

## Contents

TwinCAT HMI Project Generator .....	2
Home Page .....	2
Localization Variables .....	2
Bring User Controls into the project .....	3
Populate Manual and Automatic Mode contents .....	4
Operation Mode User Control .....	4
Feeder and Operation Mode Menu .....	6
Animation User Control .....	8
Internal symbol for Feeder selection .....	9
Events for toggling feeder data .....	10
Alarming and Messaging .....	11
Historization .....	11
Recipe Management .....	14
Enum Descriptions .....	15
Appendix – Build User Controls (UCs) .....	17
Create Automatic Axis Info UC .....	17
Create Manual Axis UC .....	18
Create FeederAutomatic UC .....	20
Create FeederManual UC .....	21
Create Animation UC .....	22

## TwinCAT HMI Project Generator

New TwinCAT HMI Project using Project Generator wizard

1. Base Application template
2. Responsive Application
3. Default theme – Base
4. Navigation menu on the left; TcHmiContainer on the right
5. Two views – Mobile and Desktop switched at 1000px width
6. Add 7 pages:
  - a. Home
  - b. Automatic Mode
  - c. Manual Mode
  - d. Events
  - e. Trending
  - f. Recipes
  - g. Settings

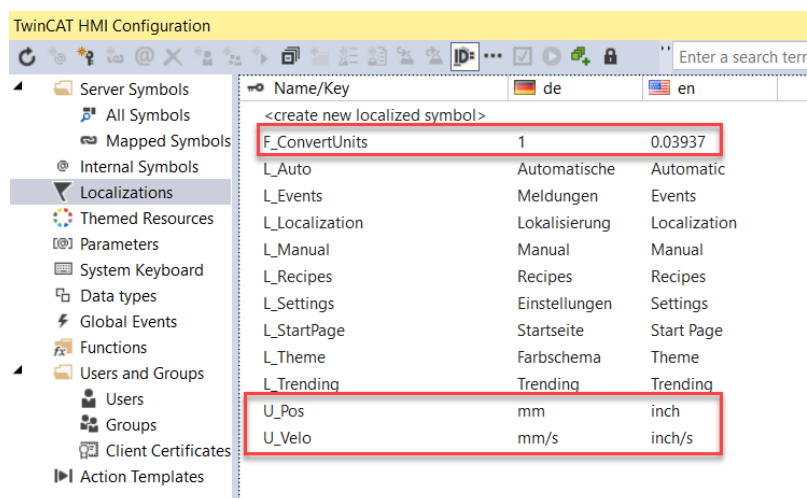
## Home Page

1. Click on Gallery Explorer -> Samples -> Woodworking line -> Substations -> Beckhoff\_hmi\_plain\_no\_bg\_v02.png
2. Drag and drop the picture into the Images folder of the HMI project tree
3. Drag and drop this image from the solution tree into the Designer in Home.content
4. Rename image and change layout to make it fit entire content page

## Localization Variables

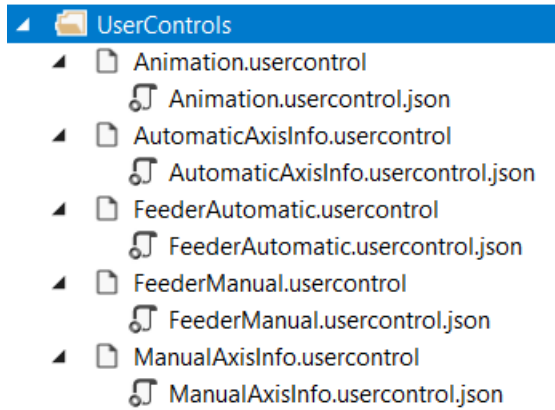
Add the following localization variables (Ensure case matches the case in the picture otherwise pre-done control bindings will not work):

1. U\_Pos
2. U\_Velo
3. F\_ConvertUnits [1 (de) : 0.03937 (en)]

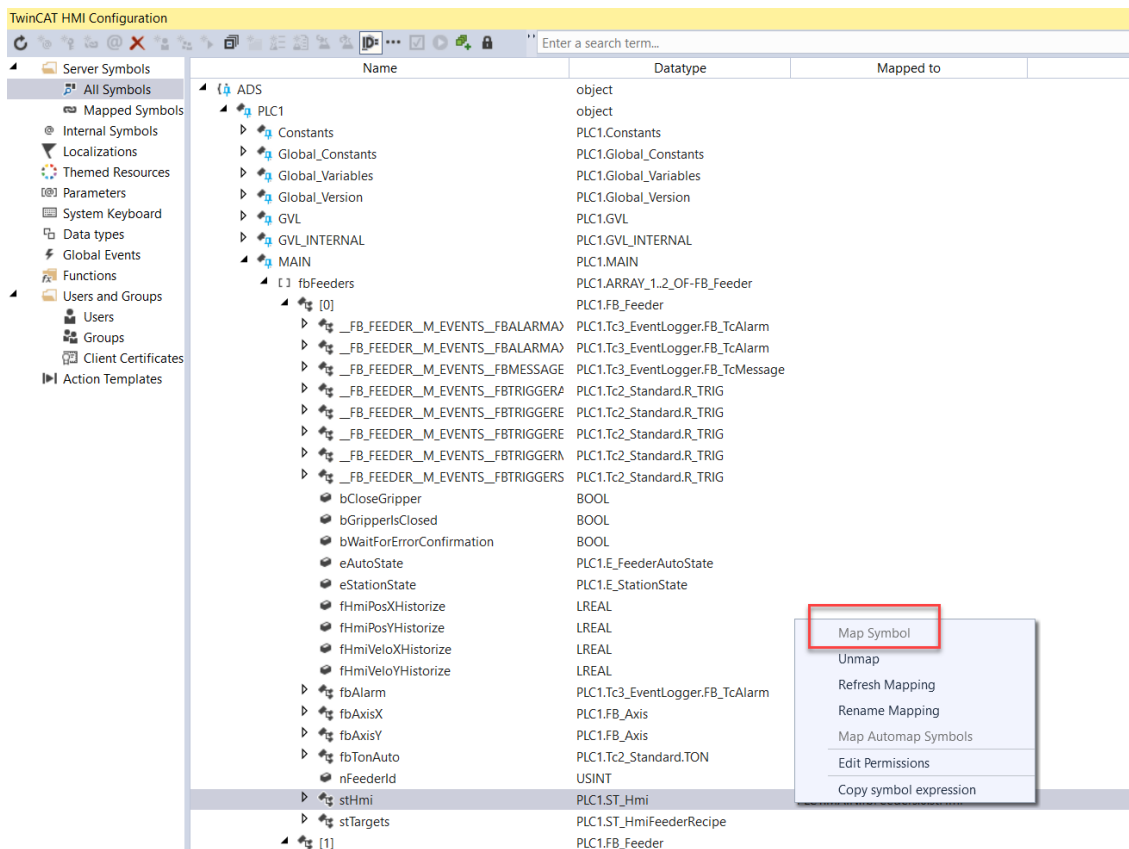


## Bring User Controls into the project

1. Create "UserControls" Folder. Ensure name is exact otherwise UC references in HTML might get broken
2. Then add each UC inside the UserControls folder by clicking on "Add Existing Item"

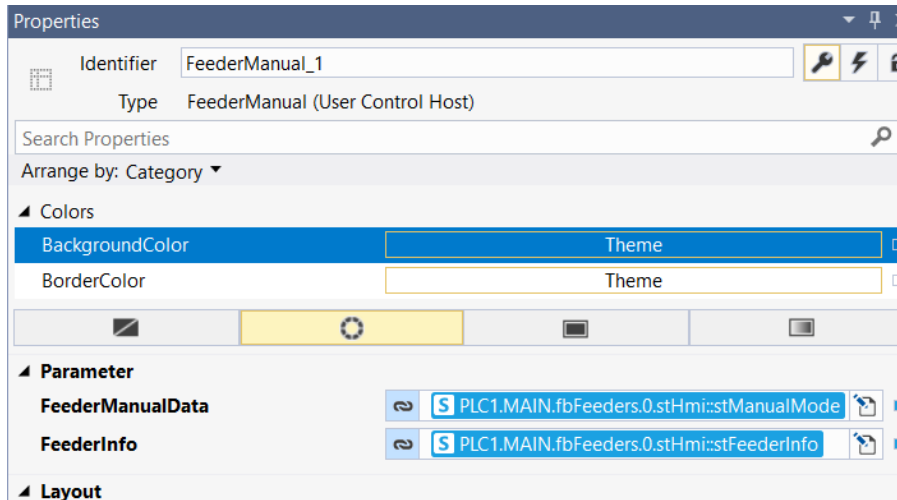


3. Map stHmi variables for both feeders from the PLC. This will make PLC datatypes available inside the HMI project. UC parameters use these datatypes:

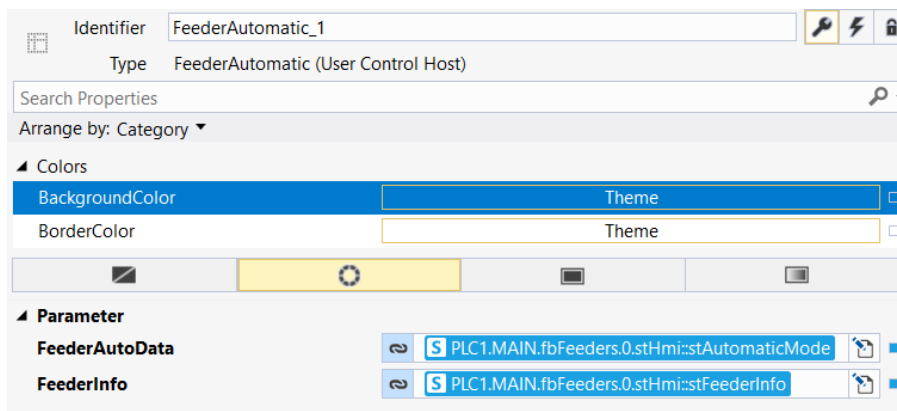


## Populate Manual and Automatic Mode contents

1. Add one instance of FeederManual UC on ManualMode content and bind to PLC symbols:

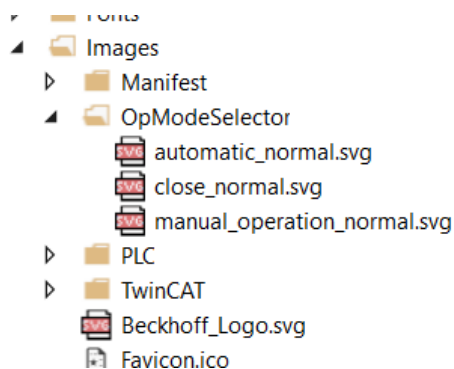


2. Add one instance of FeederAutomatic UC on AutoMode content and bind to PLC symbols:



## Operation Mode User Control

1. Create a new UC "OpModeSelector" and add "Grid" control with 3 rows
2. Add one button each to all rows with 2% spacing at Top and Bottom, 30% spacing at Left and Right
3. Add folder under Images and add images from the Images folder in the repo:



4. Add these images as icons to buttons with 100% width and height
5. Add parameter of type `feeders[0].stHmi.OperationMode` to the UC

Edit/Define Parameters

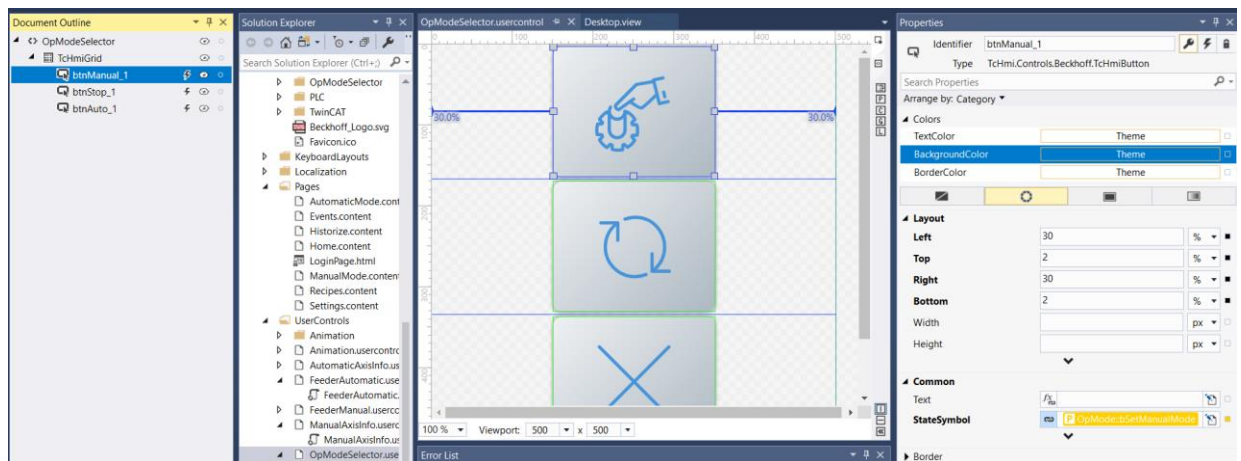
Description:

**Parameters:**

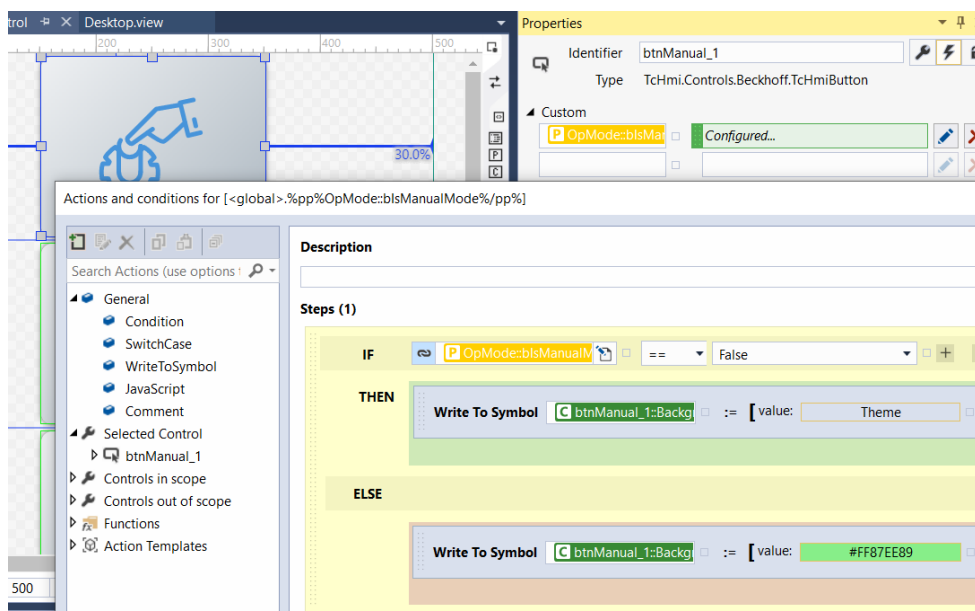
Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
OpMode	S PLC1.ST_HmiC	(Object) ...	(Object) ...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

`tchmi:server#/definitions/PLC1.ST_HmiOperationMode`

6. Map 'State symbol' of each button to parameters `bSetAutoMode`, `bSetManualMode`, `bSetStopMode` respectively

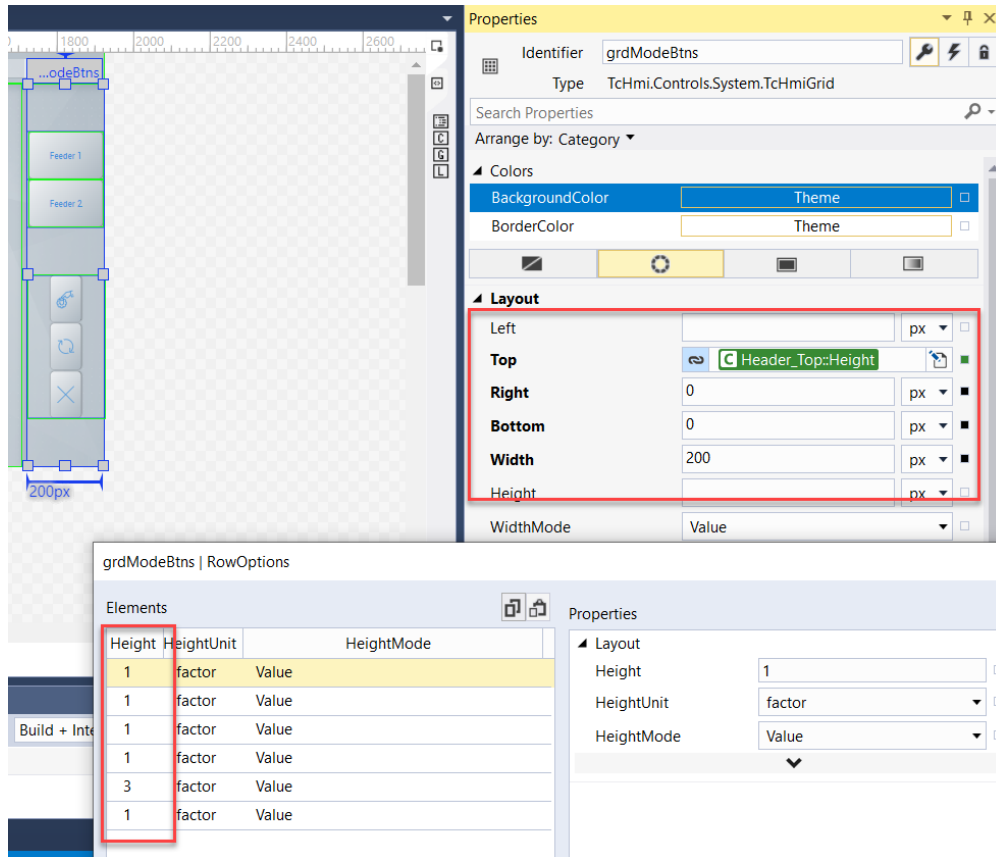


7. Add custom trigger conditions for each button for example: 'isAutoMode' -> if 'isAutoMode TRUE then background color Green ELSE theme'



## Feeder and Operation Mode Menu

1. Change the width of TcHmiContainer (right side of Desktop.view) to 200 px
2. Add Grid occupying all space of this container with 6 rows – all cells with factor 1, fifth cell with factor 3



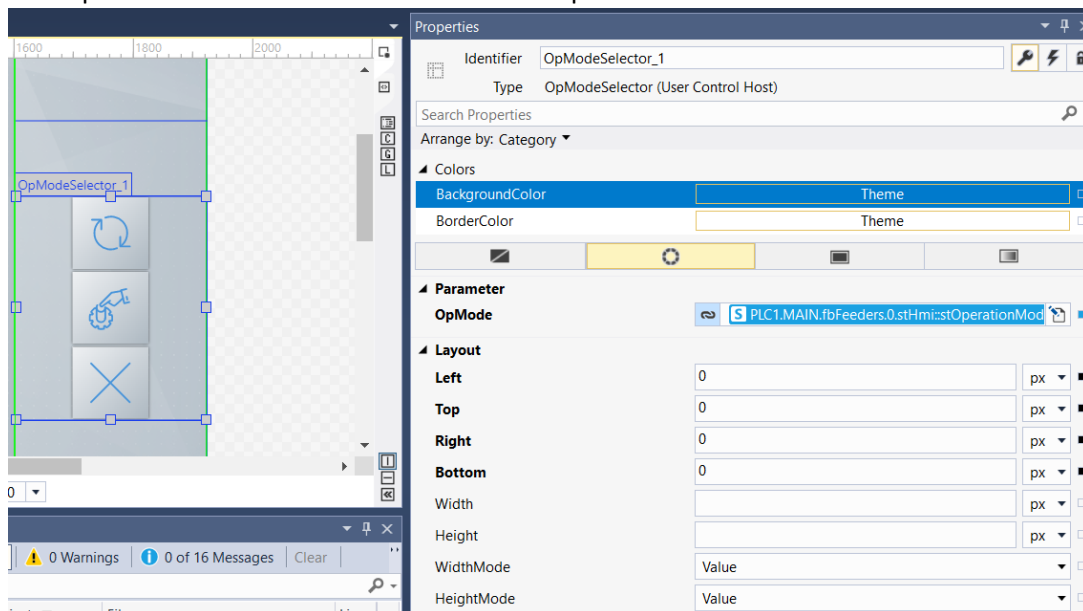
The screenshot shows the TcHmiDesigner interface with a design view on the left and a Properties panel on the right. The design view shows a vertical container with a width of 200px, containing two 'Feeder' labels and a grid of buttons. The Properties panel for the 'grdModeBtms' control (Type: TcHmi.Controls.System.TcHmiGrid) is shown. The 'Layout' section is highlighted with a red box, showing the following settings:

- Left: px
- Top: Header\_Top::Height
- Right: 0 px
- Bottom: 0 px
- Width: 200 px
- Height: px
- WidthMode: Value

Below the Properties panel, the 'grdModeBtms | RowOptions' table is visible, showing the row configuration:

Height	HeightUnit	HeightMode
1	factor	Value
1	factor	Value
1	factor	Value
1	factor	Value
3	factor	Value
1	factor	Value

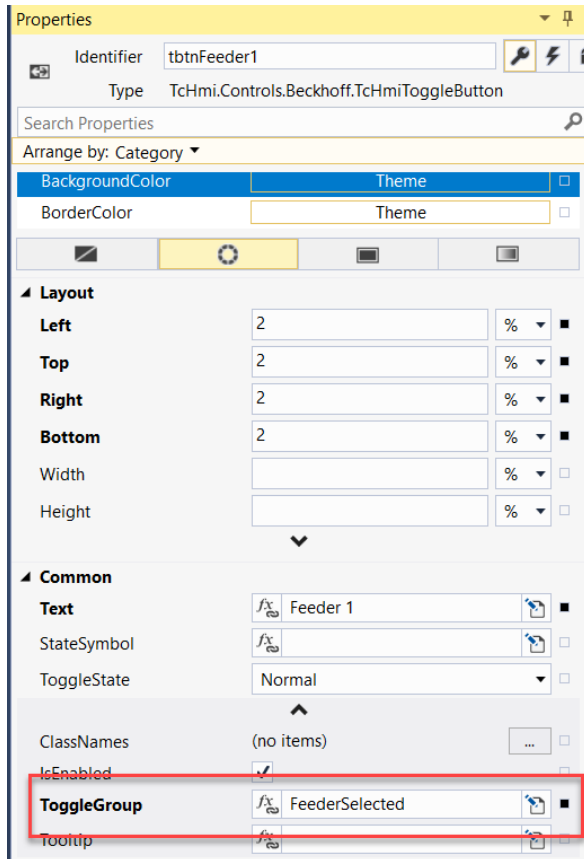
3. Add OpMode UC instance in the 5<sup>th</sup> cell and map to Feeder 1 variable from the HMI



The screenshot shows the TcHmiDesigner interface with a design view on the left and a Properties panel on the right. The design view shows the 'OpModeSelector\_1' control placed in the 5th cell of the grid. The Properties panel for the 'OpModeSelector\_1' control (Type: OpModeSelector (User Control Host)) is shown. The 'Parameter' section is highlighted, showing the 'OpMode' property mapped to the variable 'PLC1.MAIN.fbFeeders.0.stHmi::stOperationMod'. The 'Layout' section shows the following settings:

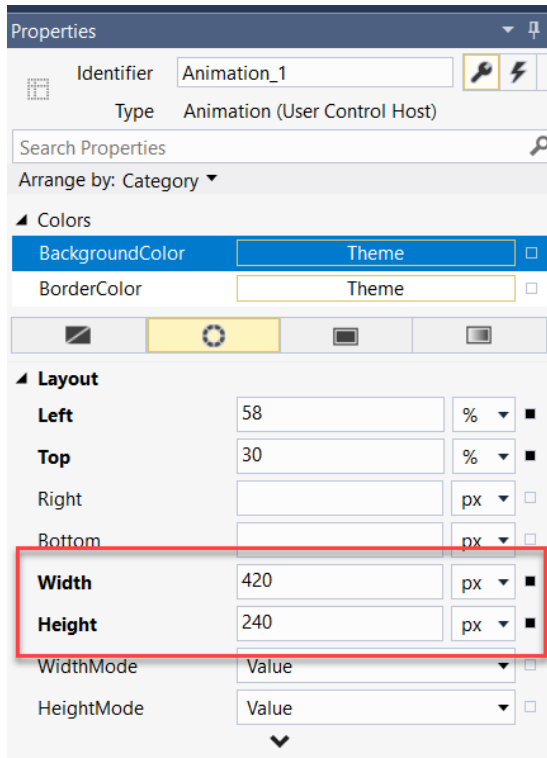
- Left: 0 px
- Top: 0 px
- Right: 0 px
- Bottom: 0 px
- Width: px
- Height: px
- WidthMode: Value
- HeightMode: Value

4. Add toggle buttons in second and third cells of the grid – 2% spacing from Left Right Top Bottom. Change the Text attributes to “Feeder 1” and “Feeder 2” respectively.
5. Add ‘ToggleGroup’ for each button, which will ensure only one button from the group is toggled on at a time

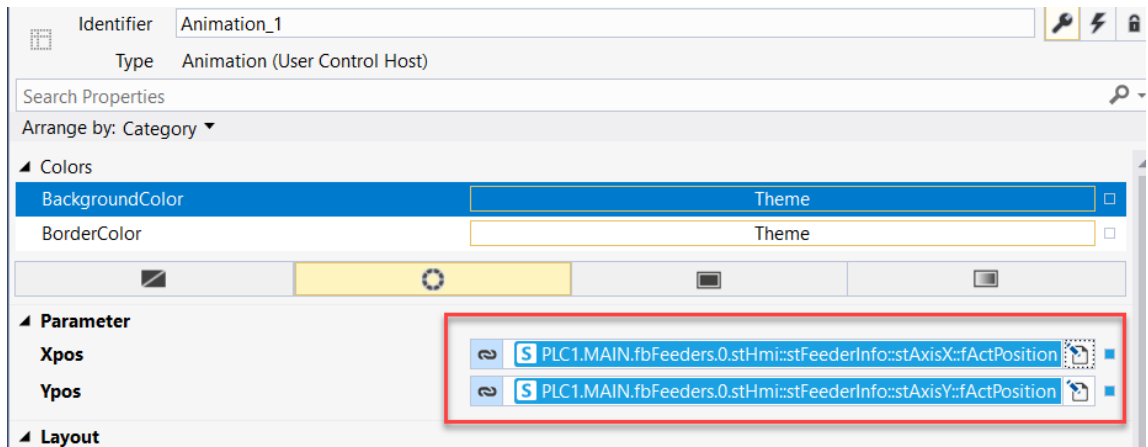


## Animation User Control

1. Add one instance of the Animation UC on the Home page with the following layout properties:



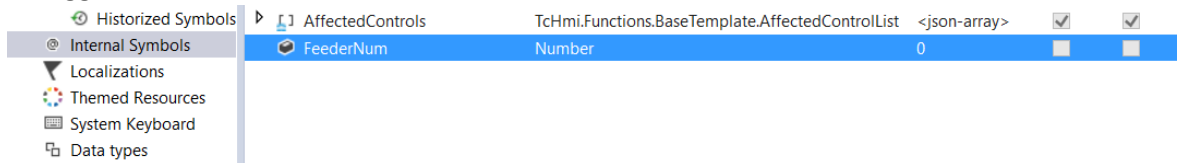
2. Bind the parameters of this UC instance:



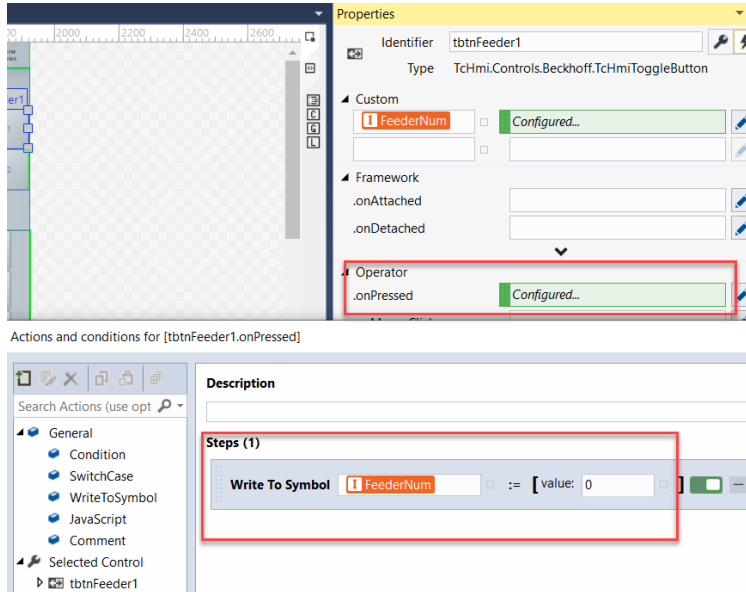


## Internal symbol for Feeder selection

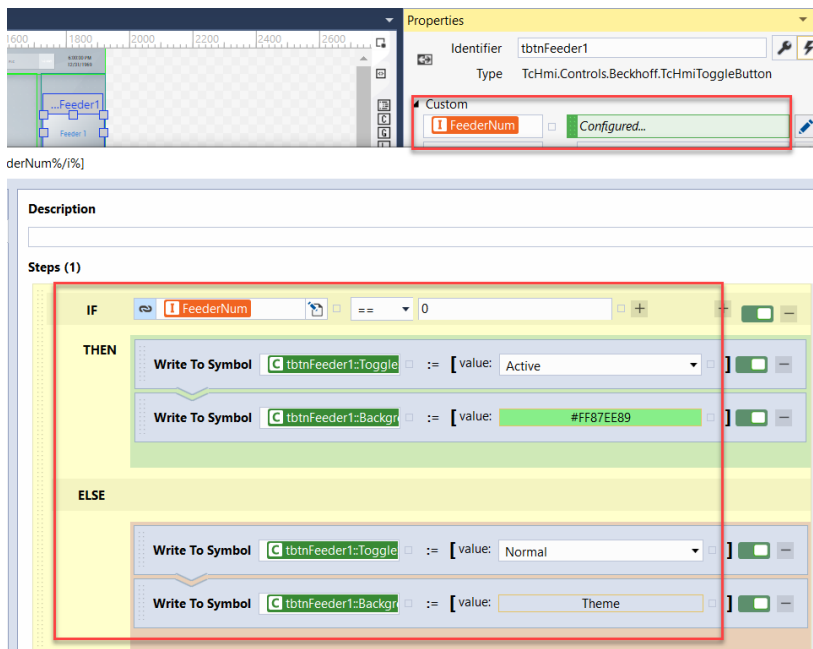
1. We need to add an internal variable to switch Feeders between 1 and 2. This symbol will be used to toggle feeder data



2. Write to internal symbol values "0" or "1" at 'onPressed' event of each toggle button:

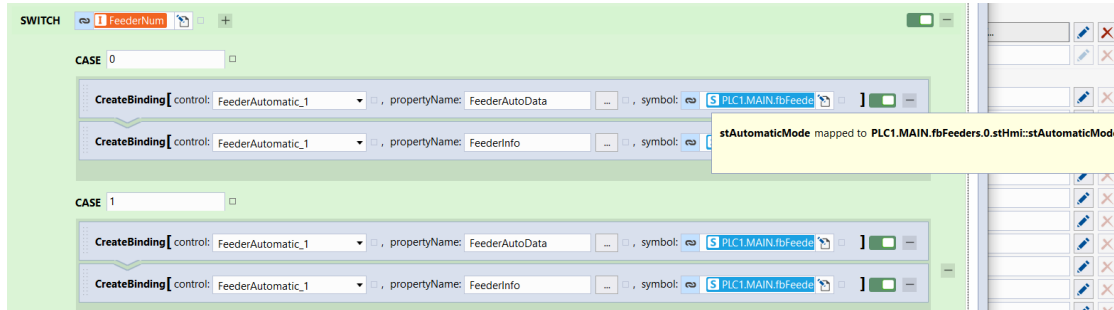


3. Add custom trigger condition with internal symbol to change toggleState and BackgroundColor

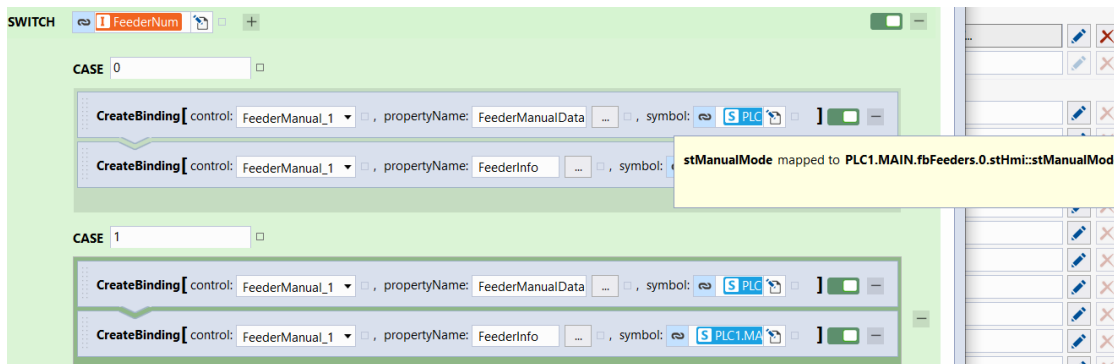


## Events for toggling feeder data

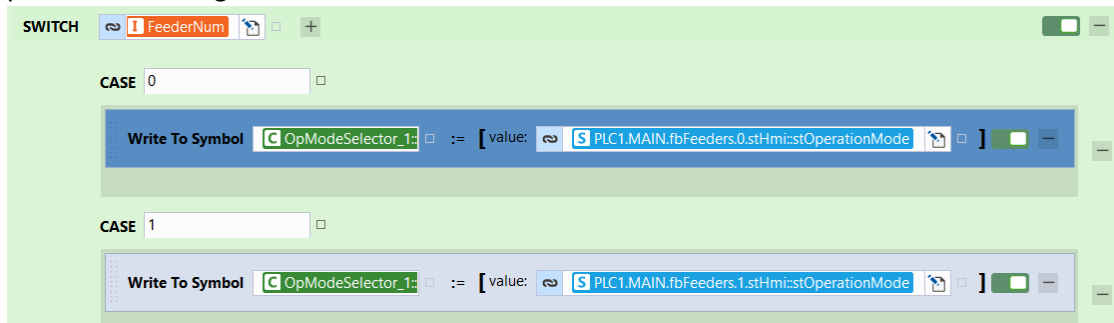
1. Open AutomaticMode.content and add custom event triggered by the internal symbol on this page. Configure actions to switch bindings from Feeder 1 to 2 or vice-versa by using the CreateBinding function:



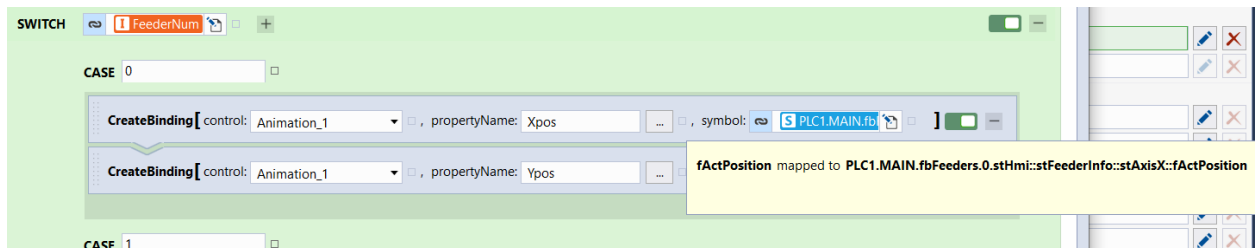
2. Add the same custom event on Manual.content and use CreateBinding to change 2 bindings:



3. Add the same custom event on OpMode UC instance and use CreateBinding to change its parameter binding:



4. Add the same custom event on Animation UC:



## Alarming and Messaging

1. Event Grid has been added for us on the Events page when we used project generator
2. Unique attributes of the event grid include showing menu bar, setting filters
3. On the runtime menu bar, we can see domains such as ADS, TcHmiSrv and more, but not “EventLogger”
5. We need to add the event logger server extension through Package management
6. This will show events from the EventLogger domain where:
  - a. Messages can be triggered by switching operation modes
  - b. Alarms are triggered by jogging with a high velocity

## Historization

1. Add Trendline control instance on the Trending.content page
2. Set ‘Start’ attribute to PT2M and ‘End’ to Latest
3. Add 2 y-axes (we need to ensure they don’t overlap so data can be seen clearly)
  - a. Position from 0-5000 with units

tlcHistorize | Y-Axis Definitions

Id	Axis Name
1	Position
2	Velocity

Properties

**Colors**

- Label font color: Theme
- Axis color: Theme
- Axis name font color: Theme

**General**

- Id: 1
- Position: Left
- Show axis: ☒
- Main-tick min-value: 0
- Main-tick max-value: 5000 \* L.F.ConvertUnits
- Auto scaling: ☐
- Logarithmic scale: ☐
- Axis labeling: Number
- Decimal places: 2
- Show labels: ☒
- Show axis name: ☒
- Axis name: Position
- Axis name font-family: [font family]
- Axis name font-size: 15
- Axis name font-size-u...: px
- Axis name font-weight: Bold
- Unit: U\_pos

OK Cancel

- b. Velocity from -2000-2000 with units

tlcHistorize | Y-Axis Definitions

Id	Axis Name
1	Position
2	Velocity

Properties

Colors

General

Id: 2

Position: Right

Show axis: ☒

Main-tick min-value:  $-2000 \cdot \frac{1}{F} \cdot \text{ConvertUnits}$

Main-tick max-value:  $2000 \cdot \frac{1}{F} \cdot \text{ConvertUnits}$

Auto scaling: ☐

Logarithmic scale: ☐

Axis labeling: Number

Decimal places: 2

Show labels: ☒

Show axis name: ☒

Axis name:  $\frac{1}{F} \cdot \text{Velocity}$

Axis name font-family:  $\frac{1}{F} \cdot$

Axis name font-size: 15

Axis name font-size-u...: px

Axis name font-weight: Bold

Unit:  $\frac{1}{F} \cdot \text{U.Velo}$

OK Cancel

4. Add 4 LineGraphDescriptions – set colors, symbol, y-axisID, Scale factor, Legend name

tlcHistorize | LineGraphDescriptions

Id	Y-Axis Name	Line Width
1	Position	1
1	Position	1
2	Velocity	1
2	Velocity	1

Properties

Colors

Line color: #FF4794DA

Point dot fill color: #FF4794DA

Point dot stroke color: #FF4794DA

Fill color: #FF4794DA

General

Symbol: PLC1.MAIN.fbFeeders.0.fHmiPosXHistoriz

Y-Axis Id: Position

LineWidth: 1

Scale factor:  $\frac{1}{F} \cdot \text{F} \cdot \text{ConvertUnits}$

Legend name:  $\frac{1}{F} \cdot \text{Position X}$

Point dot: ☐

Point dot radius: 3

Point dot stroke width: 1

Point dot in stopmode: ☒

FillMode: None

FillTransparency: 50

OK Cancel

5. There are no historization symbols to pick from. We need to add historize server extension from NuGet package management.
6. Map the following symbols and historize:

TwinCAT HMI Configuration

Name	Datatype	Interval	Max Entries	Recording
PLC1.MAIN.fbFeeder1.fHmiPosXHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder1.fHmiPosYHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder1.fHmiVeloXHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder1.fHmiVeloYHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder2.fHmiPosXHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder2.fHmiPosYHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder2.fHmiVeloXHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>
PLC1.MAIN.fbFeeder2.fHmiVeloYHistorize	LREAL	PT1S	10000	<input checked="" type="checkbox"/>

7. Historizing with interval of 1s will show us “unsmooth” data. Change historize settings to 10ms:
8. Add trigger condition for Feeder number and change LineGraphDescriptions:









The screenshot shows the TwinCAT HMI Configuration interface with several windows open:

- Data Definitions:** A table with columns 'Id', 'Y-Axis Name', and 'Line Width'. It contains two rows of generated data.
- Properties:** A window showing properties for the selected data. The 'Line color' is set to #FF6AECF5. The 'General' section shows 'Symbol' as 'PLC1.MAIN.fbFeeder1.fHmiPosXHistorize' and 'Y-Axis Id' as '1'.
- Properties:** A window showing properties for the selected data. The 'Custom' section shows 'FeederNum' as 'Configured...'.
- Properties:** A window showing properties for the selected data. The 'Write To Symbol' section shows 'TicHistorizeLineGra' as 'value: (4 items)'.

## Recipe Management



1. Add Recipe Management Server extension
2. Add two recipe types – Feeder1 and Feeder2
3. Select Feeder1 Recipe symbols and rename display names

Recipes.content Feeder1 Desktop.view

Display Name	Symbol	Default	Min	Max	Unit	Enabled
Target1XPos	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::fTarget1PositionX	500	0	1000	mm	<input checked="" type="checkbox"/>
Target1YPos	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::fTarget1PositionY	500	0	1000	mm	<input checked="" type="checkbox"/>
Target2XPos	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::fTarget2PositionX	1000	0	1000	mm	<input checked="" type="checkbox"/>
Target2YPos	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::fTarget2PositionY	1000	0	1000	mm	<input checked="" type="checkbox"/>
Velocity	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::fVelocity	50	0	100	%	<input checked="" type="checkbox"/>
RecipeName	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::sRecipeName	Default				<input checked="" type="checkbox"/>
Target1Name	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::sTarget1Name	Target 1				<input checked="" type="checkbox"/>
Target2Name	 PLC1.MAIN.fbFeeders[0]::stHmi::stFeederRecipe::sTarget2Name	Target 2				<input checked="" type="checkbox"/>

4. Repeat steps for Feeder2
5. Add 2 folders under Recipes and add one Default recipe for each FeederType:











Enter a search term...

Name	Based on Recipe Type
Feeder1	
 DefaultFeeder1	Feeder1
Feeder2	
 DefaultFeeder2	Feeder2

6. Add Recipe Edit control on Recipe page:

Toolbox

Search Toolbox

	Linear Gauge
	Localization Select
	Numeric Input
	Object Browser
	Password Input
	Radial Gauge
	Radio Button
	Recipe Edit
	Recipe Select
	Spinbox Input

This control allows to modify, add, delete and activate recipes on runtime

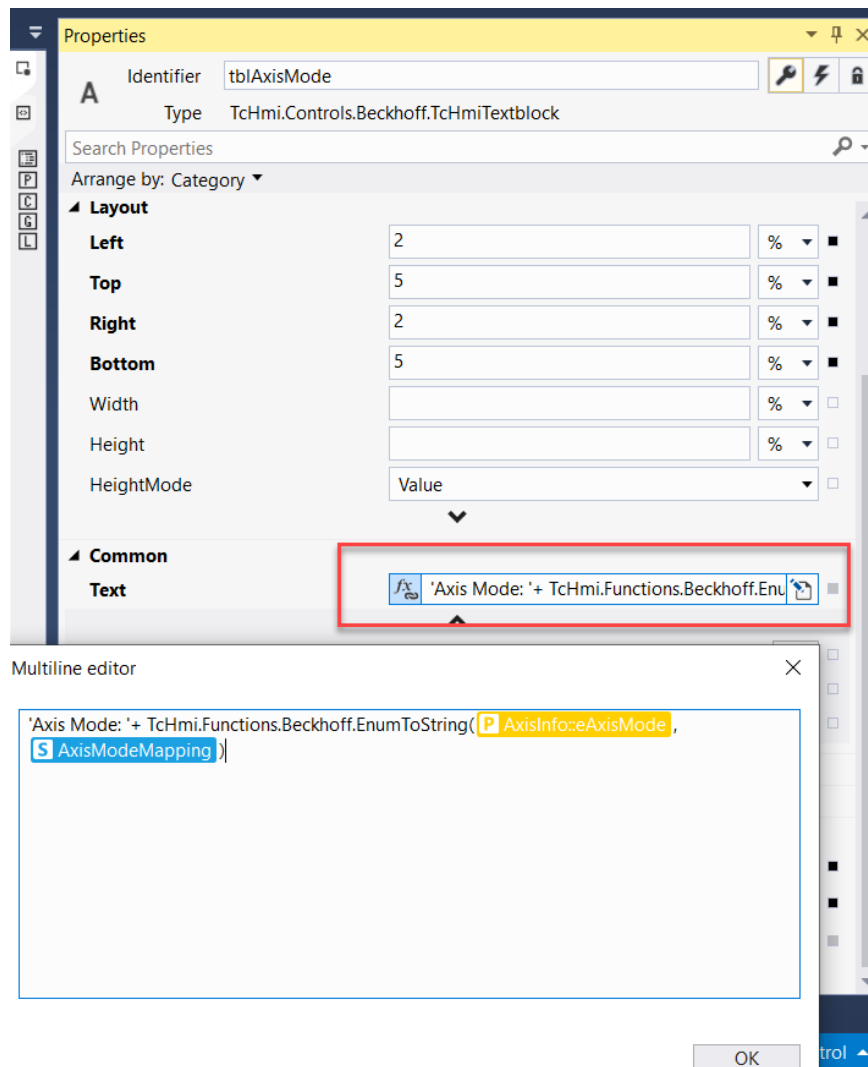
## Enum Descriptions

TcHmi offers a function to convert Enum numbers to strings in the HMI. For this, we need to create a "Mapping variable"

1. Create server symbol 'AxisModeMapping' of type EnumMapping with Default value:

```
{  
  0: "INIT",  
  1: "ENABLE",  
  2: "JOG",  
  3: "POSITIONING",  
  4: "STOP",  
  5: "RESET",  
  6: "ERROR"  
}
```

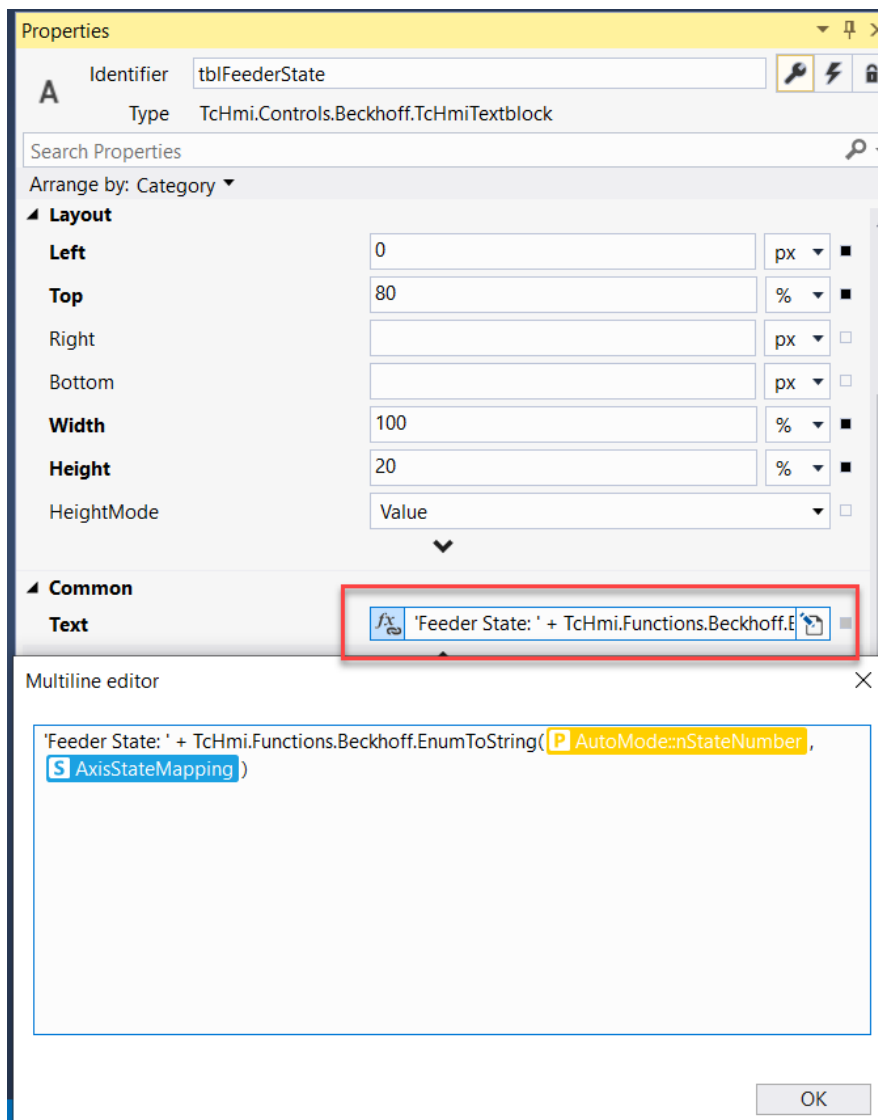
Bind Axis mode textblock in Automatic base UC:



2. Create server symbol 'AutoStateMapping' of type EnumMapping with Default value:

```
{
  0: "Reset",
  1: "MoveBackToTop",
  2: "MoveToTarget1PosX",
  3: "MoveToTarget1PosY",
  4: "Grab",
  5: "MoveBackToTop",
  6: "MoveToTarget2PosX",
  7: "MoveToTarget2PosY",
  8: "Place"
}
```

Bind FeederAutomatic UC:

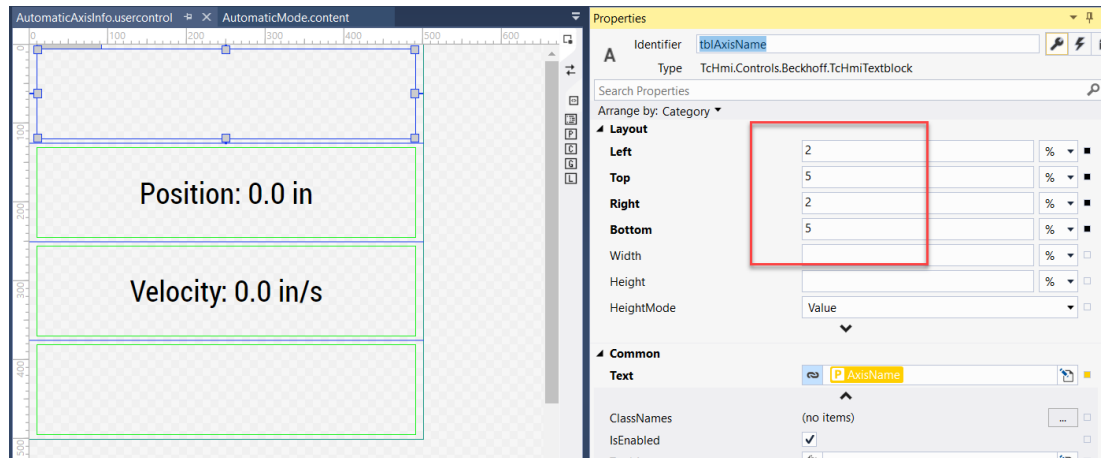




## Appendix – Build User Controls (UCs)

### Create Automatic Axis Info UC

1. Grid centered inside UC host with 4 rows
2. Add textblock inside each UC – rename each textblock to AxisName, Position, Velocity, AxisMode:



3. Create 2 parameters for UC – AxisName (String), AxisInfo (PLC.ST\_HmiAxisInfo)

#### Edit/Define Parameters

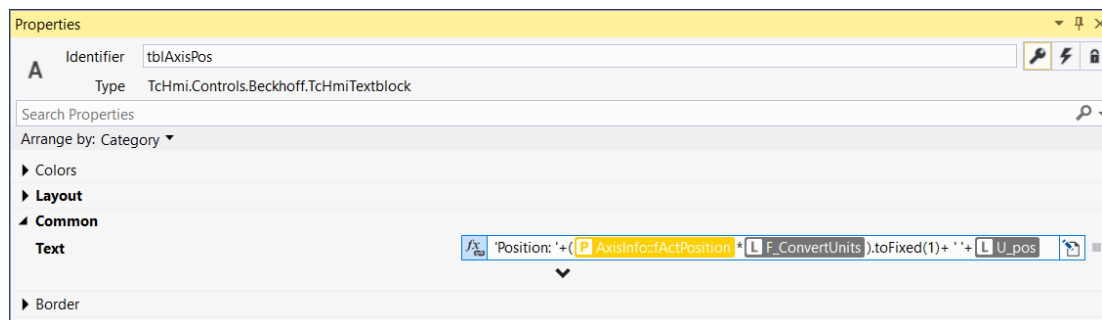
Description:

**Parameters:**

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
AxisName	G String			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AxisInfo	S PLC1.ST_HmiA	(Object)	(Object)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

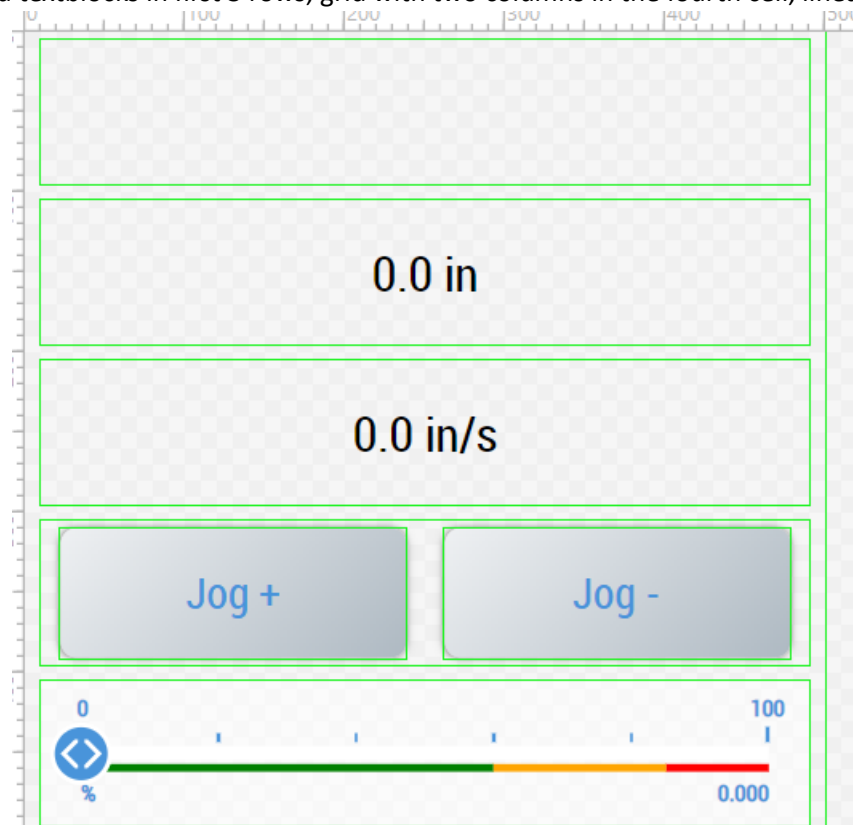
tchmi:server#/definitions/PLC1.ST\_HmiAxisInfo

4.
  - a. Bind Axis Name parameter to “Text” attribute.
  - b. Create bindings of sub-variables from AxisInfo parameter with other textblocks in the UC host. Make it a function binding and add raw text, F\_ConvertUnits, U\_Pos, .toFixed(1) for showing only one decimal point:



## Create Manual Axis UC

1. Grid centered inside UC host with 5 rows
2. Add textblocks in first 3 rows, grid with two columns in the fourth cell, linear gauge in 5<sup>th</sup> cell:



3. Create 6 parameters for UC – JogPlus, JogMinus, PositionFeedback, VelocityFeedback, VelocityCmd, AxisName:

Edit/Define Parameters




Description:

Parameters:


Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility	
AxisName	G String			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
JogPlus	G BOOL	False ▼	False ▼	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
JogMinus	G BOOL	False ▼	False ▼	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
VelocityCmd	G Number			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
PositionFeedback	G Number			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
VelocityFeedback	G Number			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

4. Bind from parameters to Axis Name, Axis ActPos (use localization symbol), Axis ActVelo 'Text' attribute
5. Bind from parameters to Jog + and – state symbol attribute
6. Create two-way binding with LinearGauge "value" and set range for display:

Properties

Identifier    

Type TcHmi.Controls.Beckhoff.TcHmiLinearGauge

Search Properties 

Arrange by: Category ▾

BorderColor  Ineme

Layout

Left  % ▾ ■

Top  % ▾ ■




Right  % ▾ ■

Bottom  % ▾ ■

Width  % ▾ □

Height  % ▾ □

Common

Value    

MinValue  □

MaxValue  ■

SetPoint  □

Editable ☒ ■

IgnoreInvalidValues ☐ □


IgSetVelocity | Range


Elements

Start	End	Color
0	60	#FF008000
60	85	#FFFFA500
85	100	#FFFF0000

Properties

Colors

Color  



R 0  
G 128  
B 0  
A 255

#FF008000




General

Start  □


End  □

OK Cancel

Properties

Identifier    

Type TcHmi.Controls.Beckhoff.TcHmiLinearGauge

Search Properties 

Arrange by: Category ▾


Editable ☒ ■


IgnoreInvalidValues ☐ □

Border

Background Image

Value

Step   ■

Range  ■

ShowValueText ☒ ▾

Labels

ShowLabels ☒ ■


LabelRange  ■

LabelPosition  ▾ □

Ticks

ShowTicks ☒ ▾

Unit

Unit   ■

UnitOrientation  ▾ □

## Create FeederAutomatic UC

### 1. Add 2 parameters:

Edit/Define Parameters

Description:						
<b>Parameters:</b>						
Name	Datatype	(Drop) Default Value	Default Value Internal	Bindable	Visibility	
FeederAutoData	S PLC1.ST_HmiA	(Object) ...	(Object) ...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
FeederInfo	S PLC1.ST_HmiF	(Object) ...	(Object) ...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

### 2. Add two instances of automatic axis Info UC and map their "parameter attributes" to X and Y AxisInfo parameters from parent UC:

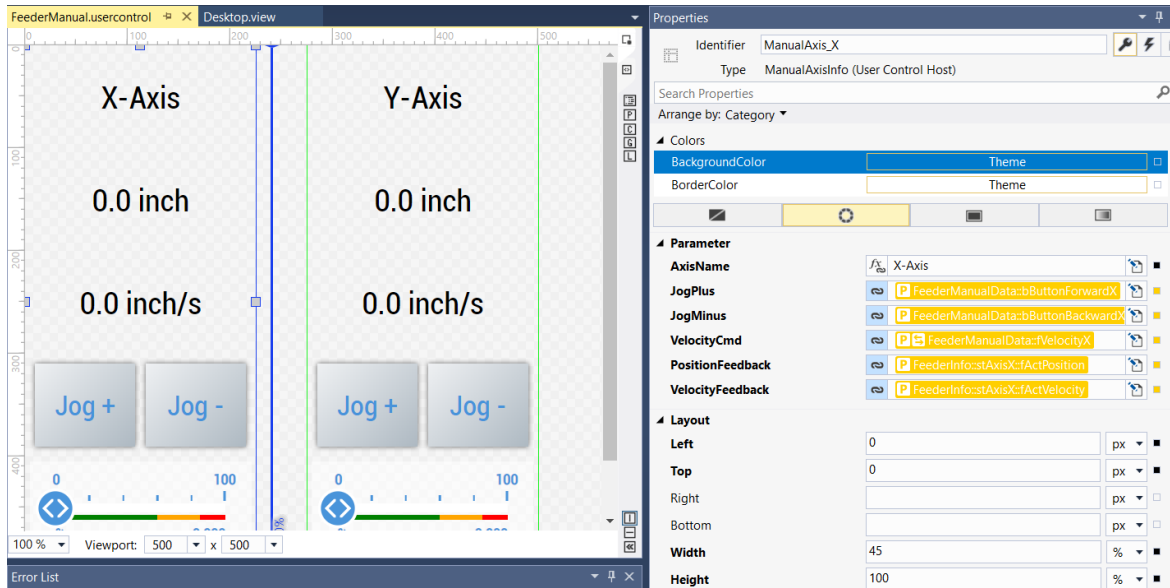
The screenshot shows the Visual Studio IDE with the 'FeederAutomatic.usercontrol' open. The design area displays two side-by-side boxes labeled 'X-Axis' and 'Y-Axis'. Each box contains text for 'Position: 0.0 inch', 'Velocity: 0.0 inch/s', and 'Axis Mode: 0'. Below these boxes is a label 'Feeder State: 0'. The Properties window on the right shows the 'AxisInfo\_X' property being set to 'FeederInfo::stAxisX'.

### 3. Add textblock in bottom 20% of space:

The screenshot shows the Visual Studio IDE with the 'FeederAutomatic.usercontrol' open. The design area displays a single box labeled 'Feeder State: 0'. The Properties window on the right shows the 'Text' property being set to 'Feeder State: ' + FeederAutoData::nStateNumber'.

## Create FeederManual UC

1. Add two instances of manual axis UC

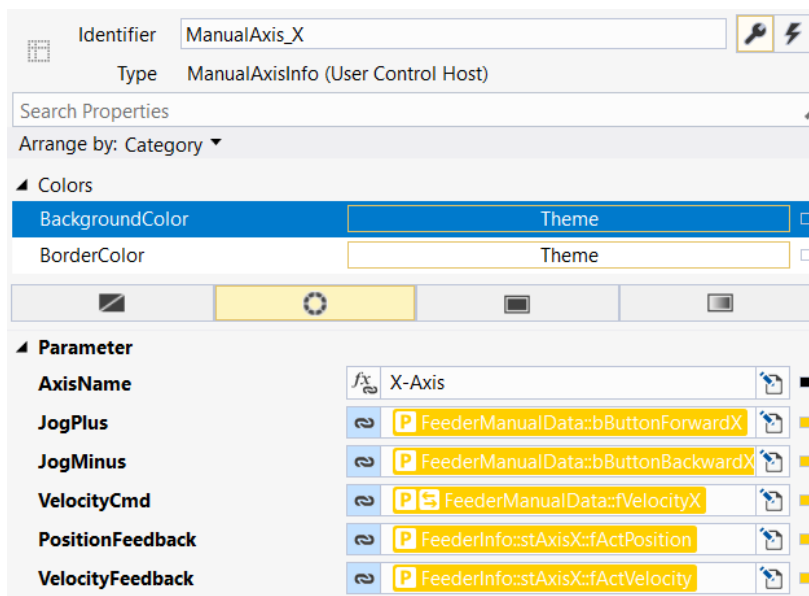


2. Add 2 parameters – FeederManualData (ST\_HmiManualMode), FeederInfo (ST\_HmiFeederInfo)

Edit/Define Parameters

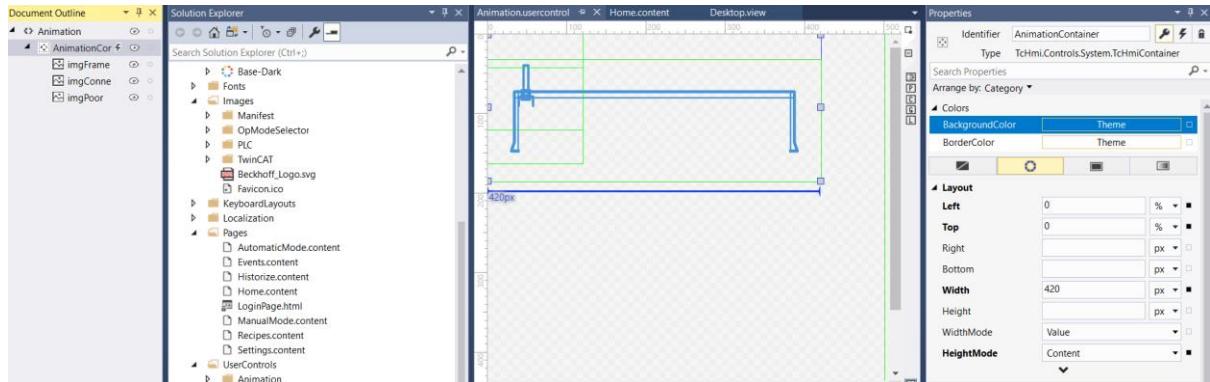
Description:						
<b>Parameters:</b>						
Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility	
FeederManualData	S PLC1.ST_HmiManualMode	(Object) ...	(Object) ...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
FeederInfo	S PLC1.ST_HmiFeederInfo	(Object) ...	(Object) ...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

3. Bind nested UC parameters to parent UC



## Create Animation UC

1. Add folder under UserControls and add images from the "Images" folder in the repo
2. Add container with the following Layout settings:



3. Create UC parameters:

Edit/Define Parameters

Description:

**Parameters:**

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
Xpos	G Number			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Ypos	G Number			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

4. Add frame, connection, poor images with the following layout:

Identifier:

Type:

Search Properties

Arrange by: Category

BorderColor:

**Layout**

**Left:**  px

**Top:**  px

**Right:**  px

**Bottom:**  px



**Width:**  px

**Height:**  px


**WidthMode:**

**HeightMode:**

Properties





Identifier   

Type TcHmi.Controls.Beckhoff.TcHmImage




Search Properties 


Arrange by: Category ▾


BorderColor  Theme


   


Layout


Left    


Top  px ▾ 


Right  px ▾ 

Bottom  px ▾ 




Width  px ▾ 

Height  px ▾ 


WidthMode  ▾ 

HeightMode  ▾ 

Properties





Identifier    

Type TcHmi.Controls.Beckhoff.TcHmImage




Search Properties 




Arrange by: Category ▾


BorderColor  Theme


   


Layout


Left    

Top    

Right  px ▾ 

Bottom  px ▾ 

Width  px ▾ 

Height  px ▾ 

WidthMode  ▾ 