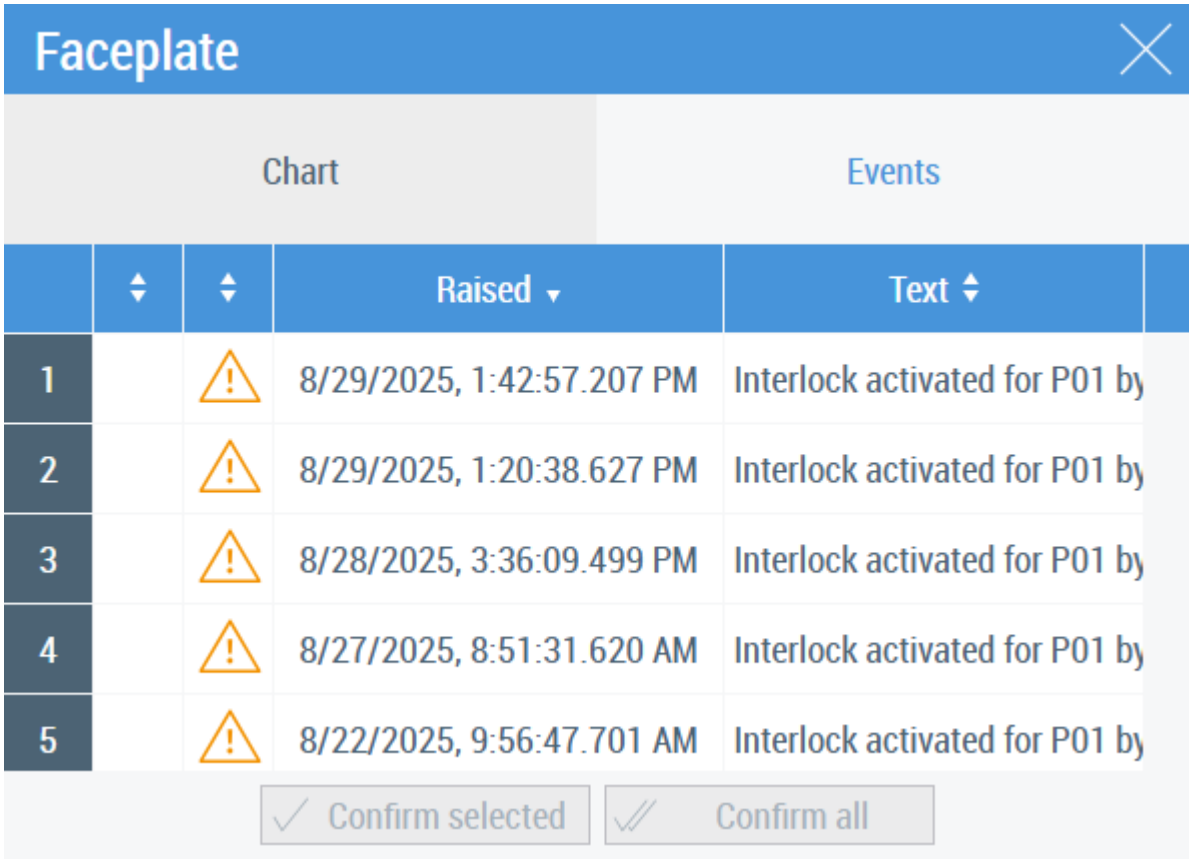


Folgende Ereignisquellen werden als angezeigt ausgewählt:

- SymbolExpression für den TagName-Wert von DataAssembly FB P01 (sollte manuell eingegeben oder von irgendwoher kopiert werden)
  `%s%ADS.PLC1.GVL_MTP.P01::TagName%/s%`
  Es kann eine beliebige andere SymbolExpression sein, die einen Namen für die Ereignisquelle angibt. Dies ist eine universelle Methode, um auf die Ereignisquelle zu verweisen.
- SymbolPath für den DataAssembly FB P01 (sollte manuell eingegeben oder von irgendwoher kopiert werden)
  `ADS.PLC1.GVL_MTP.P01`
  Es kann ein SymbolPath für einen FB sein, der eine MTP DataAssembly darstellt. Dies ist eine spezifische Methode, um auf Ereignisse der DataAssembly für die MTP PLC Library zu verweisen. Diese Methode ist derzeit auf der Seite der MTP PLC Library nicht implementiert.

Der Wert `param::sourceName` in der Spalte **Path** sollte manuell eingegeben werden.

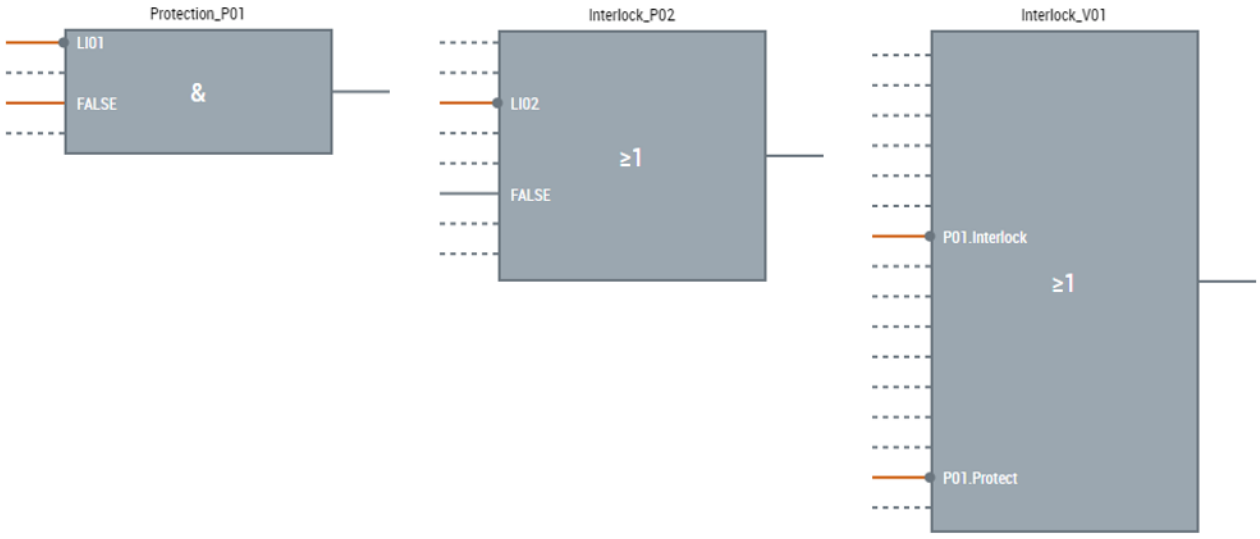
Das Ergebnis der Konfiguration ist unten dargestellt:



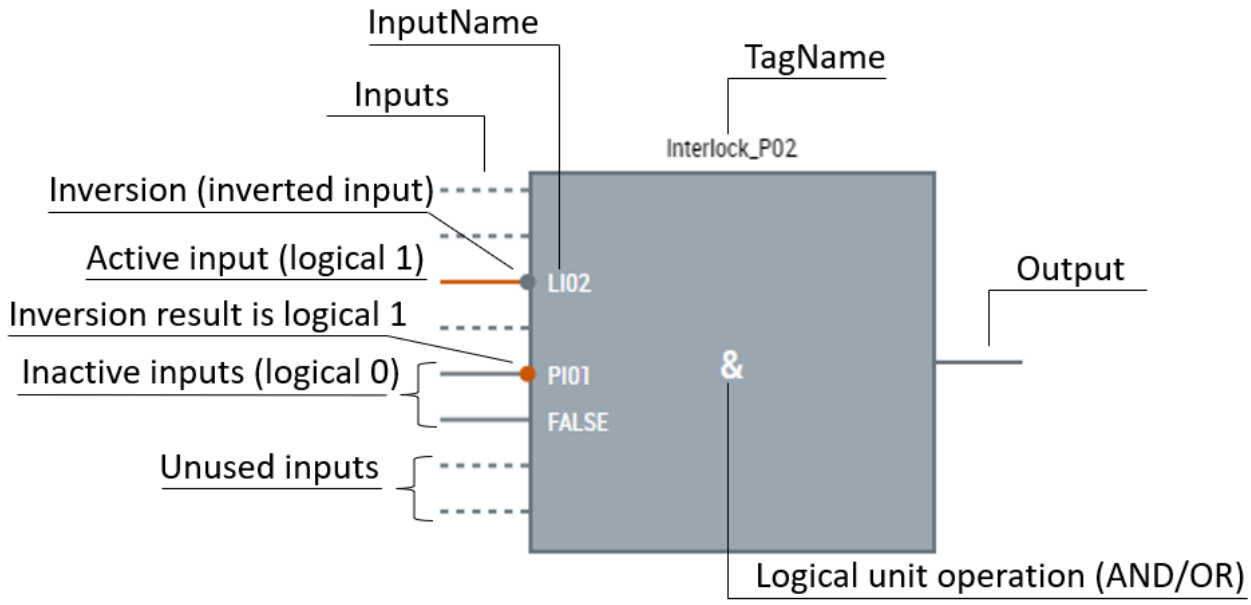
4.4.5 Interlocks Tab

Um in einem vordefinierten Faceplate zu erscheinen, sollte die Interlocks-Registerkarte mit der **Interlocks Tab**-Eigenschaft (siehe Tabelle [▶ 29]) des HMI Process Library-Controls konfiguriert werden. Die Konfiguration der Interlocks-Registerkarte wird im Kapitel Kategorie: Faceplate [▶ 31] erläutert.

Je nach Anzahl der Verriegelungsgründe kann das Control 4, 8 oder 16 Eingänge haben:



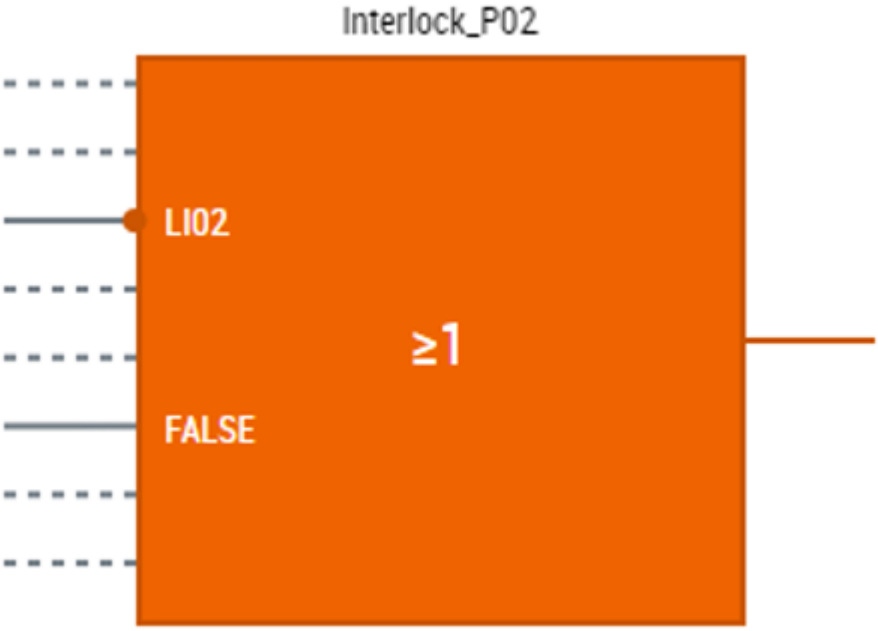
Die nachstehende Abbildung beschreibt die Elemente des Interlock-Controls:



Die Bezeichnung für die UND-Verknüpfung ist **&,** für die ODER-Verknüpfung ist **≥1**. Ist das Ergebnis der Eingabe-Invertierungsoperation

- logisch 0, wird es grau dargestellt;
- logisch 1, wird es orange dargestellt.

Die logische Verknüpfung (UND oder ODER) wird mit den Werten der Control-Eingänge durchgeführt, wobei unbenutzte Eingänge ausgeschlossen werden. Die Invertierung der Eingabe wird berücksichtigt. Ist das Ergebnis der logischen Verknüpfung 0, bleibt das Control grau. Wenn das Ergebnis 1 ist, bedeutet dies, dass die Sperre aktiv ist, und das Control und der Ausgang werden orange:

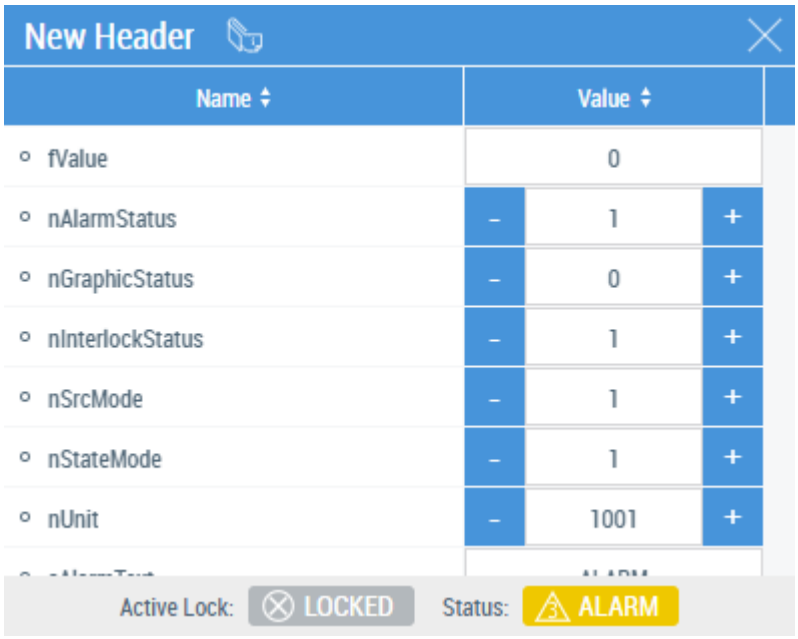


Parameter

LockView[N]_Interlocks.usercontrol (wobei N 4, 8 oder 16 sein kann) hat nur einen **DataSymbol**-Parameter, der eine Instanz des entsprechenden MTP LockView FB mit der entsprechenden Nummer der Verriegelungsgründe sein sollte.

4.4.6 ObjectBrowser-Ansicht auf einem Faceplate

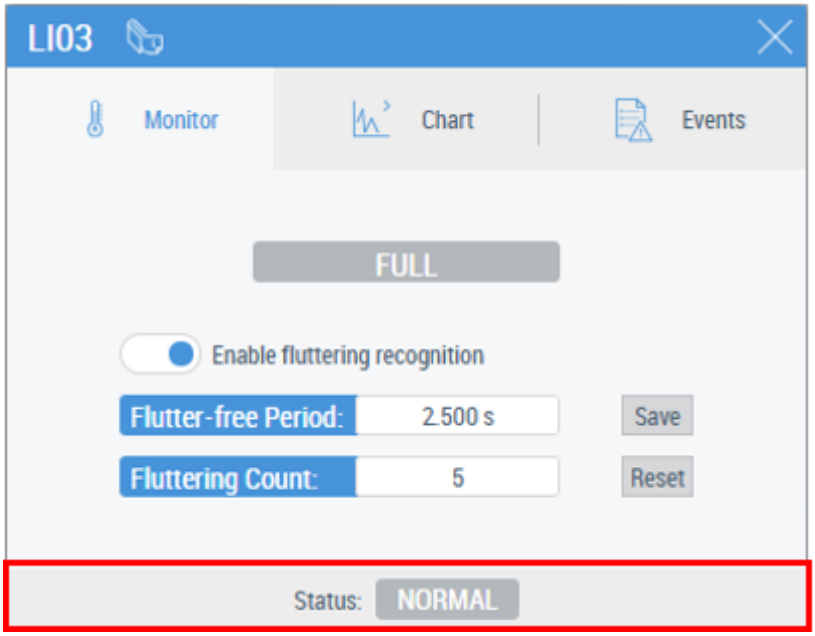
Wenn für das HMI Process Library-Control kein Faceplate definiert ist (über die **Data Symbol**-Eigenschaften MTP-Typ, **Tabs** oder **Faceplate Control**-Eigenschaften), zeigt das Faceplate-Daten an, die über die SPS-Attribut-Funktionalität definiert sind (Kapitel SPS-Attribut-Funktionalität [▶ 75]). Diese Daten können in der ObjectBrowser-Ansicht sowohl gelesen als auch geschrieben werden:



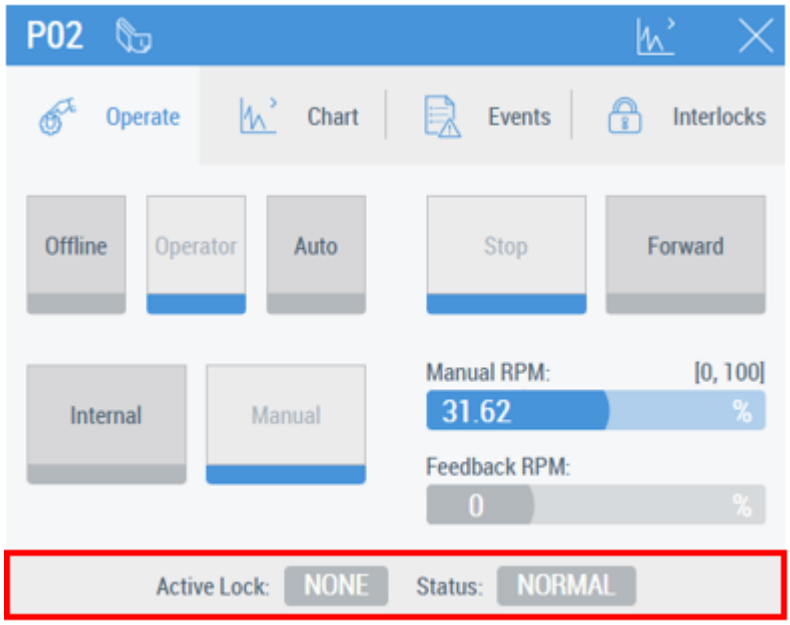
4.4.7 Fußzeile

Einige der vordefinierten Faceplates verfügen über eine Fußzeile, in der aktuelle Sperren, Stati, Warnungen und Alarme für das jeweilige P&ID-Element angezeigt werden. Welche Elemente in der Fußzeile vorhanden sind, ist abhängig vom Typ der DataAssembly, die der MTP PLC Library FB zur Verfügung stellt.

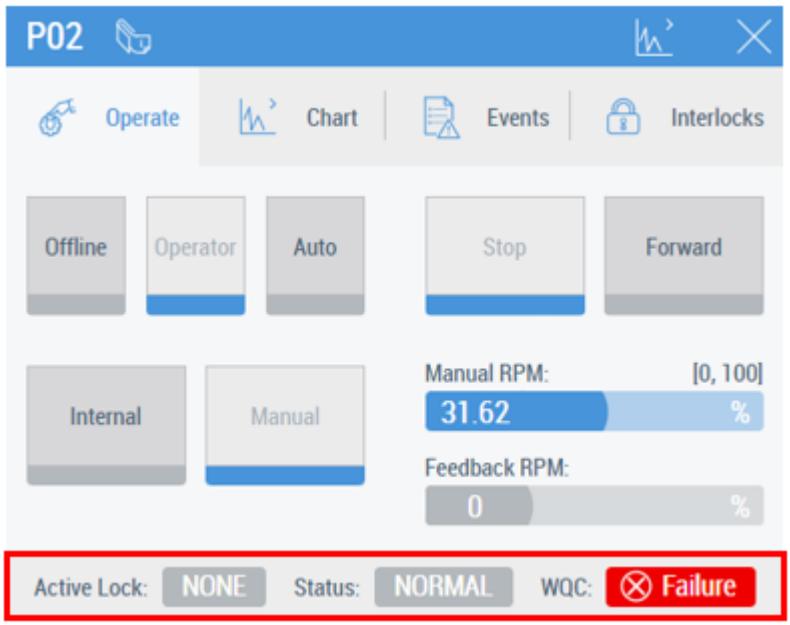
Dies ist ein Beispiel für ein Faceplate, bei dem die Fußzeile nur das **Status**-Element enthält (im "NORMAL" Status):



Dies ist ein Beispiel für ein Faceplate, bei dem die Fußzeile das **Active Lock**-Element (im "NONE" Status) und das **Status**-Element (im "NORMAL" Status) enthält:



Bei Faceplates mit Fußzeile wird das **WQC** Meldeelement zusätzlich zu anderen Meldungen angezeigt, wenn der WQC-Status einen aktiven Wert aufweist (siehe Tabelle [▶ 71]). Ein Beispiel für ein aktives **WQC**-Meldungselement:



Die folgende Tabelle zeigt die **Status**-Elementstati und enthält eine Liste der MTP-Typen, die den jeweiligen Status auf ihren vordefinierten Faceplates aufweisen können:

#	Status-Elementstatus	Beschreibung	MTP-Typen
1	NORMAL	Normaler Status	AnaMon DIntMon AnaDrv MonAnaDrv AnaVlv MonAnaVlv BinMon BinDrv MonBinDrv BinVlv

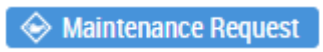
#	Status-Elementstatus	Beschreibung	MTP-Typen
			MonBinVlv TriPosVlv MonTriPosVlv
2		Alarm oben aktiv	AnaMon DIntMon
3		Alarm unten aktiv	AnaMon DIntMon
4		Warnung oben aktiv	AnaMon DIntMon
5		Warnung unten aktiv	AnaMon DIntMon
6		Toleranz oben aktiv	AnaMon DIntMon
7		Toleranz unten aktiv	AnaMon DIntMon
8		Sichere Position ist aktiviert (sicherer Betrieb)	AnaDrv MonAnaDrv AnaVlv MonAnaVlv BinDrv MonBinDrv BinVlv MonBinVlv TriPosVlv MonTriPosVlv
9		Antriebsschutz ausgelöst: Es liegt ein Problem mit dem Antrieb vor	AnaDrv MonAnaDrv BinDrv MonBinDrv
10		Drehzahlalarm oben ist aktiv	MonAnaDrv
11		Drehzahlalarm unten ist aktiv	MonAnaDrv
12		Der Hold State ist aktiv nach dem Auslösen des Überwachungsfehlers	MonAnaDrv MonAnaVlv MonBinDrv MonBinVlv MonTriPosVlv
13		Safe Position ist aktiv nach Auslösen des Überwachungsfehlers	MonAnaDrv MonAnaVlv MonBinDrv MonBinVlv MonTriPosVlv
14		Signalflattern erkannt	BinMon

#	Status-Elementstatus	Beschreibung	MTP-Typen
15		Fallback-Meldung, wenn der Alarm aktiv ist	
16		Fallback-Meldung, wenn die Warnung aktiv ist	
17		Fallback-Meldung, wenn die Toleranz aktiv ist	

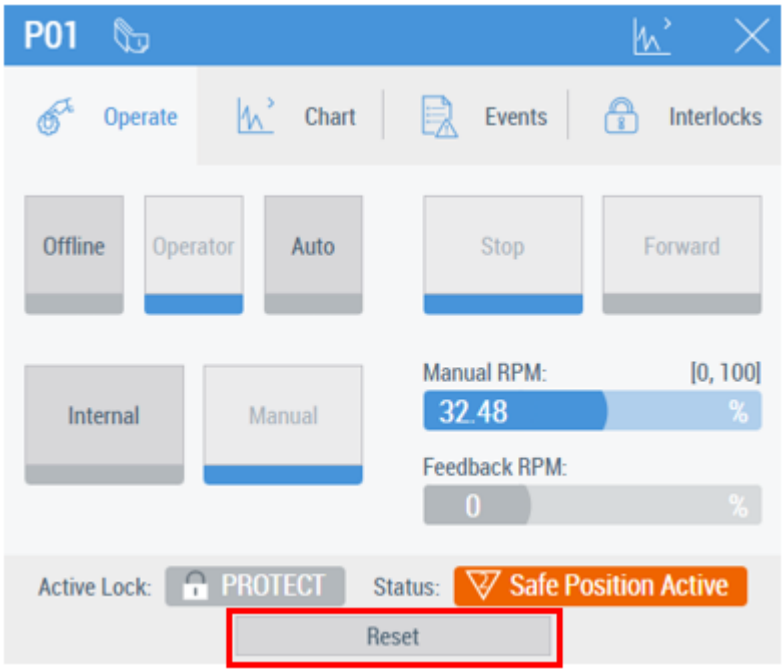
Die folgende Tabelle beschreibt die **Active Lock**-Elementstati und enthält eine Liste der MTP-Typen, die den jeweiligen Status auf ihren vordefinierten Faceplates haben können:

#	Active Lock-Elementstatus	Beschreibung	MTP-Typen
1		Keine aktiven Sperren	AnaDrv MonAnaDrv AnaVlv MonAnaVlv BinDrv MonBinDrv BinVlv MonBinVlv TriPosVlv MonTriPosVlv
2		Ermöglicht Controls	
3		Verhindert eine Aktivierung und setzt es in eine sichere Position	
4		Verhindert eine Aktivierung und setzt es in eine sichere Position, erfordert einen Resetbefehl	

Die folgende Tabelle beschreibt die **WQC**-Elementstati und enthält eine Liste der MTP-Typen, die den jeweiligen Status auf ihren vordefinierten Faceplates haben können:

#	WQC-Elementstatus	Beschreibung	MTP-Typen
1		Fehler	AnaDIntMon DIntMon AnaDrv MonAnaDrv AnaVlv MonAnaVlv BinMon BinDrv MonBinDrv BinVlv MonBinVlv TriPosVlv MonTriPosVlv
2		Funktionsprüfung	
3		Wartungsanfrage	
4		Außerhalb der Spezifikation	
5		Außer Betrieb	

Wenn ein Notfallstatus des Controls aktiv ist, erscheint die Reset-Schaltfläche:



4.5 Funktionen

Die HMI Process Library enthält eine Reihe von Funktionen. Die folgende Tabelle zeigt weitere allgemeine Funktionen aus der HMI Process Library:

Name	Kategorie	Beschreibung
AnalogCompressionValueFormatter	Formatting	Gibt für den übergebenen Wert eine Zeichenkette im komprimierten Format zurück (nicht länger als die angegebene Anzahl an Zeichen, Vorzeichen und Dezimaltrennzeichen ausgenommen). Typ des Rückgabewerts: *String*. Argumente: • value: Der Wert, der im komprimierten Format dargestellt wird. Von -10^{15} bis $+10^{15}$ (ohne Grenzwerte). Der betragsmäßig kleinste Wert kann 10^{-15} betragen. Typ: *String*. • maxCharacters: Maximale Anzahl von Zeichen, die der Wert haben sollte (ohne Vorzeichen und Dezimaltrennzeichen; kann nicht kleiner als 4 sein). Typ: *Nummer*. • showPlus: Wenn true, wird das Pluszeichen für positive Werte angezeigt. Typ: *Boolean*. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Wert undefiniert oder null ist. Der Default-Fallbackwert ist 'NaN'. Typ: *String*.
AnalogValueFormatter	Formatting	Gibt eine formatierte Zeichenkette für den übergebenen Wert zurück. Typ des Rückgabewerts: *String*. Argumente: • value: Der zu formatierende Wert.

Name	Kategorie	Beschreibung
		Typ: *String*. • maxDecimals: Wie viele Dezimalstellen der Wert haben soll. Typ: *Nummer*. • showPlus: Wenn true, wird das Pluszeichen für positive Werte angezeigt. Typ: *Boolean*. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Wert undefiniert oder null ist. Der Default-Fallbackwert ist 'NaN'. Typ: *String*.
BinaryValueFormatter	Formatting	Gibt eine formatierte Zeichenkette für den übergebenen Wert zurück. Typ des Rückgabewerts: *String*. Argumente: • value: Der zu formatierende Wert. Typ: *String*. • trueValue: Ersatzzeichenkette für den true-Wert. Typ: *String*. • falseValue: Ersatzzeichenkette für den false-Wert. Typ: *String*. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Wert nicht TRUE oder FALSE ist. Der Default-Fallbackwert ist eine leere Zeichenkette. Typ: *String*.
TernaryValueFormatter	Formatting	Gibt eine formatierte Zeichenkette für den übergebenen Wert zurück. [bVal1.toString() + bVal2.toString()]. Typ des Rückgabewerts: *String*. Argumente: • value: Der zu formatierende Wert [bVal1.toString() + bVal2.toString()], gültige Werte ◦ 'truefalse' ◦ 'falsetrue' ◦ 'falsefalse' Typ: *String*. • truefalseValue: Ersatzzeichenkette für den Wert 'truefalse'. Typ: *String*. • falsetrueValue: Ersatzzeichenkette für den Wert 'falsetrue'. Typ: *String*. • falsefalseValue: Ersatzzeichenkette für den Wert 'falsefalse'. Typ: *String*. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Wert nicht im Bereich der möglichen Werte liegt. Der Default-Fallbackwert ist eine leere Zeichenkette. Typ: *String*.

Name	Kategorie	Beschreibung
MapValueFormatter	Formatting	Gibt einen Wert zurück, der dem übergebenen Key entspricht. Typ des Rückgabewerts: *String*. Argumente: • key: Der zu ersetzende Wert Typ: *Beliebig*. • mapObj: Zuordnung für Ersetzungen **key** -> **value**. Zum Beispiel: {"key1": "value1", "key2": "value2"} Typ: *Objekt*. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Key nicht unter den Keys des mapObje ist. Der Default-Fallbackwert ist eine leere Zeichenkette. Typ: *String*.

Die folgende Tabelle zeigt die zusätzlichen MTP-spezifischen Funktionen aus der HMI Process Library:

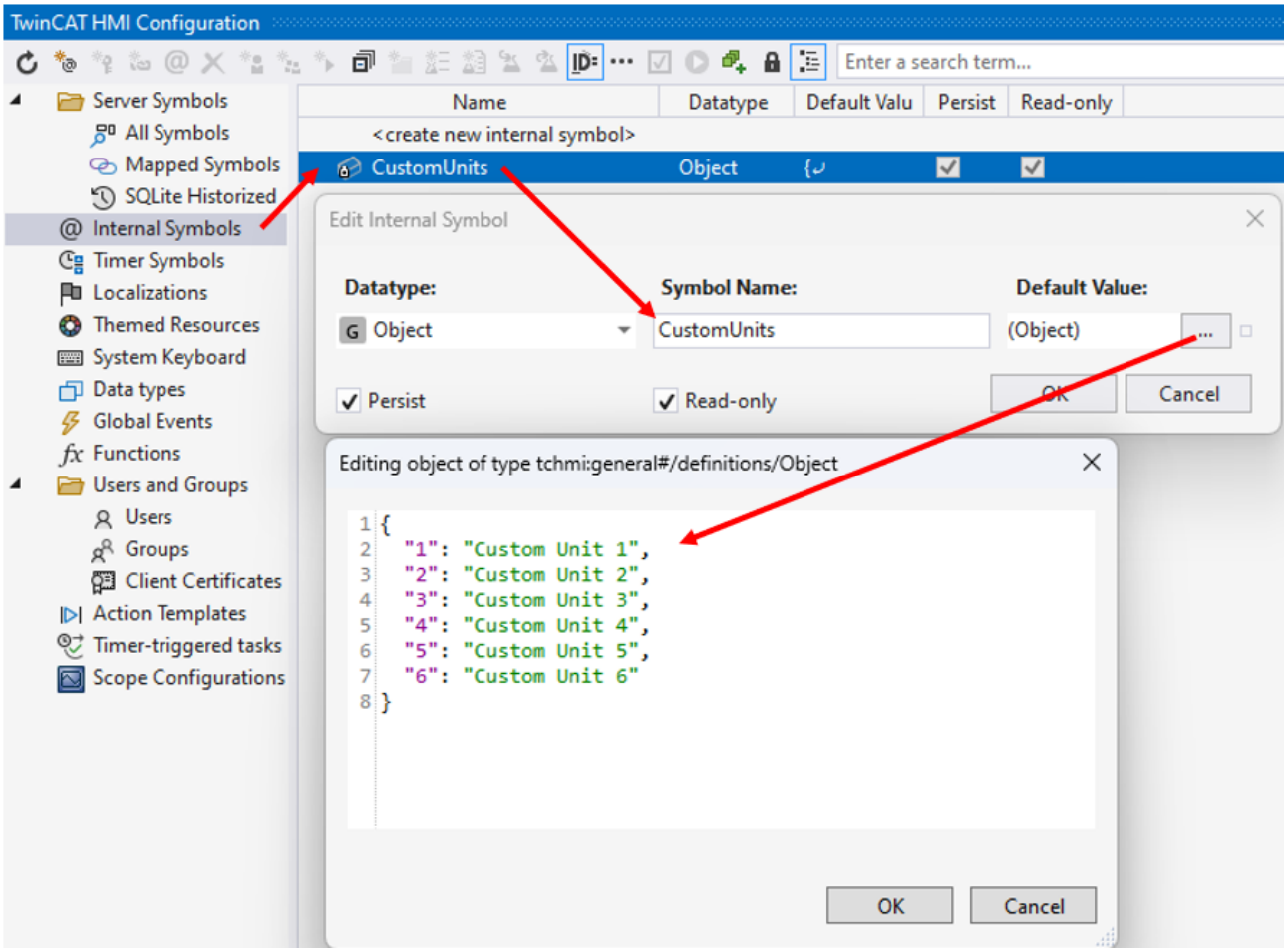
Name	Kategorie	Beschreibung
MTPIsCommandEnabled	Checking	Gibt true zurück, wenn der Befehl aktiviert ist. Typ des Rückgabewerts: *Boolean*. Argumente: • CommandEN: CommandEN-Variable des Service. Typ: *Nummer*. • Befehl: Zu prüfender Befehl. Typ: *String*.
MTPStateValueFormatter	Formatting	Rückgabe einer formatierten Statuszeichenkette für den übergebenen Wert. Typ des Rückgabewerts: *String*. Argumente: • value: Der zu formatierende Wert. Typ: *String*.
MTPUnitValueFormatter	Formatting	Gibt eine formatierte Einheiten-Zeichenkette für den übergebenen Wert zurück. Typ des Rückgabewerts: *String*. Argumente: • value: Der zu formatierende Einheiten-Wert. Typ: String. • customUnits: Wörterbuch für Ersetzungen **num** -> **unit**. **num** sollte eine positive ganze Zahl sein, die in Anführungszeichen steht. Der Defaultwert ist {"1": "Benutzerdefinierte Einheit 1", "2": "Benutzerdefinierte Einheit 2"} Typ: Objekt. • fallbackValue: Das zurückgegebene Ergebnis, wenn der Wert undefiniert oder null ist. Der Default-Fallbackwert ist eine leere Zeichenkette. Typ: *String*.

4.6 Nicht-standardisierte technische Einheiten

Die Komponenten der HMI Process Library können nicht standardisierte technische Einheiten darstellen. In der nachstehenden Tabelle sind diese Komponenten aufgeführt:

Element	Eigenschafts-/Parametername
Funktion MTPUnitValueFormatter	customUnits
Von ProcessObject geerbte Controls	Custom Units
Service-Control	Custom Units
Faceplates, die technische Einheiten darstellen	CustomUnits

Es ist möglich, die **Custom Units**-Eigenschaften und -Parameter für jedes Element einzeln festzulegen, aber am bequemsten ist es, z. B. ein internes Symbol im HMI-Projekt zu erstellen, das alle erforderlichen nicht standardisierten Einheiten definiert, und es dann mit den entsprechenden Eigenschaften/Parametern der Elemente zu verbinden. Ein Beispiel für ein internes Symbol und seine Initialisierung ist unten dargestellt:



4.7 SPS-Attribut-Funktionalität

SPS-Attribute können verwendet werden, um das TwinCAT Process HMI-Control zu konfigurieren und einige aktuelle Daten aus einem vom Benutzer erstellten FB an seine Elemente zu übergeben. Hierzu sind zwei Aktionen notwendig:

- die SPS-Attribute im FB hinzuzufügen
- den FB dem Control zuzuordnen

Benennung von FBs mit SPS-Attributen

Für eine korrekte Funktion dürfen benutzerdefinierte FB-Namen **keine** DataAssembly-Namen aus der MTP-Spezifikation (Teil 3) als Bestandteil enthalten, z. B. „BinVlv“, „AnaView“ usw. Diese Namen sind Teil der FB-Namen der MTP Runtime Library, z. B. FB_MTP_BinVlv, FB_MTP_AnaView (siehe Handbuch für TF8400 | MTP Runtime).

SPS-Attribute lassen sich in zwei Arten unterteilen:

- SPS-Konfigurationsattribute: Definieren einige Eigenschaften und das Verhalten des Controls beim Start der TwinCAT HMI-Seite.
- SPS-Datenattribute: Definieren Variablen, die Werte der Control-Elemente darstellen, wenn die TwinCAT HMI Seite läuft.

Dieses Kapitel enthält eine Liste der Attribute. Detaillierte Informationen über die Verwendung der Attribute im SPS-Programm finden Sie im Kapitel Verwendung der SPS-Attribut-Funktionalität [▶ 21].

Verwendung von [ ] in den Codevorlagen

In eckigen Klammern [ ] eingeschlossener Text in Codevorlagen ist ein Platzhalter, der durch einen entsprechenden Text ersetzt werden sollte.

Der vollständige Name des Attributs lautet `'TcHmi.ProcessLibrary.[Attribute Name]'`, wobei:

- `TcHmi.ProcessLibrary` ein fester Bestandteil ist.
- `[Attribute Name]` nur einen Namen (z. B. **RestoreBounds**) oder mehrere Namen in hierarchischer Reihenfolge, getrennt durch '.', enthalten kann (d.h. **FaceplateControl.TargetFile**).

4.7.1 Spezielle Platzhalterbeschreibung

4.7.1.1 Platzhalter PLPATH^

Der Platzhalter **PLPATH^** verweist auf das Wurzelverzeichnis der TwinCAT HMI Process Library.

4.7.1.2 Platzhalter THIS^ und THISEXP^

Der Platzhalter **THIS^** bezieht sich auf das SPS-Symbol, das dem Control zugeordnet ist. Es ersetzt SymbolPath, d.h.:

```
ADS.PLC1.GVL_MTP.V03
```

Der Platzhalter **THISEXP^** verweist auf das SPS-Symbol, das dem Control zugeordnet ist. Es ersetzt SymbolExpression, d.h.:

```
%s%THIS^%/s% or %s%ADS.PLC1.GVL_MTP.V03%/s%
```

Ein Beispiel für eine Wertzuweisung innerhalb des SPS-Attributs mit **THISEXP^** kann wie folgt aussehen:

```
DataSymbol:=THISEXP^
```

4.7.1.3 Platzhalter STAG^ und ETAG^

Der Platzhalter **STAG^** ersetzt die Start-Tags von SymbolExpression `%s%`, `%pp%`, etc.

Der Platzhalter **ETAG^** ersetzt die End-Tags von SymbolExpression `%/s%`, `%/pp%`, usw.

Ein Beispiel für eine Wertzuweisung innerhalb des SPS-Attributs unter Verwendung von **STAG^** und **ETAG^** kann wie folgt aussehen:

```
nUnit:=STAG^THIS^::nUnitETAG^
```

4.7.2 Beschreibung der SPS-Konfigurationsattribute

Fast alle SPS-Konfigurationsattribute entsprechen den im Kapitel Eigenschaften [▶ 28] beschriebenen Eigenschaften von ProcessObject sowie einige Eigenschaften, die spezifisch für die vererbten Controls sind. Nicht alle Eigenschaften sind wie die SPS-Konfigurationsattribute implementiert, aber die implementierten haben den gleichen Namen (nur ohne Leerzeichen) und die gleiche Funktionalität. Die folgende Tabelle enthält eine Liste der SPS-Konfigurationsattribute (mit Ausnahme der Faceplate-Registerkarte und der referenzierenden Funktionsattribute).

Attributname	Kategorie	Beschreibung	Mögliche Werte
GraphicControl	Common		
FaceplateControl.TargetFile	Faceplate	**Faceplate Control**-Eigenschaft: Pfad zur *.usercontrol-Datei	
FaceplateControl.Parameter	Faceplate	**Faceplate Control**-Eigenschaft: SymbolExpression für das Symbol, das Daten für das Faceplate liefert	
ShowFaceplate	Faceplate		False, True
Modal	Faceplate		False, True
Movable	Faceplate		False, True
RestorePosition	Faceplate		False, True
HideWithControl	Faceplate		False, True
ReshowWithControl	Faceplate		False, True
LabelTextHorizontalAlignment	Label		Links, Rechts, Zentriert
LabelTextVerticalAlignment	Label		Oben, Unten, Zentriert
LabelValueHorizontalAlignment	Label		Links, Rechts, Zentriert
LabelValueVerticalAlignment	Label		Oben, Unten, Zentriert
LabelModeHorizontalAlignment	Label		Links, Rechts, Zentriert
LabelModeVerticalAlignment	Label		Oben, Unten, Zentriert
LabelStatusHorizontalAlignment	Label		Links, Rechts, Zentriert
LabelStatusVerticalAlignment	Label		Oben, Unten, Zentriert
LabelTextPadding.Left	Label		[Zahl]
LabelTextPadding.LeftUnit	Label		'px'
LabelTextPadding.Top	Label		[Zahl]
LabelTextPadding.TopUnit	Label		'px'
LabelTextPadding.Right	Label		[Zahl]
LabelTextPadding.RightUnit	Label		'px'
LabelTextPadding.Bottom	Label		[Zahl]
LabelTextPadding.BottomUnit	Label		'px'
LabelValuePadding.Left	Label		[Zahl]
LabelValuePadding.LeftUnit	Label		'px'
LabelValuePadding.Top	Label		[Zahl]
LabelValuePadding.TopUnit	Label		'px'
LabelValuePadding.Right	Label		[Zahl]
LabelValuePadding.RightUnit	Label		'px'
LabelValuePadding.Bottom	Label		[Zahl]
LabelValuePadding.BottomUnit	Label		'px'
LabelModePadding.Left	Label		[Zahl]
LabelModePadding.LeftUnit	Label		'px'
LabelModePadding.Top	Label		[Zahl]
LabelModePadding.TopUnit	Label		'px'

Attributname	Kategorie	Beschreibung	Mögliche Werte
LabelModePadding.Right	Label		[Zahl]
LabelModePadding.RightUnit	Label		'px'
LabelModePadding.Bottom	Label		[Zahl]
LabelModePadding.BottomUnit	Label		'px'
LabelStatusPadding.Left	Label		[Zahl]
LabelStatusPadding.LeftUnit	Label		'px'
LabelStatusPadding.Top	Label		[Zahl]
LabelStatusPadding.TopUnit	Label		'px'
LabelStatusPadding.Right	Label		[Zahl]
LabelStatusPadding.RightUnit	Label		'px'
LabelStatusPadding.Bottom	Label		[Zahl]
LabelStatusPadding.BottomUnit	Label		'px'
Chart	Chart	Die Variable wird automatisch im TwinCAT HMI historisiert	Sie benötigt keine Werte
ShowRotation	Rotation	Speziell für Rührwerke und Förderanlagen	False, True
RotationDuration	Rotation	Speziell für Rührwerke und Förderanlagen	[Zahl]

4.7.2.1 Beschreibung der Funktionsattribute

Einige der Konfigurationsattribute sind dazu bestimmt, eine Funktion zur Handhabung des Wertes zu definieren. Das **TcHmi.ProcessLibrary.Format**-Attribut ist ein Attribut, das eine Formatierungsfunktion definiert. Die folgende Tabelle enthält eine Liste der SPS-Attribute, die zusammen mit dem **TcHmi.ProcessLibrary.Format**-Attribut verwendet werden können.

Attributname	Kategorie	Beispiel für den Wert von TcHmi.ProcessLibrary.Format
LabelValue	Label	`TcHmi.Functions.Beckhoff.PI.ProcessLibrary.AnalogCompressionValueFormatter,4`
LabelValueUnit	Label	`TcHmi.Functions.Beckhoff.PI.ProcessLibrary.MTPUnitValueFormatter`

4.7.2.2 Beschreibung der Attribute der Faceplate-Registerkarte

Die Konfigurationsattribute der Faceplate-Registerkarten definieren die Nummer und den Inhalt der Faceplate-Registerkarten. Der vollständige Attributname der Faceplate-Registerkarte lautet

'TcHmi.ProcessLibrary.Tab[Nummer].[Attributname]',

Wobei:

- **TcHmi.ProcessLibrary.Tab** ein fester Bestandteil ist
- **[Nummer]** die Nummer der Registerkarte (1, 2, 3, usw.) ist
- **[Attributname]** den Namen des Attributs der Registerkarte enthält

Die Werte der Attribute der Registerkarte können nur einen Parameter oder mehrere durch ein Komma getrennte Parameter enthalten. In der folgenden Tabelle werden die Parameter in der Reihenfolge ihres Erscheinens beschrieben.

Attributname	Beschreibung	Beispiel
Name	Titel der Registerkarte	
TargetFile	Parameter: • Pfad zur *.usercontrol-Datei, die die Registerkarte darstellt	

Attributname	Beschreibung	Beispiel
	• Aktiviert das Vorladen des Inhalts der Registerkarte (false, true) ***)** • Aktiviert die Voranbindung des Inhalts der Registerkarte (false, true) ***)** • Aktiviert das Keep Alive des Inhalts der Registerkarte (false, true) ***)**	
Parameter	SymbolExpression oder SymbolPath für das Symbol, das Daten für das Faceplate liefert	
Ausrichtung	Ausrichtung des Titels der Registerkarte. Parameter: • horizontale Ausrichtung (Links, Rechts, Zentriert) • vertikale Ausrichtung (Oben, Unten, Zentriert)	
Icon	Symbol neben dem Titel der Registerkarte. Parameter: • Pfad zur Bilddatei *.svg • Breite (Zahl) • Höhe (Zahl) • Breiteneinheiten (px, %) • Höheneinheiten (px, %)	

***)** - Detaillierte Informationen finden Sie im Abschnitt "Control-Lebenszyklus" im TE2000 TC3 HMI Handbuch.

4.7.3 Beschreibung der SPS-Datenattribute

Fast alle SPS-Datenattribute entsprechen den im Kapitel Eigenschaften [▶ 28] beschriebenen Eigenschaften von ProcessObject sowie einige Eigenschaften, die spezifisch für die vererbten Controls sind. Nicht alle Eigenschaften sind als SPS-Datenattribute implementiert, aber die implementierten haben denselben Namen (nur ohne Leerzeichen) und dieselbe Funktionalität. Die folgende Tabelle enthält eine Liste der SPS-Datenattribute.

Attributname	Kategorie	Beschreibung
GraphicStatus	Common	
HeaderDescription	Faceplate	
HeaderText	Faceplate	
LabelText	Label	
LabelValue	Label	
LabelValueAdd1	Label	Wert für zusätzlichen LabelValue 1 (siehe Anwendungsfall im Abschnitt SPS-Datenattribute [▶ 23])
LabelValueAdd2	Label	Wert für zusätzlichen LabelValue 2
LabelValueUnit	Label	
LabelValueUnitAdd1	Label	Einheiten für zusätzlichen LabelValue 1
LabelValueUnitAdd2	Label	Einheiten für zusätzlichen LabelValue 2
LabelStateMode	Label	
LabelSourceMode	Label	
LabelInterlockStatus	Label	
LabelAlarmStatus	Label	
LabelWQCStatus	Label	
LabelInterlockMessage		
LabelAlarmMessage		
LabelWQCMessage		

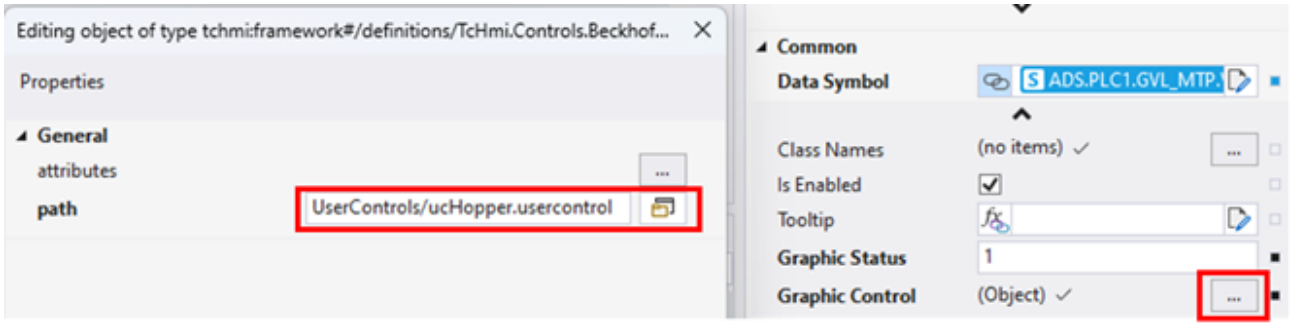
Attributname	Kategorie	Beschreibung
ValvePosition	Common	Speziell für 3- und 4-Wegeventile

5 Anpassung der Darstellung des Controls

Mit der **Graphic Control**-Eigenschaft können Sie das grafische Default-Bild des Controls ändern.

Die Verwendung von SVG-Bildern in UserControls, die für die **Graphic Control**-Eigenschaft vorgesehen sind, ermöglicht es, die in der HMI Process Library definierten Stilvorlagen zu verwenden. Hier sollten einige Empfehlungen implementiert werden.

In der Regel sollte für die **Graphic Control**-Eigenschaft **.usercontrol* bei gewünschter grafischer Darstellung des Controls Folgendes ausgewählt werden:

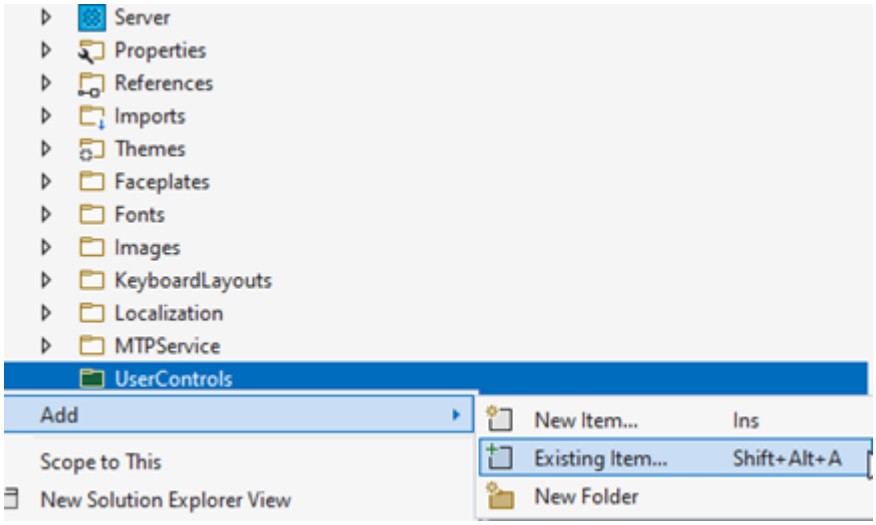


5.1 UserControl-Dateien

Jedes UserControl wird durch 2 Dateien repräsentiert:

- **.usercontrol*
- **.usercontrol.json*

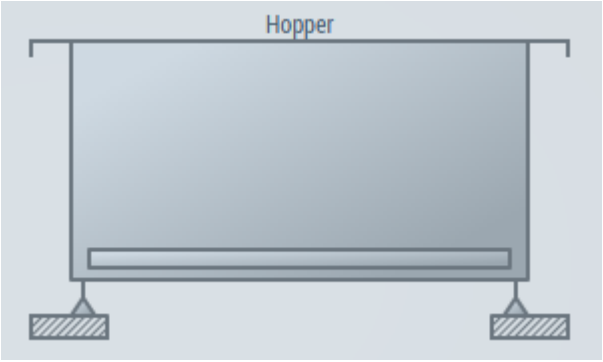
Beispielhafte UserControls für die Verwendung mit der **Graphic Control**-Eigenschaft können per E-Mail an processindustry@beckhoff.com angefordert werden. Sie sollten einfach zum HMI-Projekt hinzugefügt werden, das die HMI Process Library verwendet. Zum Beispiel kann der Ordner "*UserControls*" zum HMI-Projekt hinzugefügt werden und dann können die gewünschten UserControls als vorhandene Elemente hinzugefügt werden:



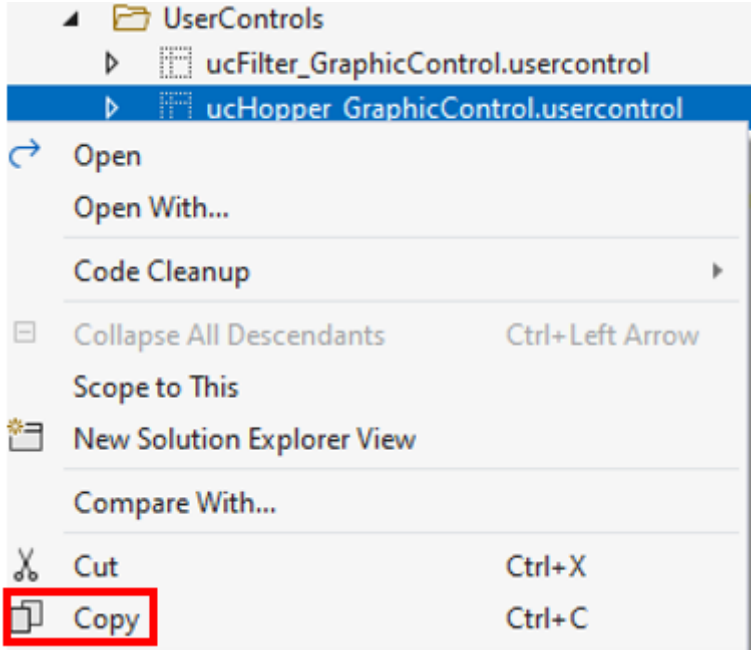
Wählen Sie einfach **.usercontrol*-Dateien aus und die entsprechenden **.json*-Dateien werden automatisch hinzugefügt.

5.2 Erstellen eines neuen UserControls mit SVG basierend auf dem bestehenden Beispiel

Weitere Erläuterungen beziehen sich auf das Beispiel ucHopper_GraphicControl



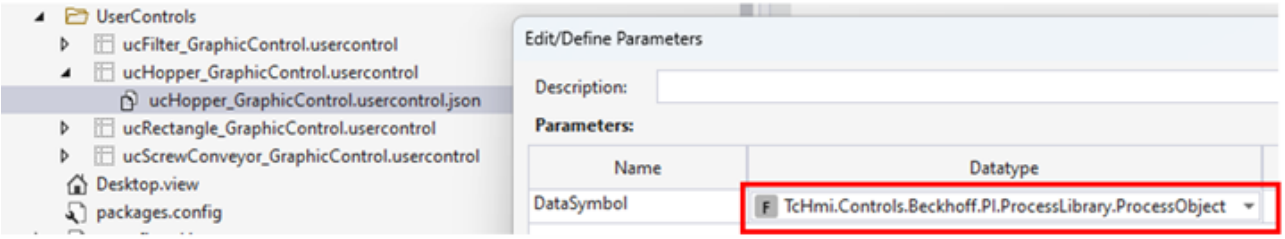
- ucHopper_GraphicControl soll kopiert und



- dann eingefügt werden (kann im gleichen Projektordner eingefügt werden).
- Der neue UserControl-Name ist in den gewünschten Namen zu ändern

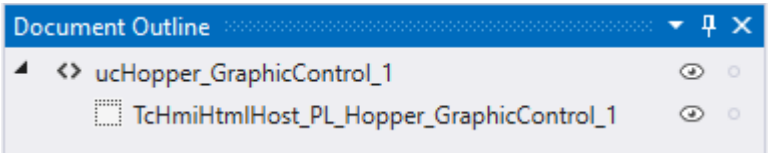
5.3 Bearbeitung der *.usercontrol.json-Datei

Der Typ des **DataSymbol**-Parameters kann bei Bedarf auf den gewünschten Typ geändert werden:



5.4 Bearbeitung der *.usercontrol-Datei

Die **.usercontrol*-Datei enthält das HTML Host-Control (TcHmi.Controls.System.TcHmiHtmlHost):



Der SVG-Code sollte direkt in HTML bearbeitet werden. Attribute und Teile, die **gelb markiert** sind, können bearbeitet werden.

5.4.1 - und <svg>-Attribute

```
<div id="ucHopper_GraphicControl_1" data-tchmi-type="TcHmi.Controls.System.TcHmiUserControl"
     data-tchmi-top="0" data-tchmi-left="0" data-tchmi-width="100" data-tchmi-height="100"
     data-tchmi-width-unit="%" data-tchmi-height-unit="%"
     data-tchmi-creator-viewport-width="90" data-tchmi-creator-viewport-height="50"
     data-tchmi-width-mode="Value" data-tchmi-height-mode="Value">
<div id="TcHmiHtmlHost_PL_Hopper_GraphicControl_1" data-tchmi-type="TcHmi.Controls.System.TcHmiHtmlHost"
     data-tchmi-height="100" data-tchmi-height-unit="%" data-tchmi-left="0" data-tchmi-left-unit="px"
     data-tchmi-top="0" data-tchmi-top-unit="px" data-tchmi-width="100" data-tchmi-width-unit="%"
     data-tchmi-width-mode="Value" data-tchmi-height-mode="Value" data-tchmi-is-enabled="true">
    <svg class="SVG" width="100%" height="100%" viewBox="0 0 90 50" preserveAspectRatio="none">
```

- Die -**id** sollte so geändert werden, dass sie im HMI-Projekt eindeutig ist.
- **Breite** und **Höhe** sollten für UserControl und SVG viewBox gleich sein. SVG viewBox legt die Abmessungen der leeren Fläche fest, auf der die SVG-Elemente platziert werden.
- Wenn das **preserveAspectRatio**-Attribut = "none", dann kann die SVG skaliert werden, ohne dass ihre Default-Proportionen beibehalten werden. Das Löschen dieses Attributs bedeutet, dass die Proportionen bei der Skalierung beibehalten werden.

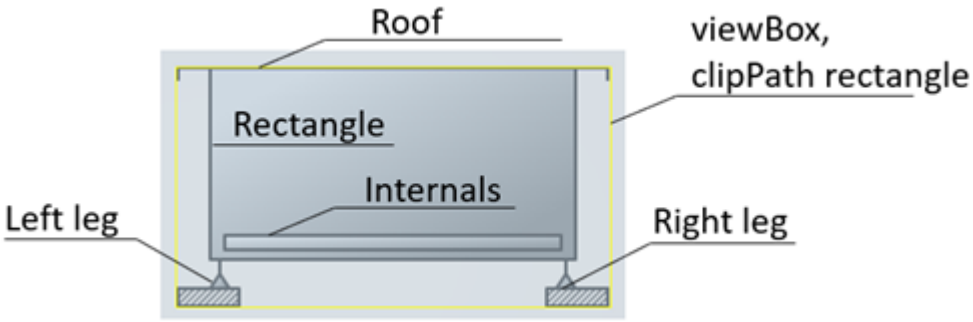
5.4.2 <defs>-Teil

```
        <defs>
            <linearGradient id="SVGBackground_{Id}" x1="0" y1="1" x2="0" y2="0"
                            class="tchmi-styleprovider-linear-gradient SVGGradient-Rotate"
                            gradientTransform="rotate(135, 0.5, 0.5)">
                <stop class="Color0" offset="0%" />
                <stop class="Color100" offset="100%" />
            </linearGradient>
        </defs>
        <defs>
            <clipPath id="SVGBorderPath_Hopper">
                <rect width="90" height="50" x="0" y="0" />
            </clipPath>
        </defs>
        <defs>
            <pattern id="diagonalHatch_Hopper" width="1" height="1"
                     patternTransform="rotate(45)" patternUnits="userSpaceOnUse">
                <line class="SVGElement" x1="0" y1="0" x2="0" y2="2" />
            </pattern>
        </defs>
```

- Der Abschnitt <linearGradient> sollte nicht geändert werden. Er stellt den Default-Stil der HMI Process Library für das **fill**-Attribut von SVG-Elementen zur Verfügung. **Id**="SVGBackground_{Id}" sollte nicht geändert werden.
- Der Abschnitt <clipPath> kann geändert oder gelöscht werden. Sein **id**-Attribut sollte geändert werden, um es innerhalb des HMI-Projekts eindeutig zu machen. <clipPath> wird benötigt, um die Grenzen von svg-Elementen zu schneiden. Es können mehrere <clipPath>-Elemente hinzugefügt werden. In diesem Fall sollten ihre **id**-Attribute unterschiedlich und innerhalb des HMI-Projekts eindeutig sein.
- Der Abschnitt <pattern> kann geändert oder gelöscht werden. Sein **id**-Attribut sollte geändert werden, um es innerhalb des HMI-Projekts eindeutig zu machen. <pattern> wird benötigt, um svg-Elemente mit bestimmten Mustern zu füllen, zum Beispiel mit Schraffuren. Es können mehrere <pattern>-Elemente hinzugefügt werden. In diesem Fall sollten ihre **id**-Attribute unterschiedlich und innerhalb des HMI-Projekts eindeutig sein.

5.4.3 Der Teil mit den SVG-Elementen

Die folgende Abbildung zeigt die SVG-Elemente des usHopper_Graphic-Control:




```
        <!--Rectangle    -->
        <rect class="SVGElement" width="76" height="40" x="7" y="0"
              fill="url(#SVGBackground_{Id})" />
        <!--Roof-->
        <path class="SVGElement SVGElement-Border" d="m0,3 0,-3 90,0 0,3"
              clip-path="url(#SVGBorderPath_Hopper)" fill="none" />
        <!--Internals-->
        <rect class="SVGElement" width="70" height="3" x="10" y="35"
              fill="url(#SVGBackground_{Id})" />
        <!--Left leg-->
        <path class="SVGElement" d="m9,40 0,3" />
        <path class="SVGElement" d="m9,43 -2,3 4,0 Z" fill="url(#SVGBackground_{Id})" />
        <rect class="SVGElement" width="13" height="4" x="0" y="46" fill="url(#diagonalHatch_Hopper)" />
        <path class="SVGElement SVGElement-Border" d="m0,46 0,4 13,0"
              clip-path="url(#SVGBorderPath_Hopper)" fill="none" />
        <!--Right leg-->
        <path class="SVGElement" d="m81,40 0,3" />
        <path class="SVGElement" d="m81,43 -2,3 4,0 Z" fill="url(#SVGBackground_{Id})" />
        <rect class="SVGElement" width="13" height="4" x="77" y="46" fill="url(#diagonalHatch_Hopper)" />
        <path class="SVGElement SVGElement-Border" d="m90,46 0,4 -13,0"
              clip-path="url(#SVGBorderPath_Hopper)" fill="none" />
    </svg>
</div>
</div>
```

Zum Zeichnen eines Bildes können verschiedene svg-Elementtypen verwendet werden. "SVGElement" sollte für alle **Klassen**-Elemente verwendet werden. Es stellt Default-Stilvorgaben der HMI Process Library zur Verfügung, z. B. Strichfarbe, Breite usw. Die "SVGElement-Border" **Klasse** kann hinzugefügt werden, wenn die Breite verdoppelt werden soll. Dies ist in der Regel der Fall, wenn der Strich mit dem Rand der SVG-ViewBox oder dem <clipPath>-Element übereinstimmt, das als **clip-path**-Attribut für dieses Element festgelegt ist.

Weitere Informationen zum Zeichnen von SVGs finden Sie hier: https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorials/SVG_from_scratch.

6 Anhang

6.1 Glossar

Abkürzung/Begriff	Bedeutung
DCS	Verteilte Steuerung
DataAssembly	Dies ist ein MTP-spezifischer Begriff, der den Datensatz bezeichnet, der die physische oder virtuelle Einheit beschreibt, die durch eine Regelung dargestellt wird, d. h. Pumpen, Behälter, Ventile, PID-Regler usw.
FB	Funktionsbaustein
HMI	Mensch-Maschine-Schnittstelle
JS	JavaScript Programmiersprache
MTP	Module Type Package
OS	Betriebssystem
P&ID	Rohrleitungs- und Instrumentierungsplan
PC	Personal Computer
PID	Proportional-Integral-Differential
SPS (PLC)	Speicherprogrammierbare Steuerung (Programmable Logic Controller)
POL	Process orchestration layer
SCADA	Überwachungssteuerung und Datenerfassung
Symbol	In TwinCAT HMI ist ein Symbol ein adressierbarer Datenpunkt, der vom HMI-Server angezeigt wird. Meistens handelt es sich dabei um eine SPS-Variable, die über das TMC exportiert wird, es kann sich aber auch um ein dynamisches oder internes Symbol handeln. Näheres dazu im Kapitel Begriffe [▶ 27]. Ein Symbol ist funktionell analog zu einem SCADA-Tag.
SymbolExpression	Eine Darstellung eines Symbols als Zeichenkette, zum Beispiel: %s%ADS.PLC1.GVL_MTP.V03%/s% oder %s%ADS.PLC1.GVL_MTP.V03::TagName%/s% Näheres dazu im Kapitel Begriffe [▶ 27].
SymbolPath	Eine Darstellung eines Symbols als Pfad, zum Beispiel: ADS.PLC1.GVL_MTP.V03 Näheres dazu im Kapitel Begriffe [▶ 27].
TMC	TMC (Symboldatei) ist eine XML-basierte Symboltabelle, die aus dem SPS-Projekt generiert wird. Er legt fest, welche SPS-Variablen (Namen, Datentypen, Attribute) von TwinCAT runtime veröffentlicht werden und damit zu adressierbaren Symbolen für Clients wie den TwinCAT HMI Server (ADS/OPC UA) werden.
UserControl	Benutzersteuerelement im TwinCAT HMI Engineering, das durch Dateien mit den **.usercontrol*- und **.usercontrol.json*-Erweiterungen repräsentiert wird
VDI	Verein Deutscher Ingenieure (Englisch: Association of German Engineers)

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Excel, IntelliSense, Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mehr Informationen:
www.beckhoff.com/mtp

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com