Contents

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
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

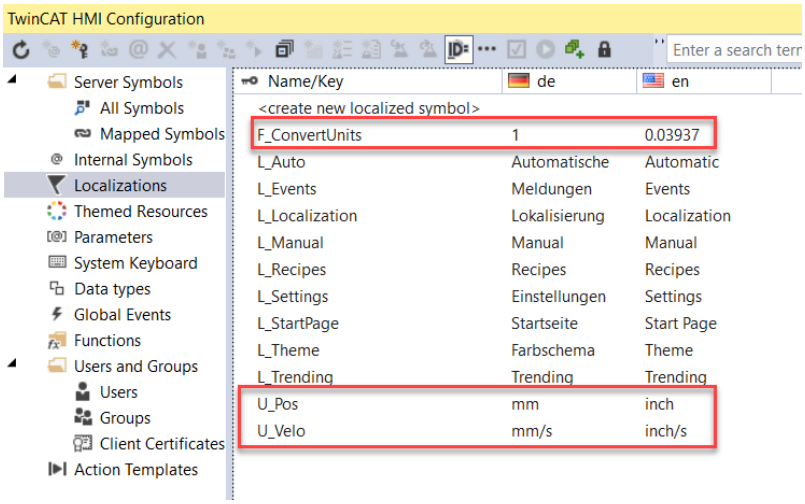
Home Page

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 from this image from the solution tree into the Designer Home.content
4. Rename image and change layout to make it fit entire content page

Localization Variables

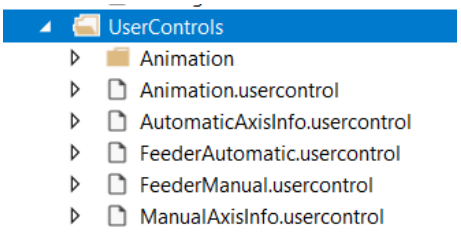
Add the following localization variables (Ensure case matches the case in the picture otherwise pre-done control bindings will not work):

1. U_Position
2. U_Velocity
3. F_ConvertUnits (1 (de) : 0.3937 (en)).

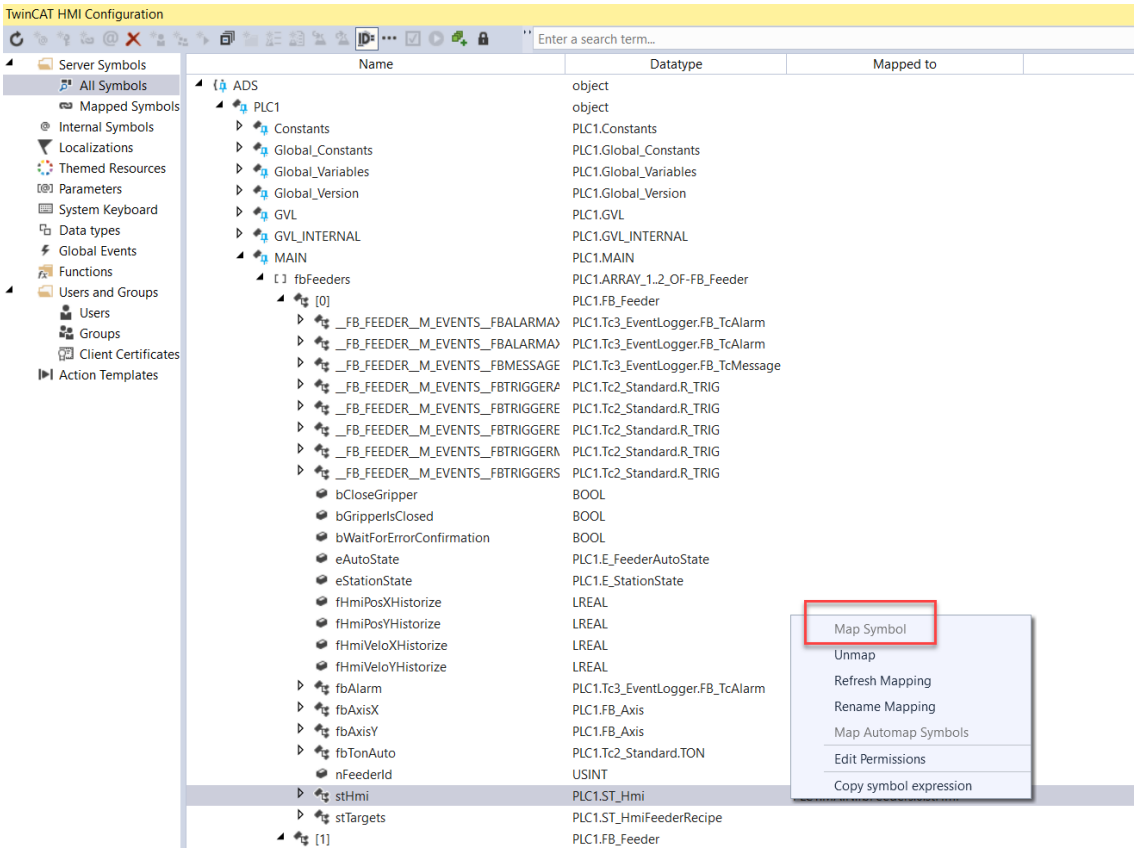


Bring User Controls into the project

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”

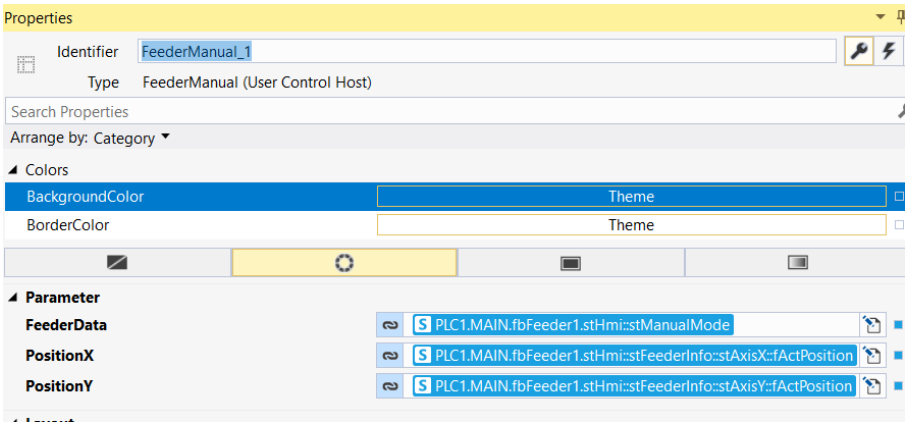


3. Map stHmi variables for both feeders from the PLC. This will make PLC datatypes available inside the HMI project. UC parameters use these datatypes:

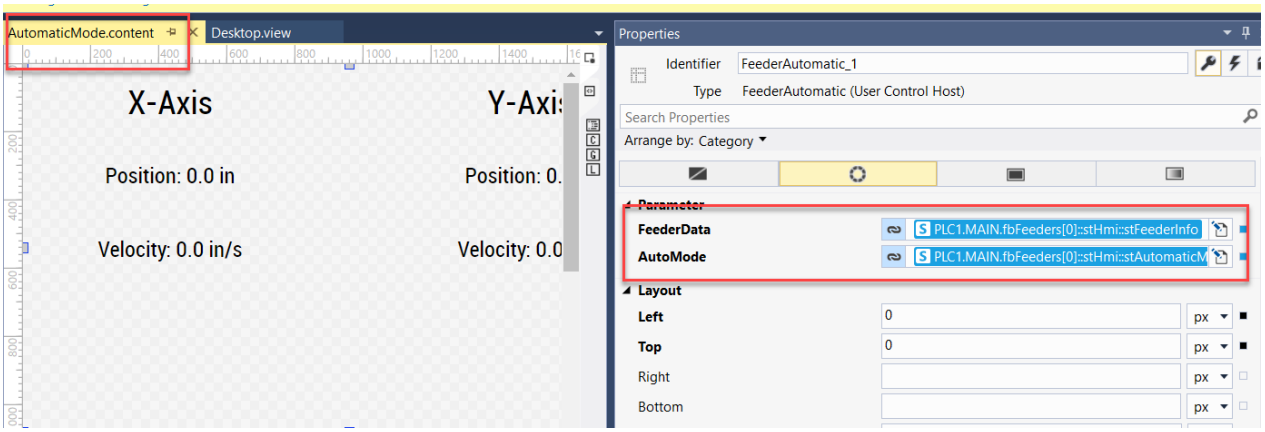


Populate Manual and Automatic Mode contents

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

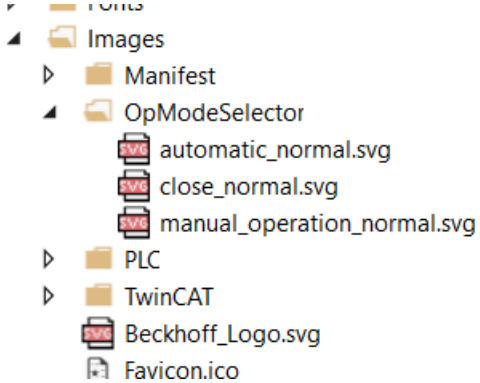


2. Add one instance of FeederAutomatic UC on AutoMode content and bind to PLC symbols:

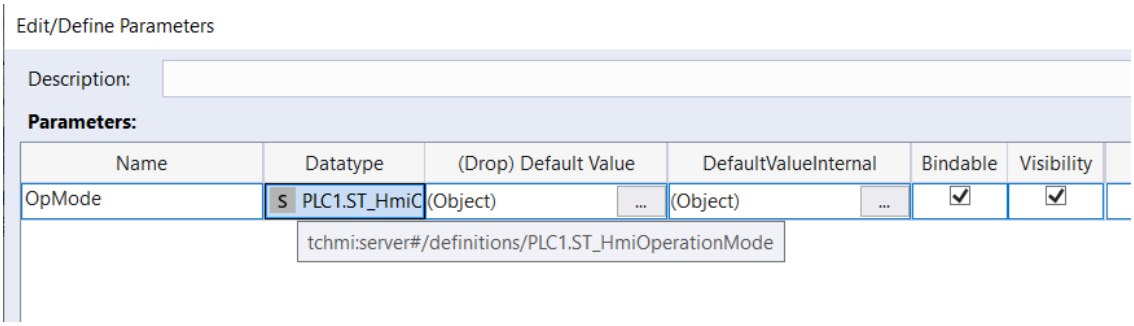


Operation Mode User Control

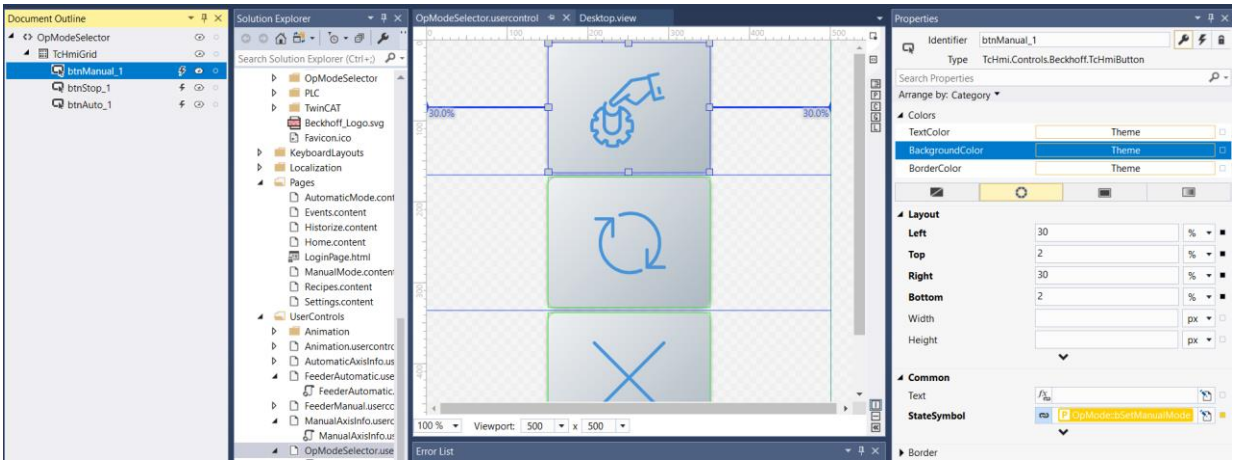
1. Create a new UC and add “Grid” control with 3 rows
2. Add one button each to all rows with 2% spacing at Top and Bottom, 30% spacing at Left and Right
3. Add folder under Images and add images from the Images folder in the repo:



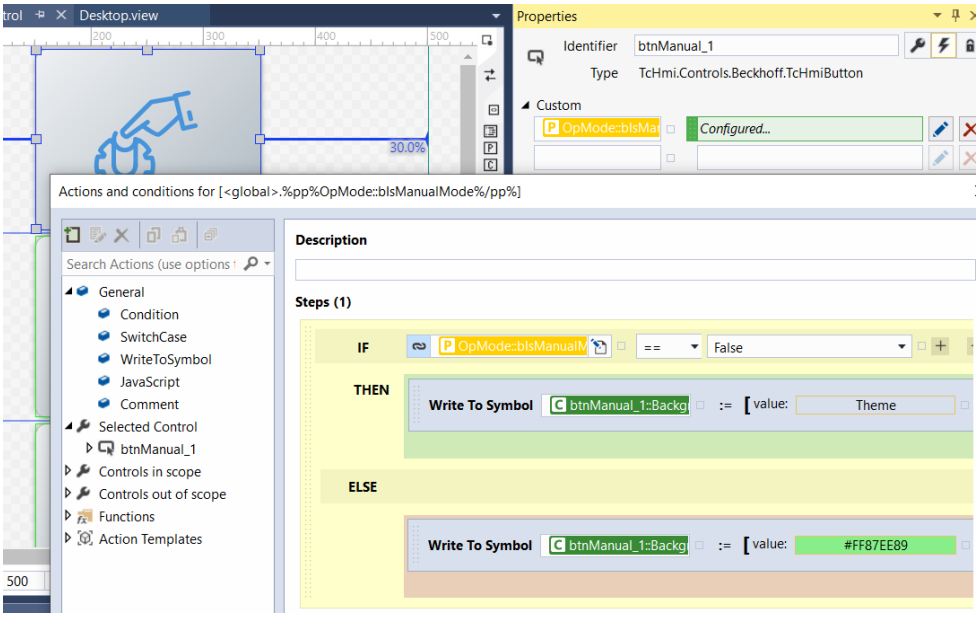
4. Add these images as icons to buttons with 100% width and height
5. Add parameter of type feeders[0].stHmi.OperationMode to the UC



6. Map ‘State symbol’ of each button to parameters bSetAutoMode, bSetManualMode, bSetStopMode respectively

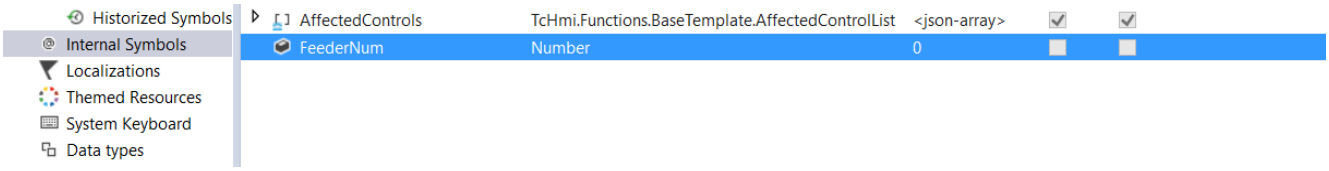


7. Add custom trigger conditions for each button for example: ‘isAutoMode’ -> if ‘isAutoMode TRUE then background color Green ELSE theme’



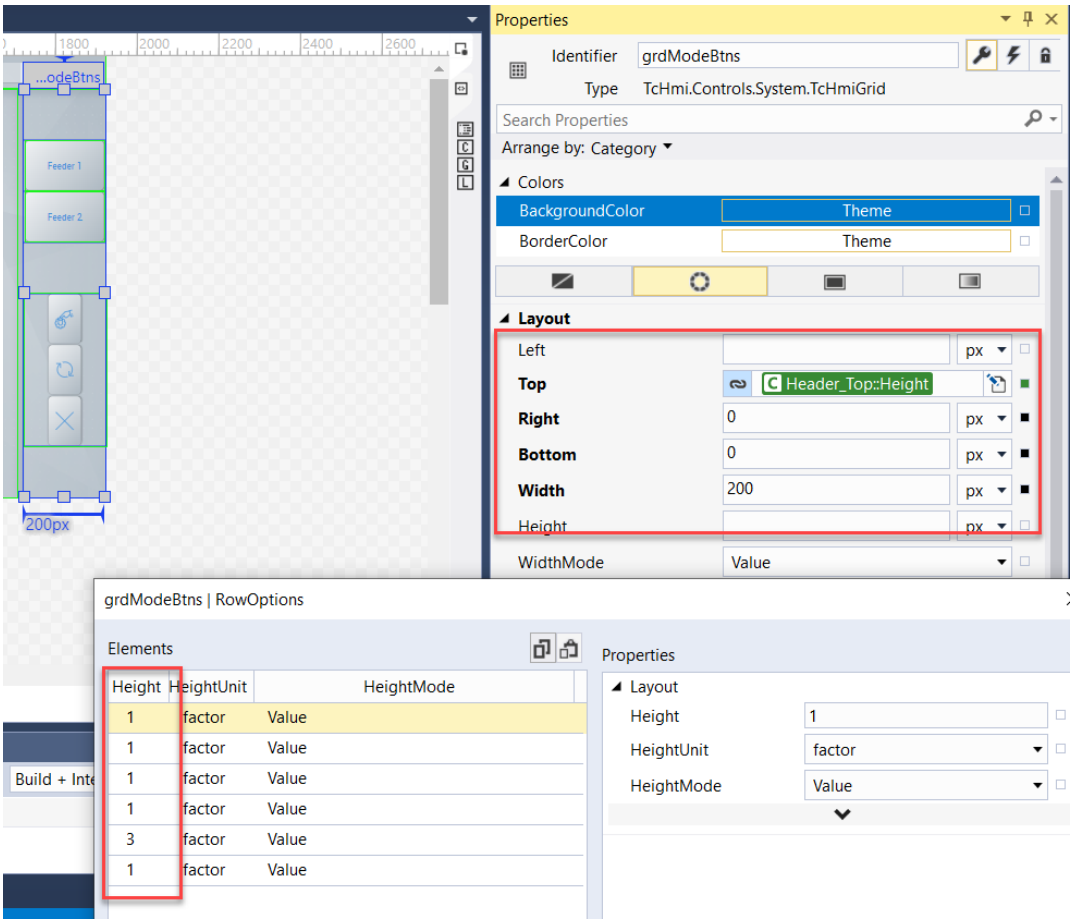
Internal symbol for Feeder selection

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

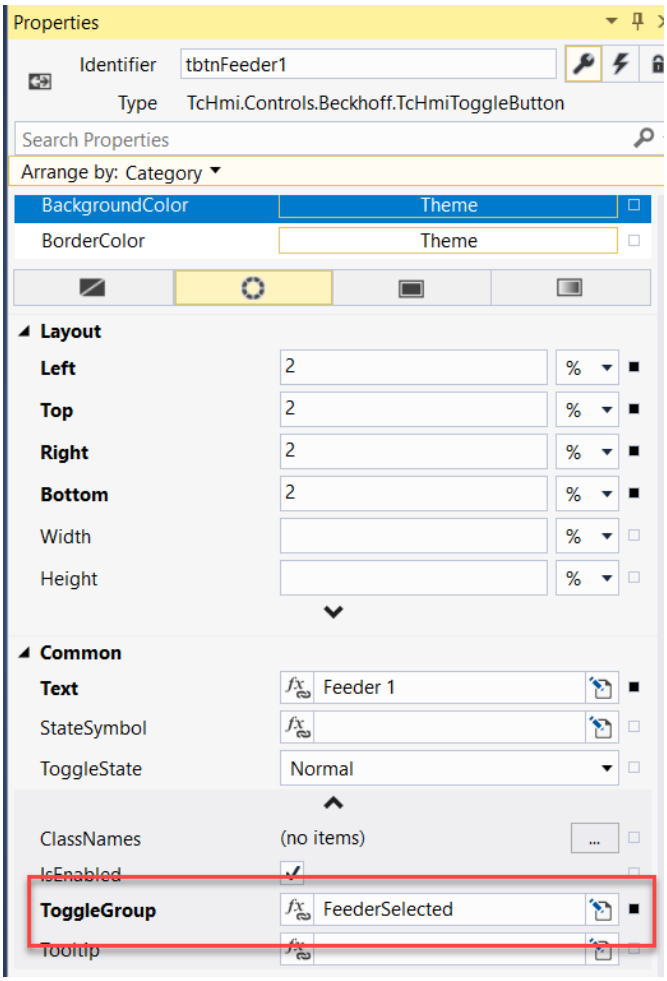


Feeder and Operation Mode Menu

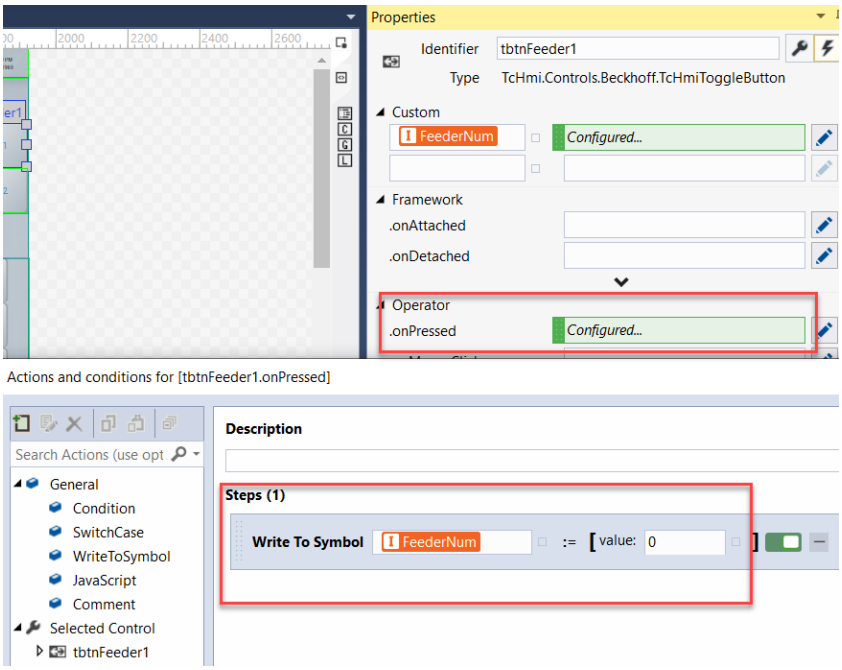
1. Add Grid on the right side of Desktop.view. Bind ‘Top’ with Height of header bar, ‘Right’ 0, ‘Bottom’ 0, ‘Width’ 200px
2. Change Right of Region_Center to Width of grdModeBtns
3. Add 6 rows – all cells with factor 1, fifth cell with factor 3



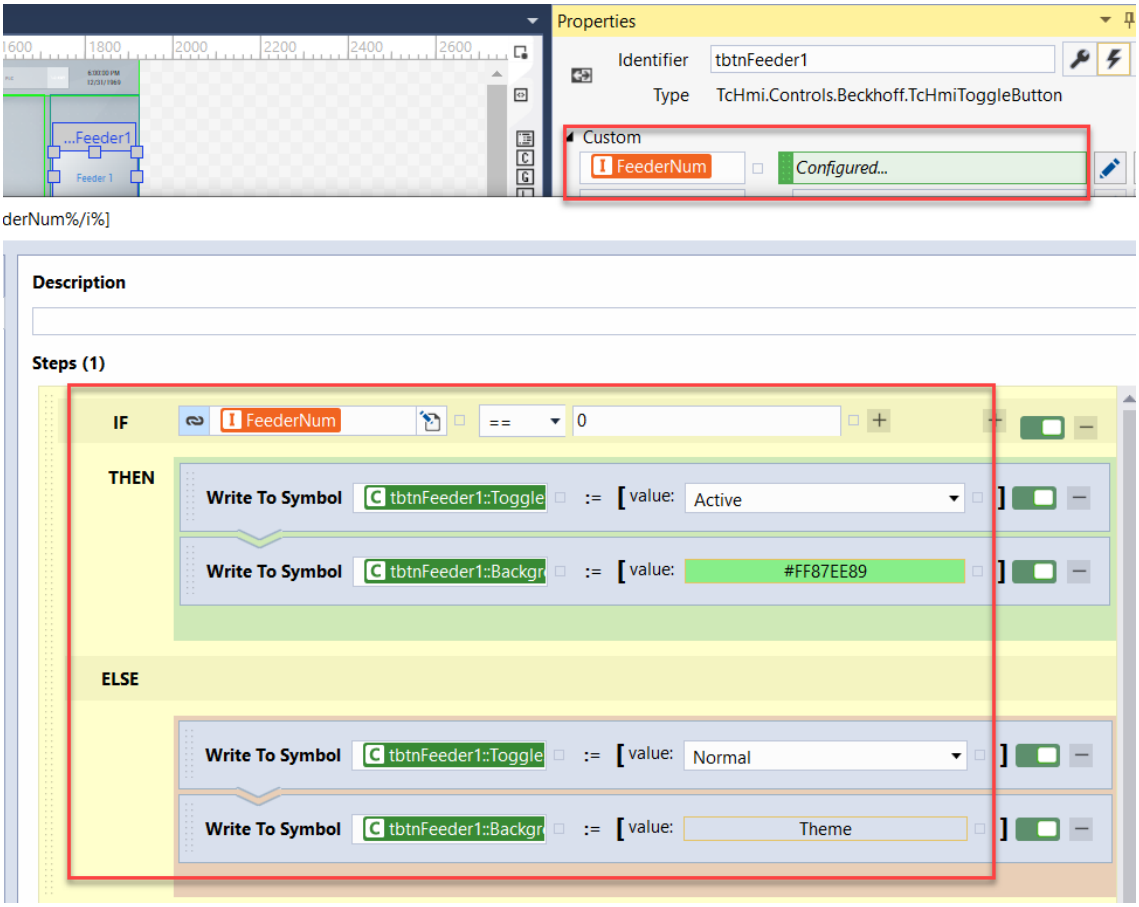
4. Add OpMode UC instance and map to Feeder 1 variable from the HMI
5. Add toggle buttons in second and third cells of the grid – 2% spacing from Left Right Top Bottom
6. Add ‘ToggleGroup’ for each button, which will ensure only one button from the group is toggled on at a time



7. Write to internal symbol at ‘onPressed’ event of each toggle button

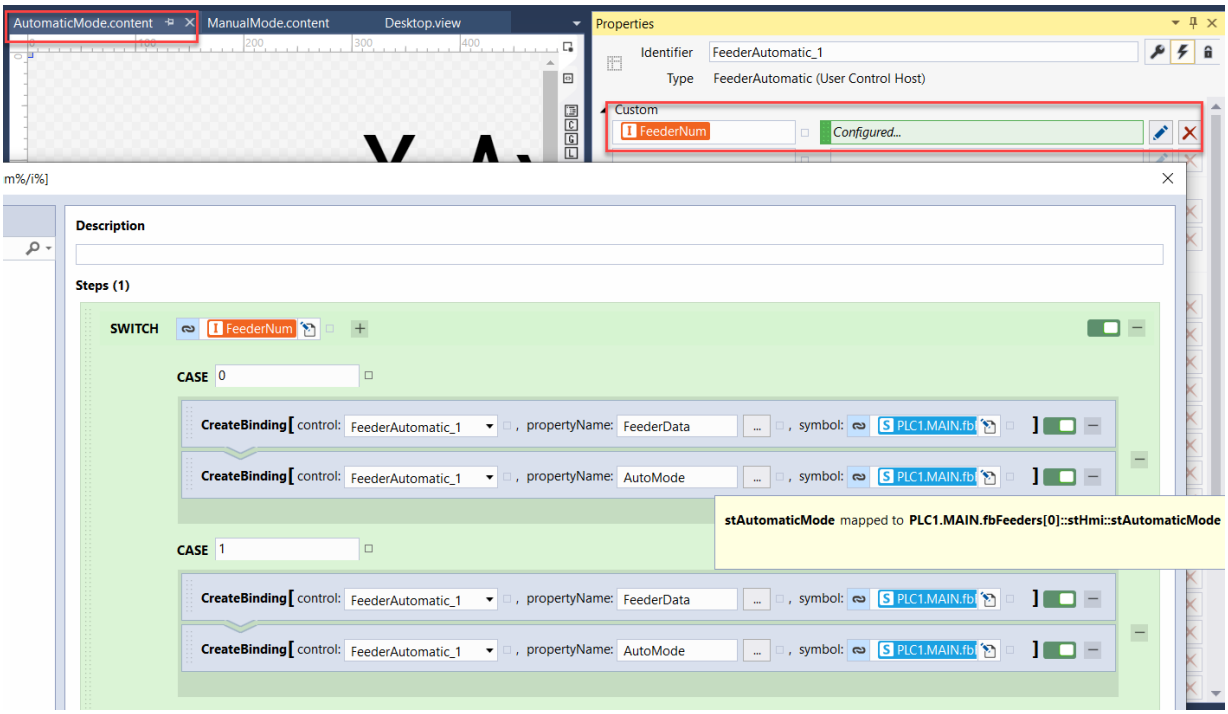


8. Add custom trigger condition with internal symbol to change toggleState and BackgroundColor

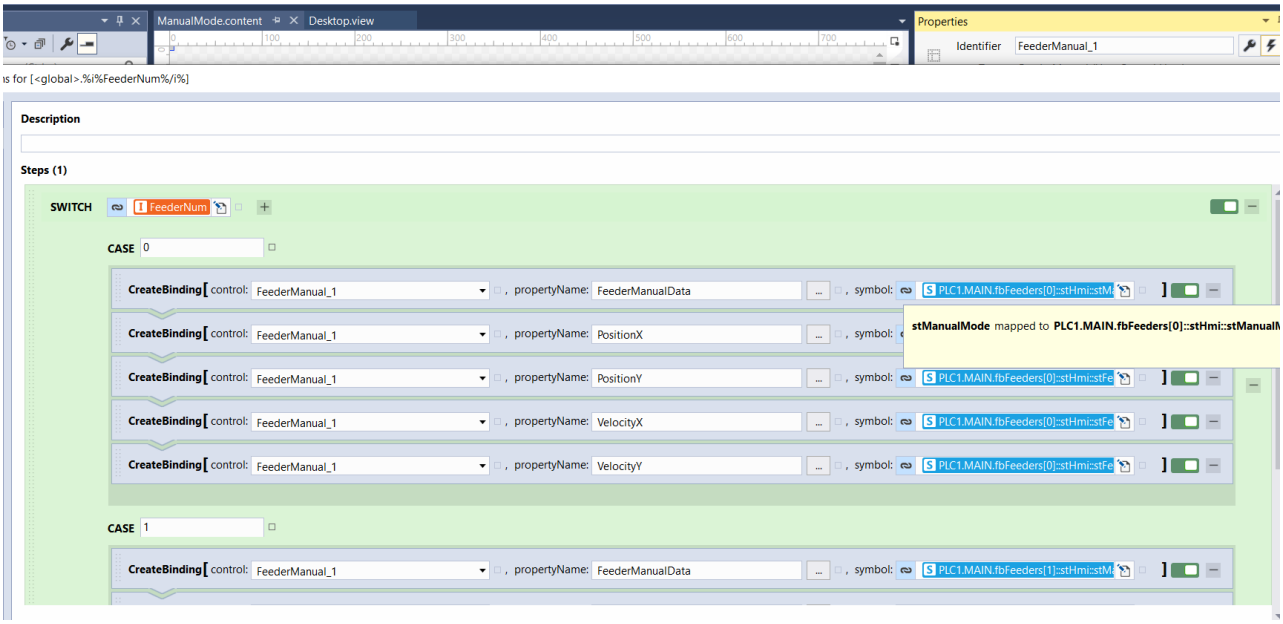


Events for toggling feeder data

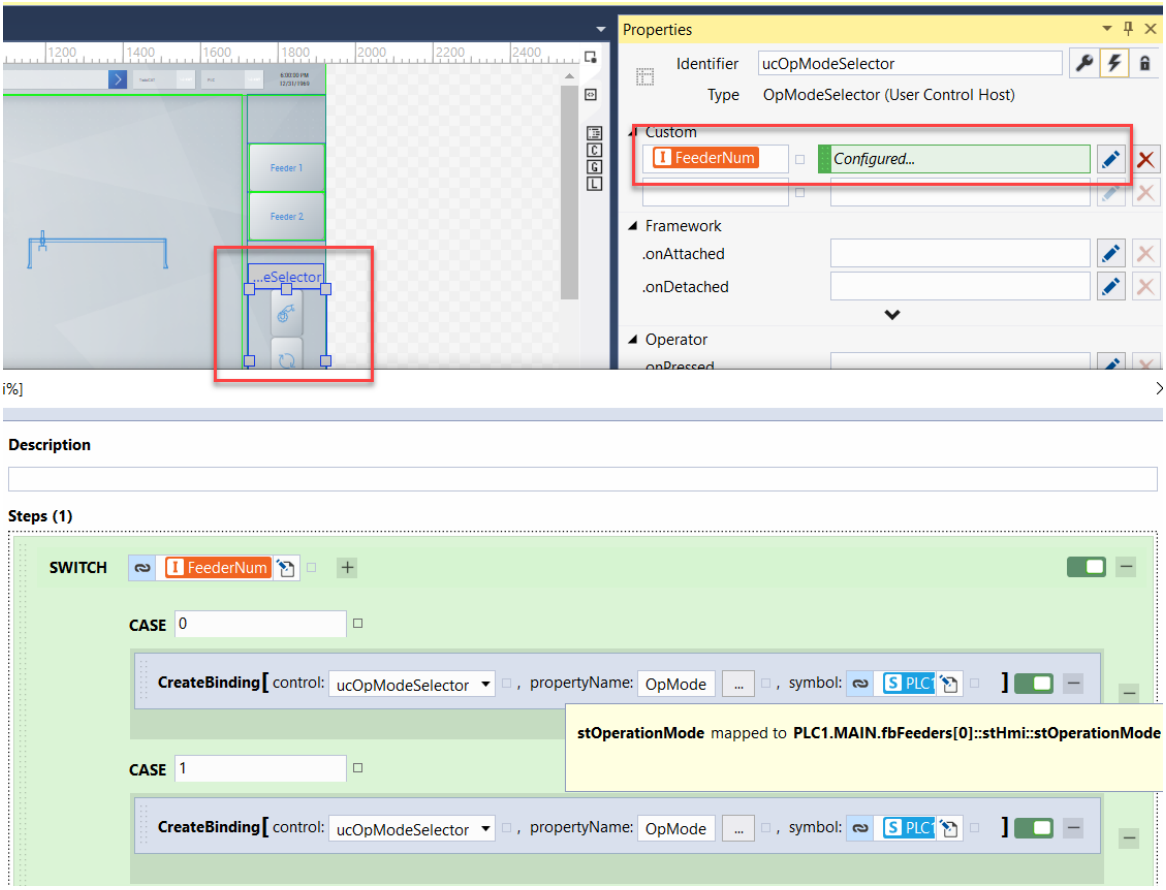
1. Open AutomaticMode.content and add custom event triggered by the internal symbol on this page. Actions will be to switch bindings from Feeder 1 to 2 or vice-versa, which is done by using the CreateBinding function:



2. Add the same custom event on Manual.content and use CreateBinding to change 5 bindings:



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

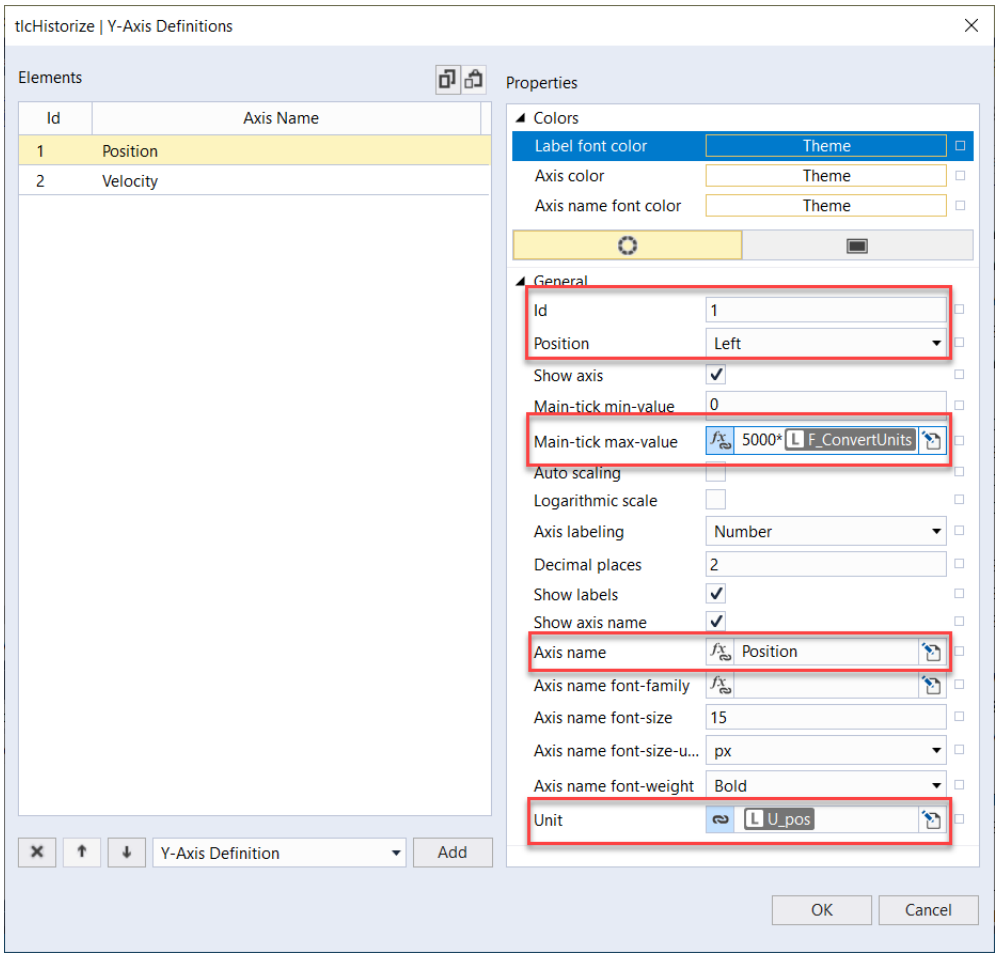

Properties
Identifier
ucOpModeSelector
Type
OpModeSelector (User Control Host)
Custom
FeederNum
Configured...
Framework
.onAttached
.onDetached
Operator
Feeder 1
Feeder 2
Description
Steps (1)
SWITCH
FeederNum
CASE
0
CreateBinding[control: ucOpModeSelector , propertyName: OpMode ... , symbol: PLC]
stOperationMode mapped to PLC1.MAIN.fbFeeders[0]::stHmi::stOperationMode
CASE
1
CreateBinding[control: ucOpModeSelector , propertyName: OpMode ... , symbol: PLC]

Alarming and Messaging

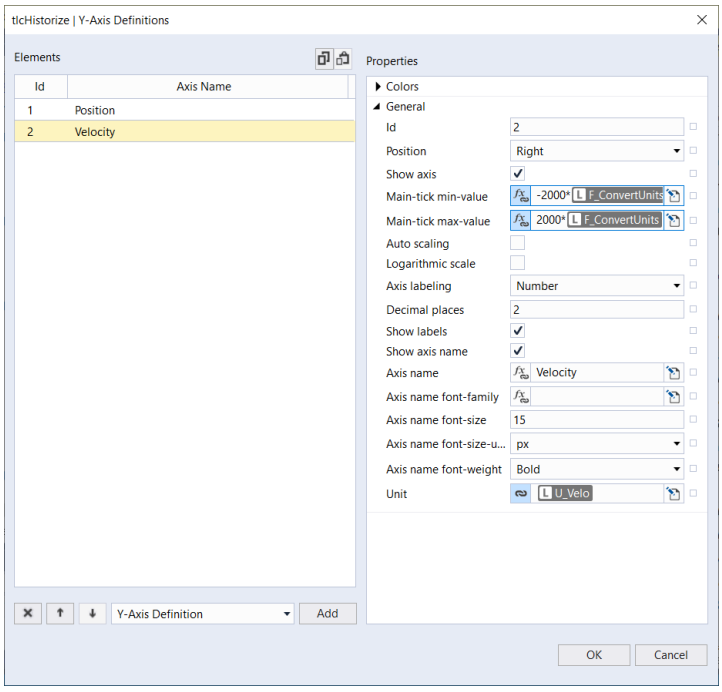
1. Event Grid has been added for us on the Events page
2. Attributes of the grid include showing menu bar, setting filters
3. On the runtime menu bar, domains include ADS, TcHmiSrv and more, but not “EventLogger”
4. Add event logger server extension
5. This will show events from the EventLogger domain where:
    a. Messages can be triggered by switching operation modes
    b. Alarms are triggered by jogging with a high velocity (which can be set on the PLC)

Historization

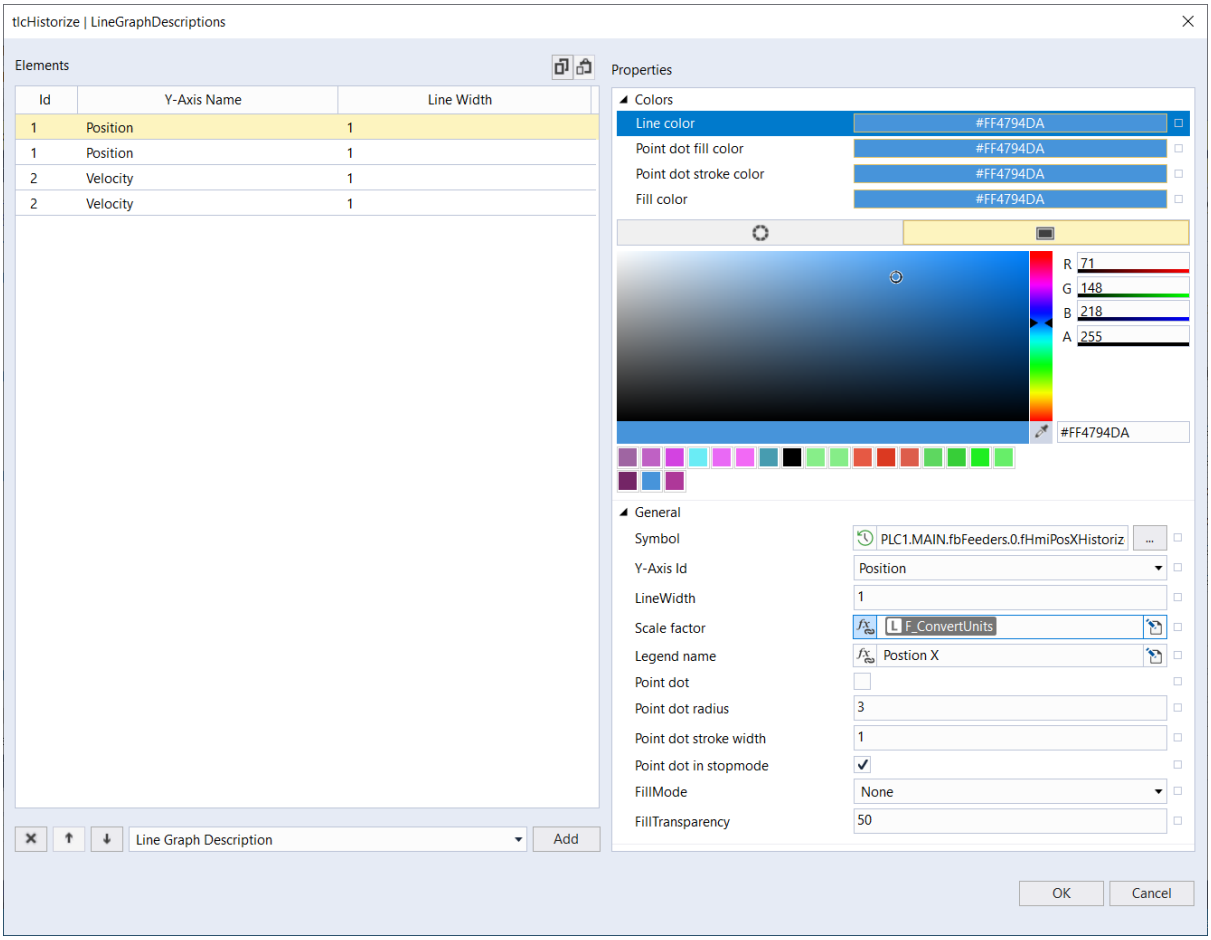
1. Add Trendline control instance on the Trending.content page
2. Set ‘Start’ attribute to PT2M and ‘End’ to Latest
3. Add 2 y-axes (we need to ensure they don’t overlap so data can be seen clearly)
    a. Position from 0-5000 with units



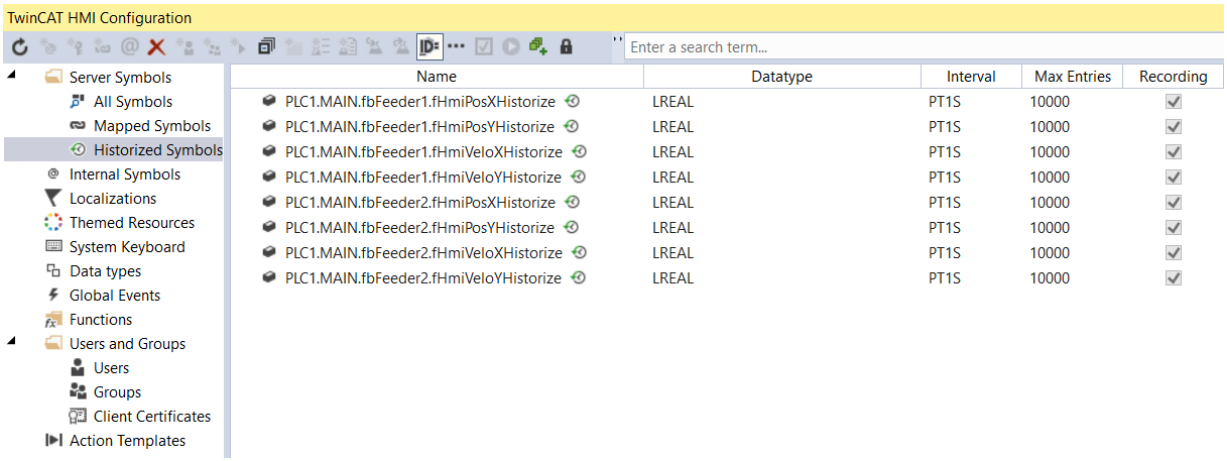
    b. Velocity from -2000-2000 with units



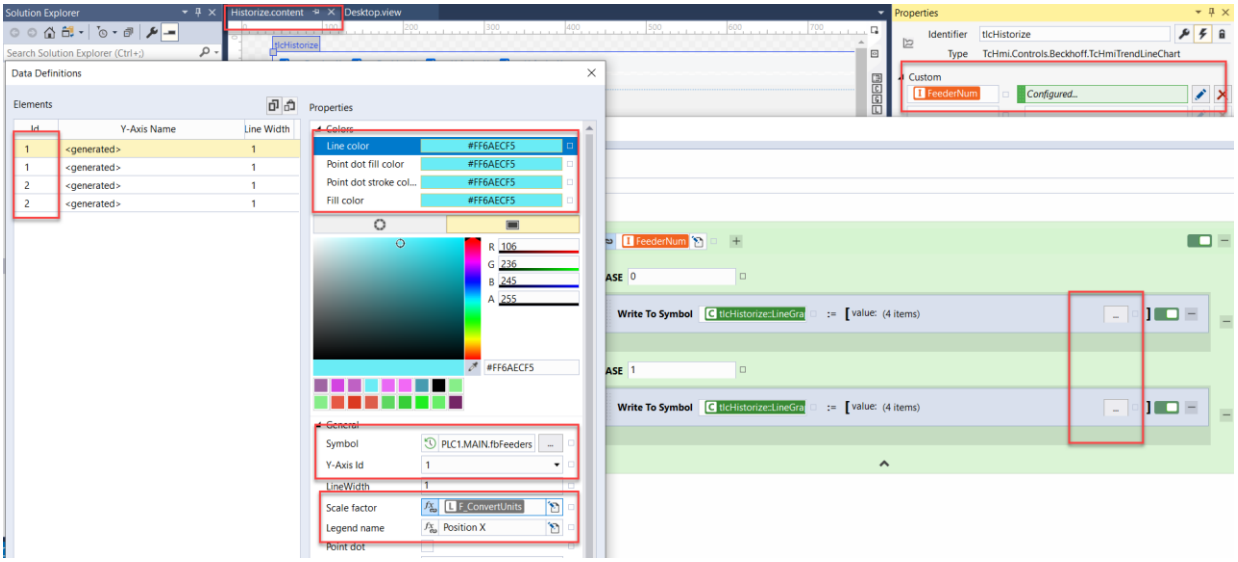
4. Add 4 LineGraphDescriptions – set colors, symbol, y-axisID, Scale factor, Legend name



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:

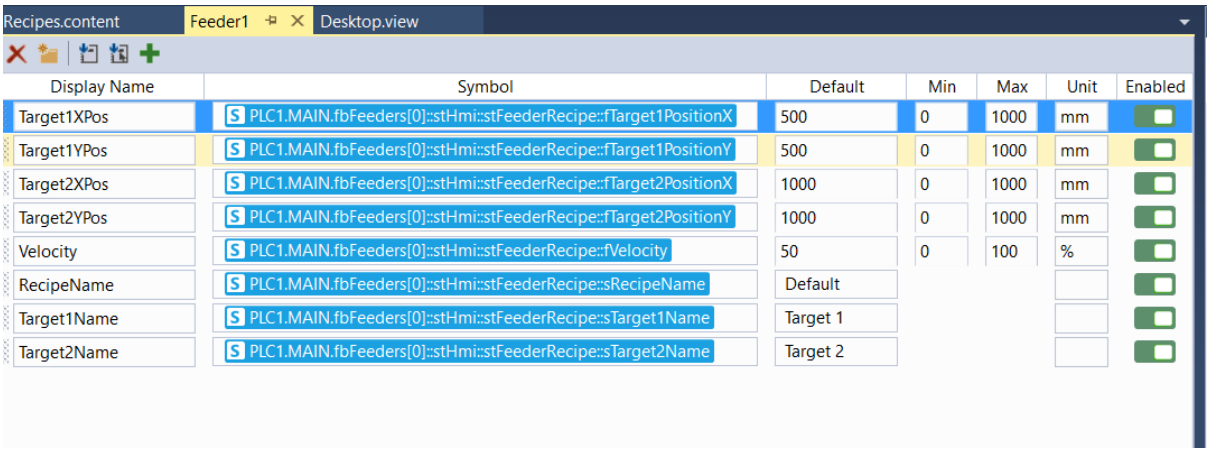


7. Historizing with interval of 1s will show us “unsmooth” data. Change historize settings to 10ms:
8. Add trigger condition for Feeder number and change LineGraphDescriptions:

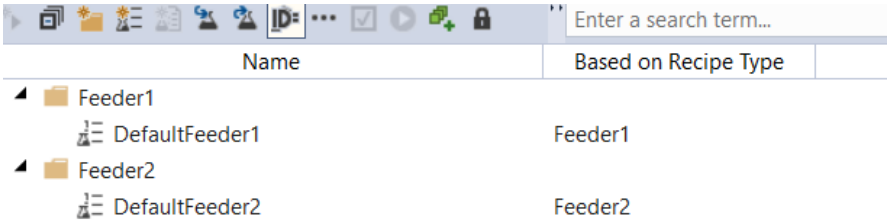


Recipe Management

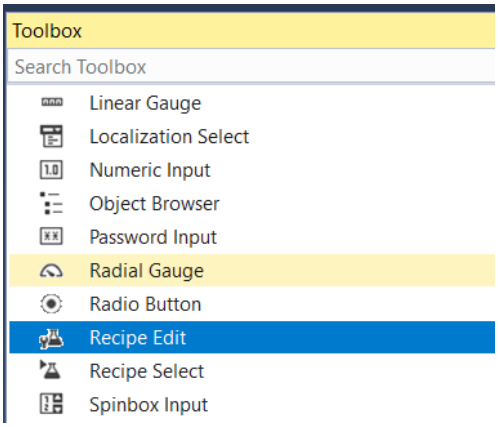
1. Add Recipe Server extension
2. Add two recipe types – Feeder1 and Feeder2
3. Select Feeder1 Recipe symbols and rename display names



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



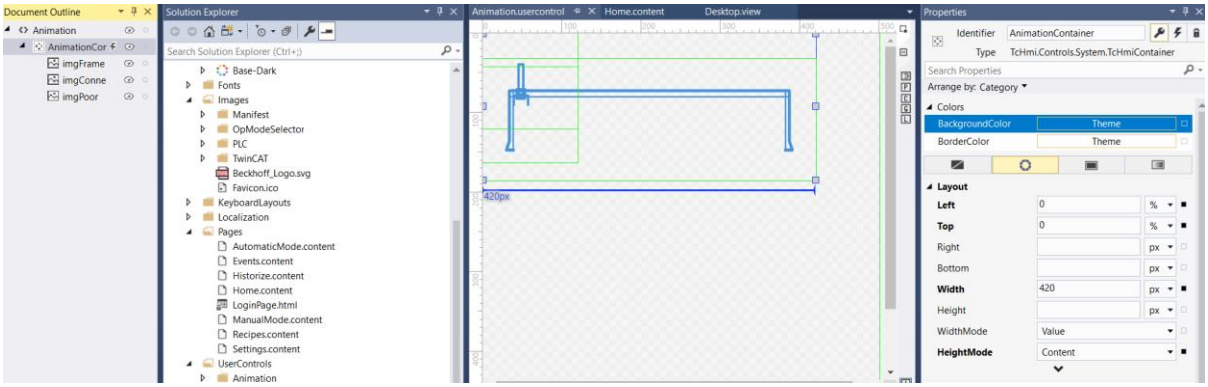
6. Add Recipe Edit control on Recipe page:



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

Animation User Control

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:

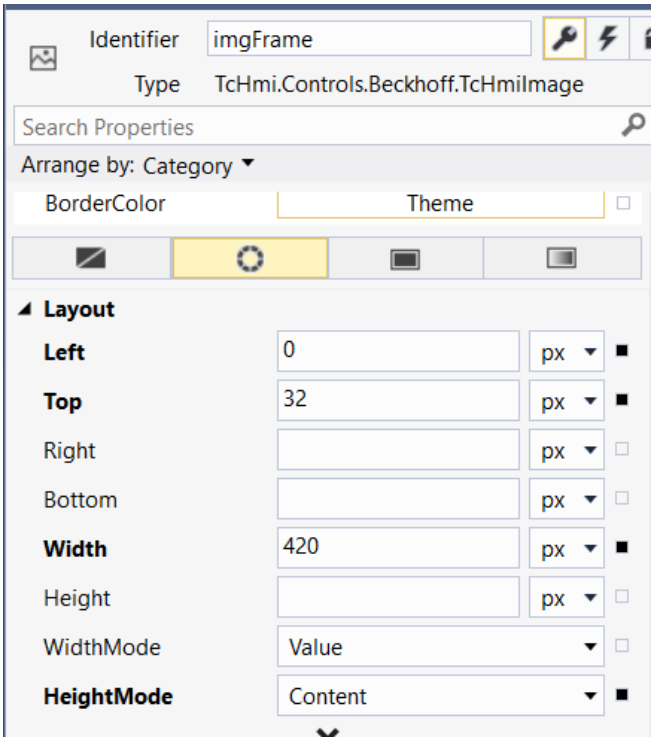
Edit/Define Parameters

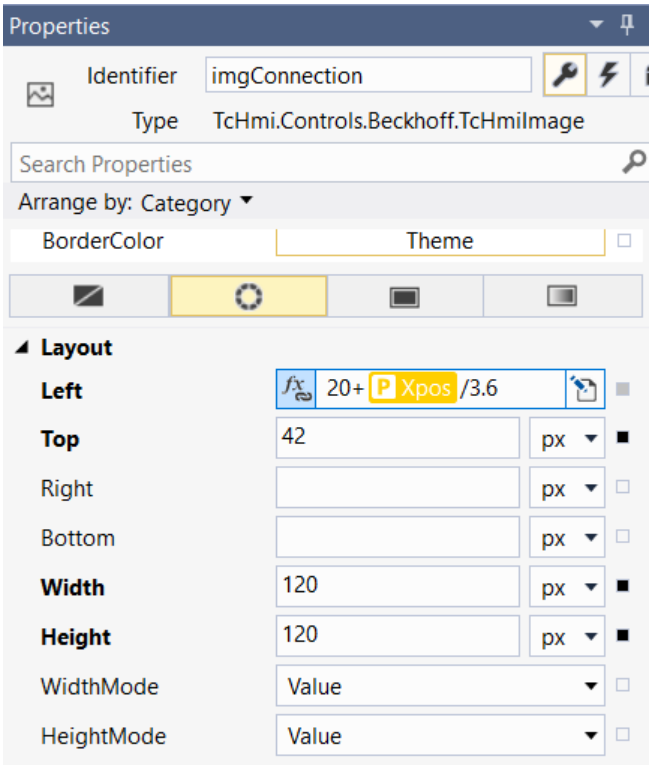
Description:

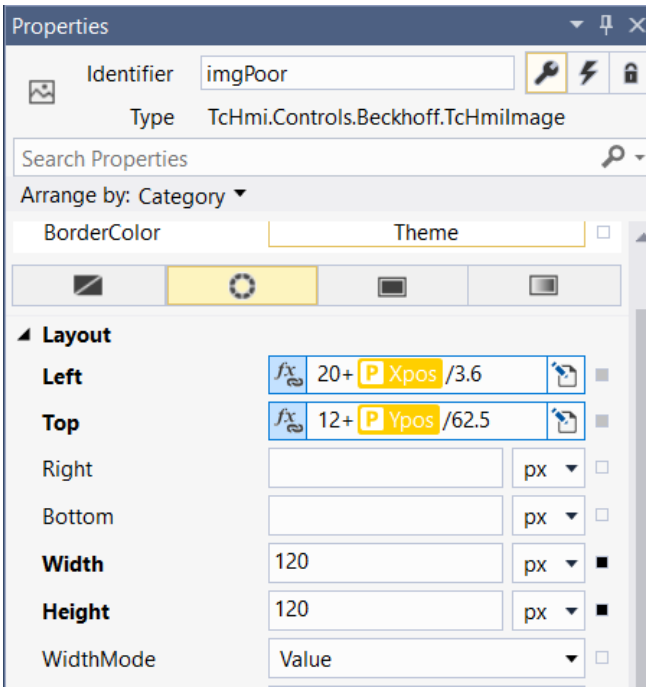
Parameters:

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
Xpos	G Number			☑	☑
Ypos	G Number			☑	☑

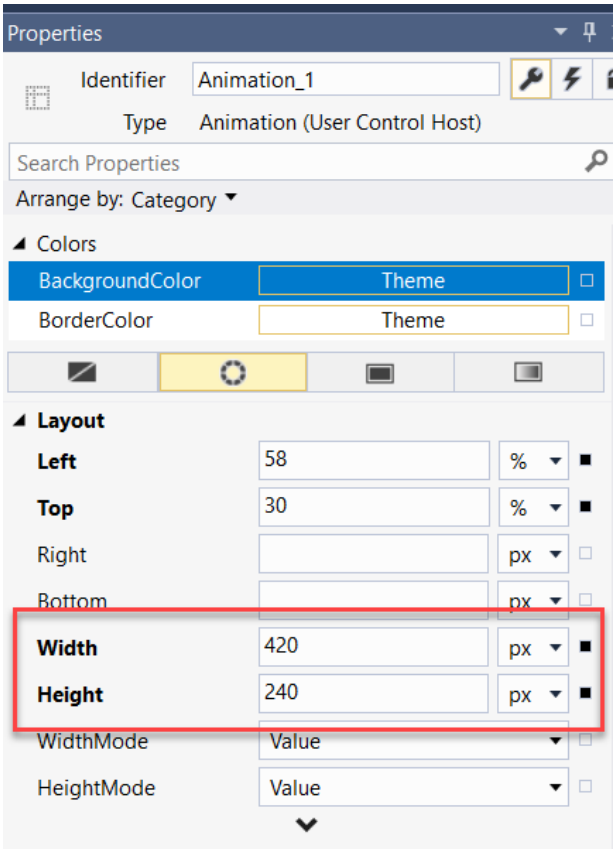
4. Add frame, connection, poor images with the following layout:



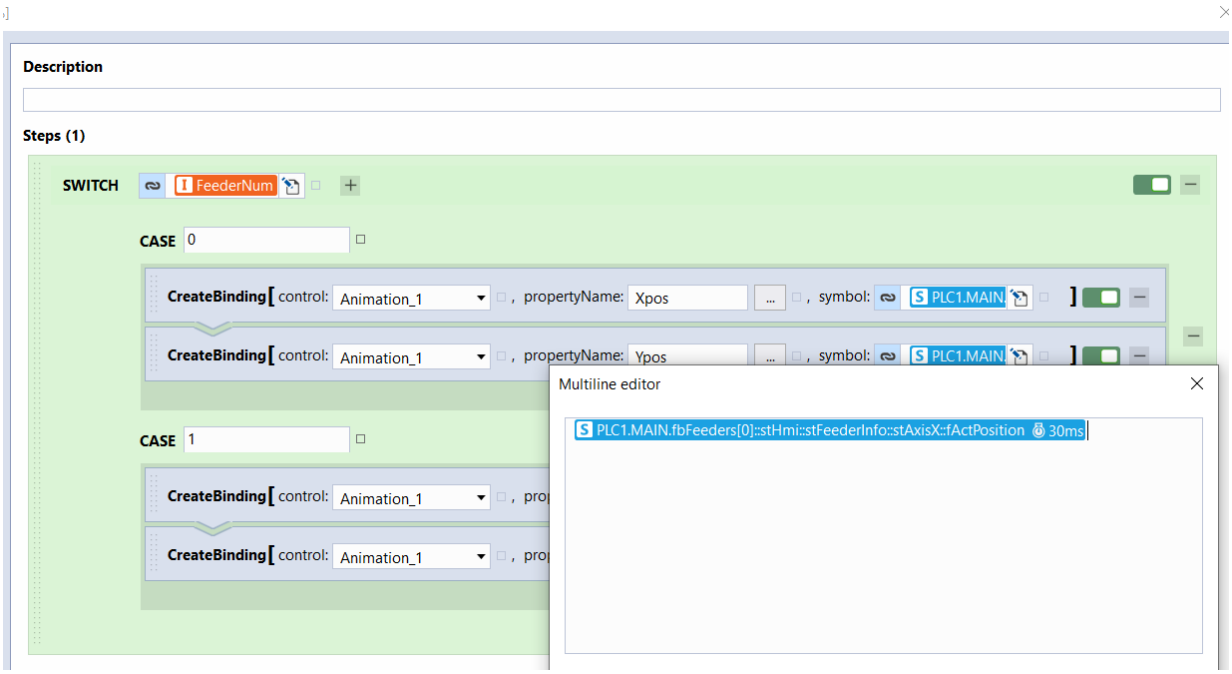




5. Add one instance of the Animation UC on the Home page with the following layout properties:



6. Add trigger event for feeder change and configure actions to bind X and Y positions from Feeder 1 or 2:



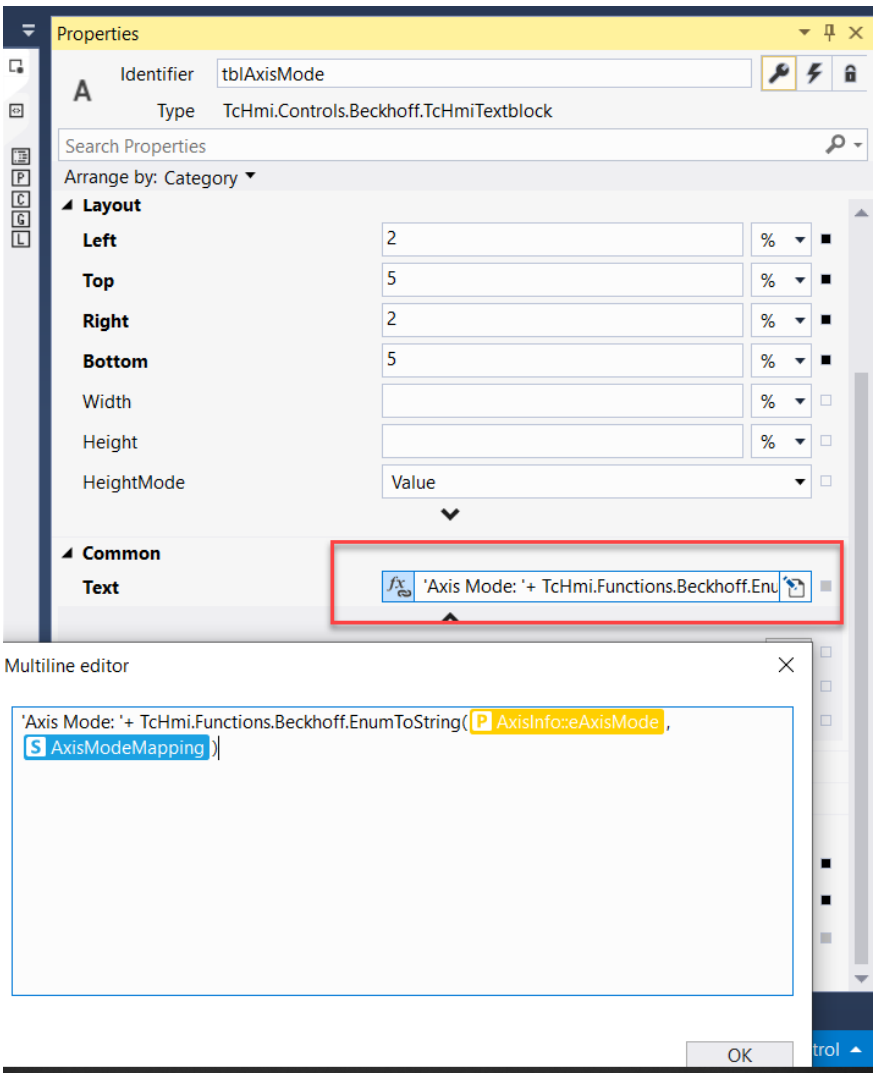
Enum Descriptions

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"
}
```

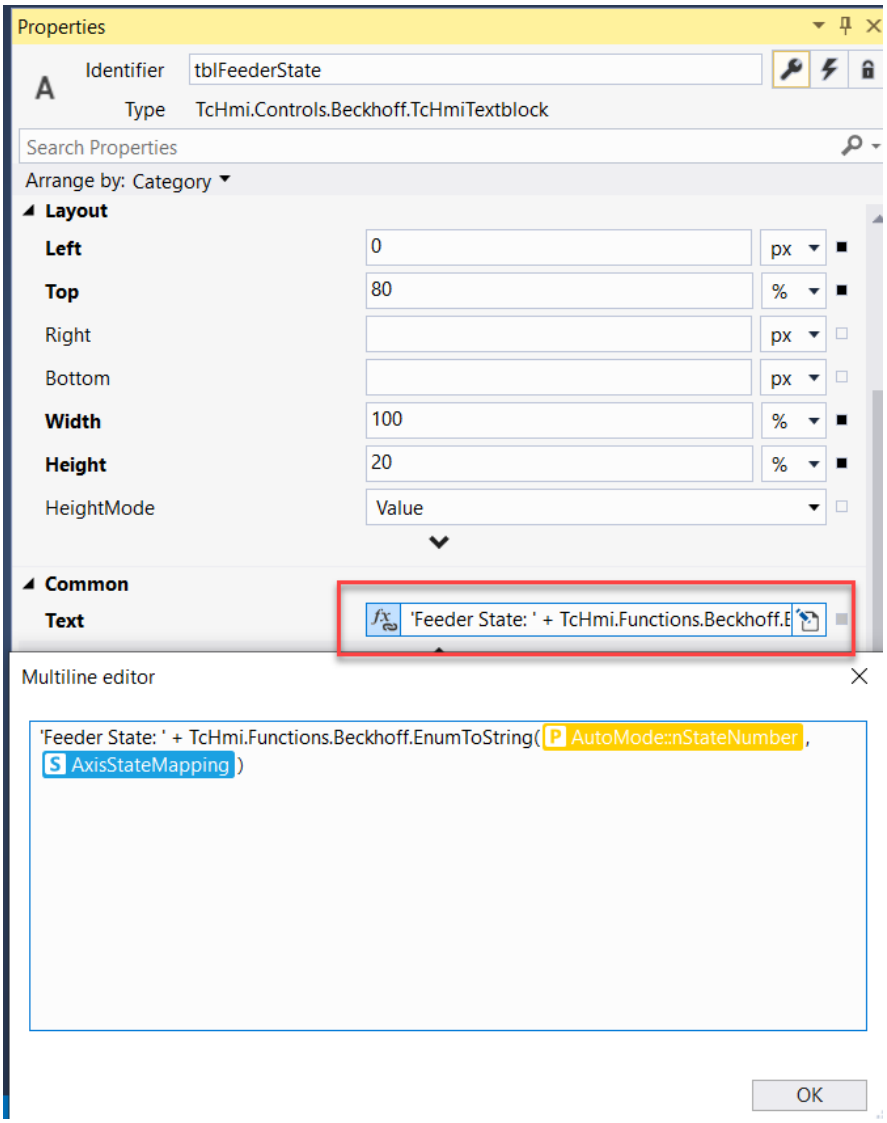
Bind Axis mode textblock in Automatic base UC:



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"
}
```

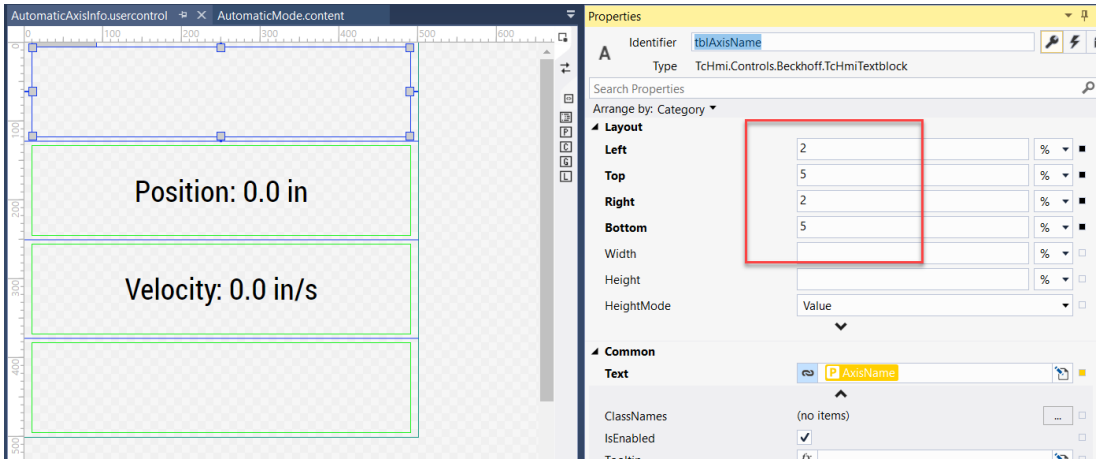
Bind FeederAutomatic UC:



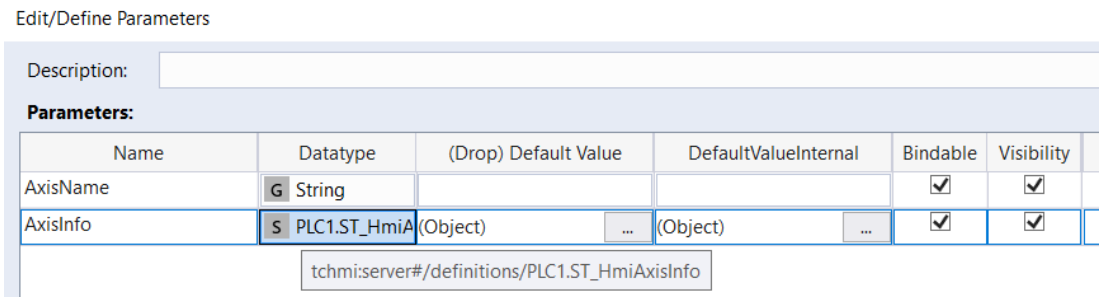
Appendix – Build User Controls (UCs)

Create Automatic Axis Info UC

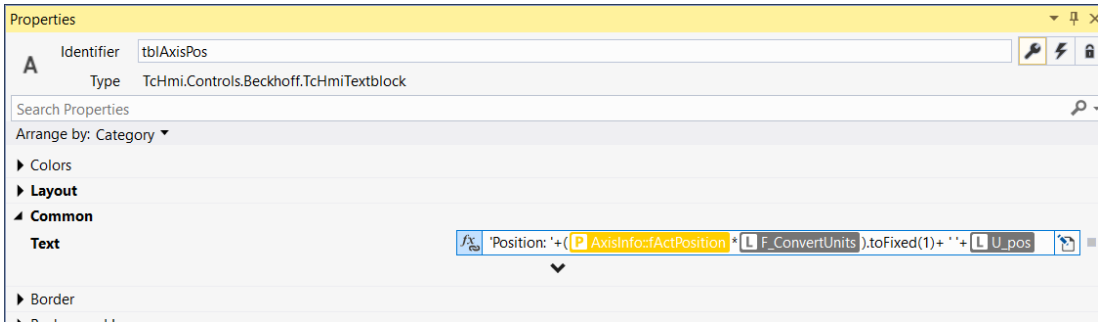
1. Grid centered inside UC host with 4 rows
2. Add textblock inside each UC – rename each textblock to AxisName, Position, Velocity, AxisMode:



3. Create 2 parameters for UC – AxisName (String), AxisInfo (PLC.ST_HmiAxisInfo)



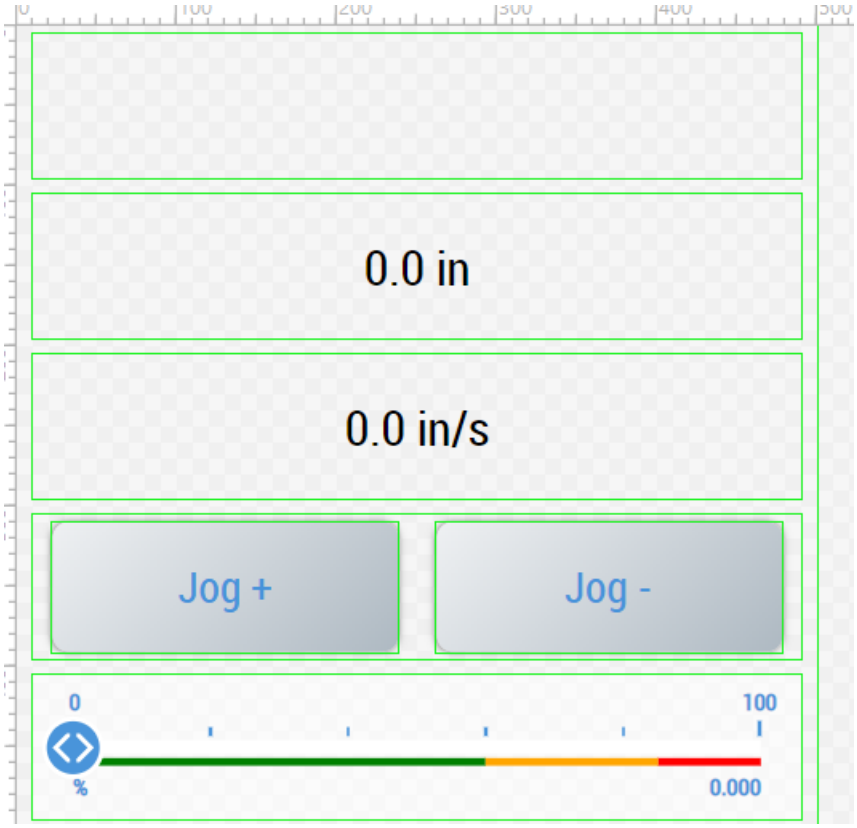
4. Bind Axis Name parameter to “Text” attribute. Make it a function binding and add raw text, F_ConvertUnits, U_Pos,.toFixed(1) for showing only one decimal point:



Create similar binding of sub-variables from AxisInfo parameter with other textblocks in the UC host.

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 5th cell:



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

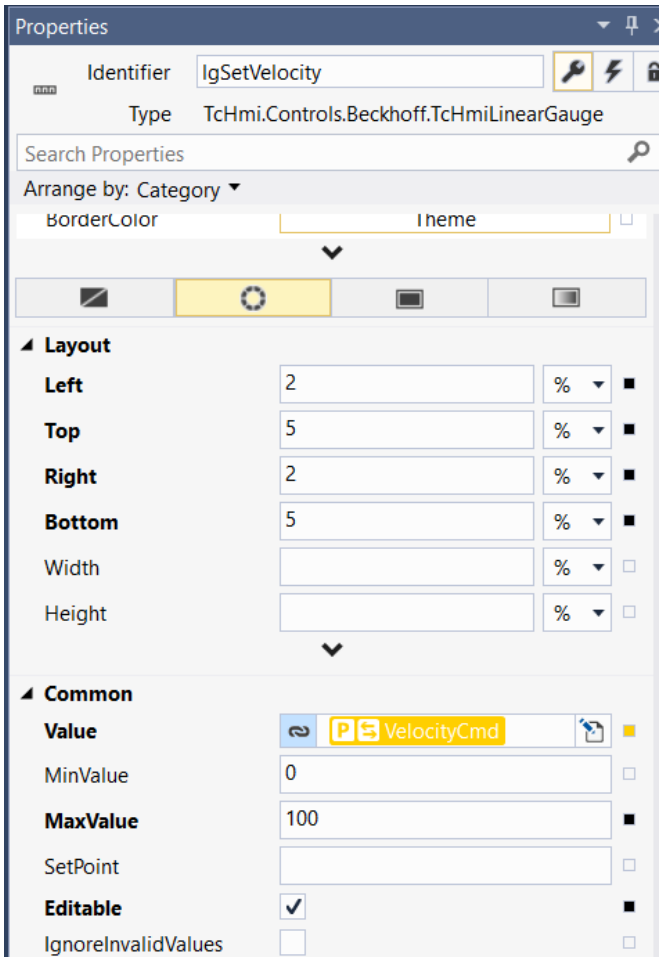
Edit/Define Parameters

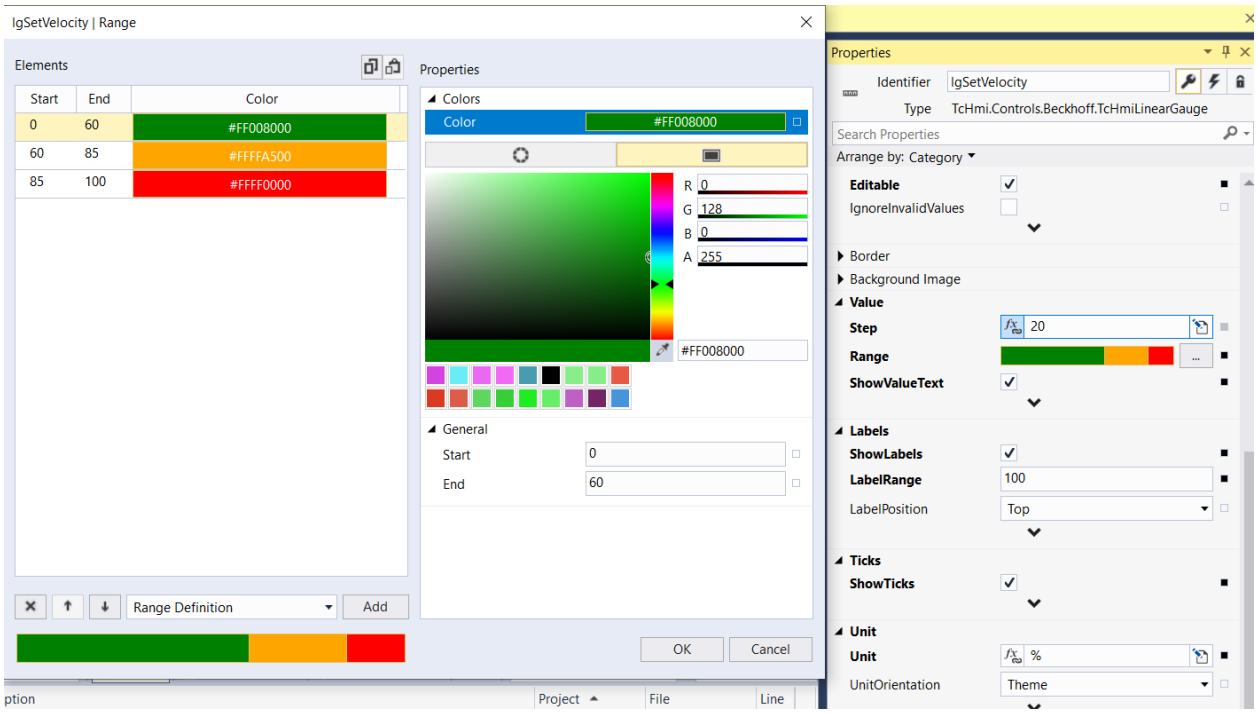
Description:

Parameters:

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
AxisName	G String			☑	☑
JogPlus	G BOOL	False	False	☑	☑
JogMinus	G BOOL	False	False	☑	☑
VelocityCmd	G Number			☑	☑
PositionFeedback	G Number			☑	☑
VelocityFeedback	G Number			☑	☑

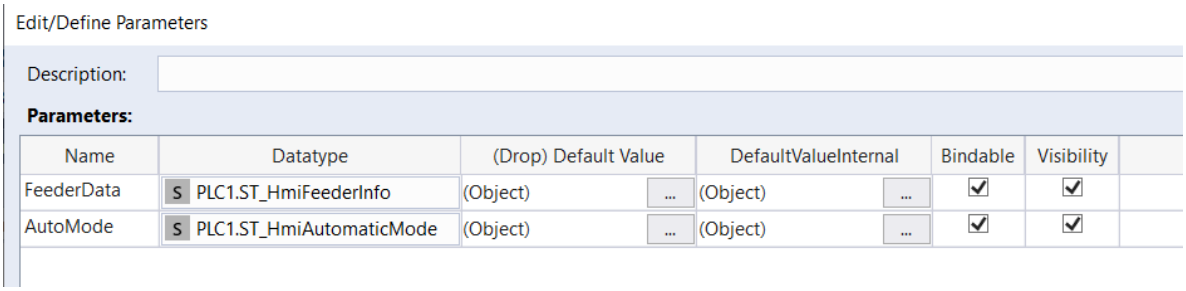
4. Bind from parameters to Axis Name, Axis ActPos (use localization symbol), Axis ActVelo ‘Text’ attribute
5. Bind from parameters to Jog + and – state symbol attribute
6. Create two-way binding with LinearGauge “value” and set range for display:


Properties
Identifier
lgSetVelocity
Type
TcHmi.Controls.Beckhoff.TcHmiLinearGauge
Search Properties
Arrange by: Category
BorderColor
Theme
Layout
Left
2
%
Top
5
%
Right
2
%
Bottom
5
%
Width
%
Height
%
Common
Value
VelocityCmd
MinValue
0
MaxValue
100
SetPoint
Editable
IgnoreInvalidValues

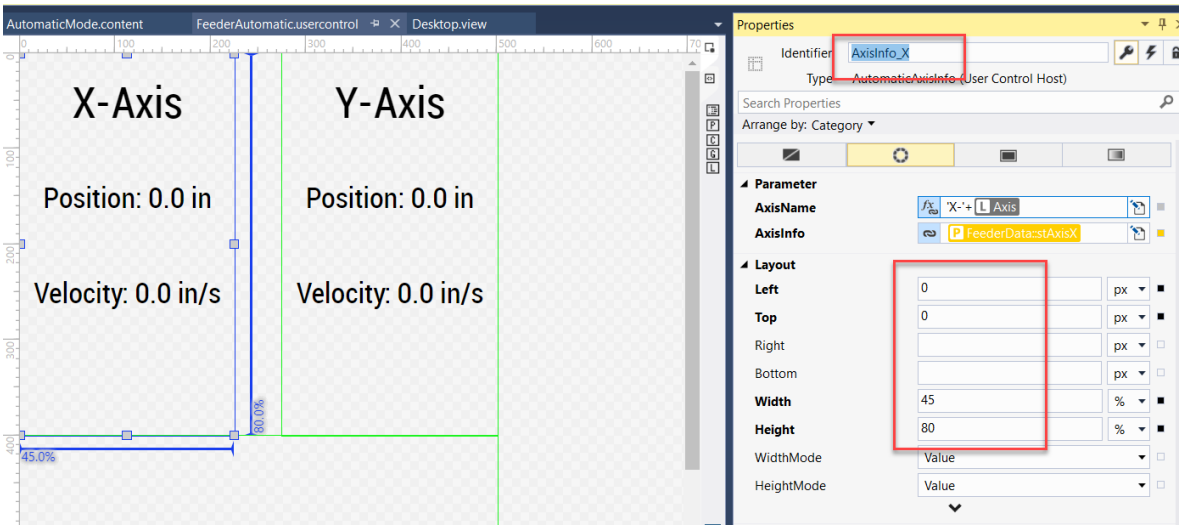

lgSetVelocity | Range
Elements
Start
End
Color
0
60
#FF008000
60
85
#FFFFA500
85
100
#FFFF0000
Range Definition
Add
Properties
Colors
Color
#FF008000
R 0
G 128
B 0
A 255
#FF008000
General
Start
0
End
60
OK
Cancel
Project
File
Line
Properties
Identifier
lgSetVelocity
Type
TcHmi.Controls.Beckhoff.TcHmiLinearGauge
Search Properties
Arrange by: Category
Editable
IgnoreInvalidValues
Border
Background Image
Value
Step
20
Range
ShowValueText
Labels
ShowLabels
LabelRange
100
LabelPosition
Top
Ticks
ShowTicks
Unit
Unit
%
UnitOrientation
Theme

Create FeederAutomatic UC

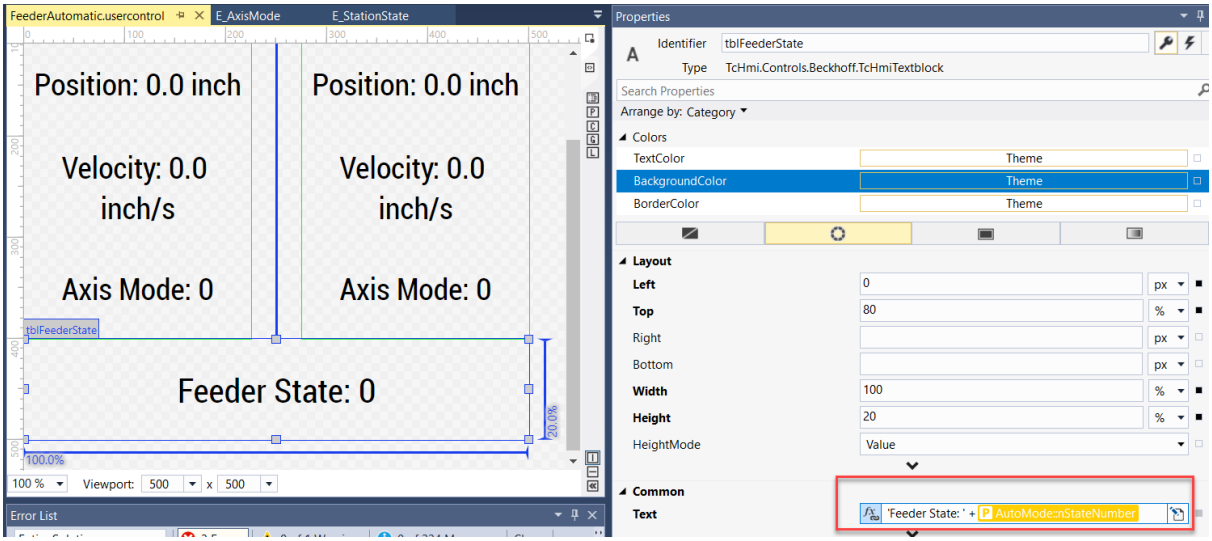
1. Add 2 parameters:



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

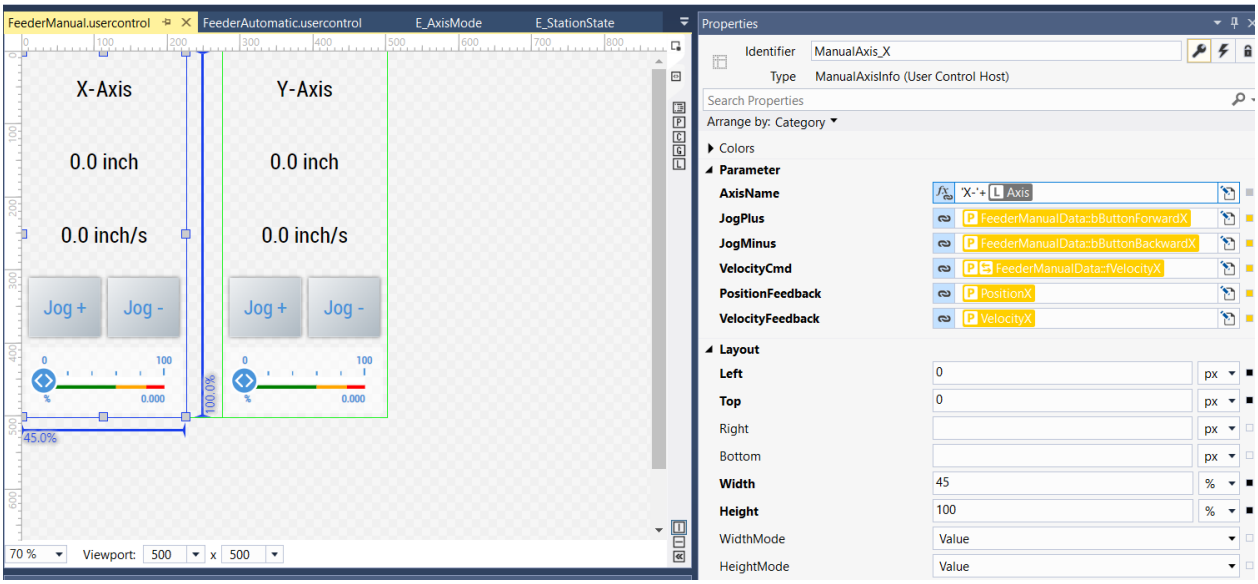


3. Add textblock in bottom 20% of space:

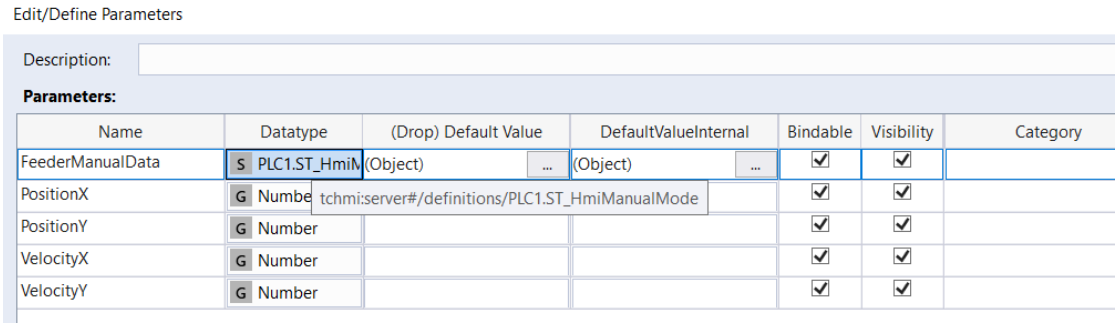


Create FeederManual UC

1. Add two instances of manual axis UC



2. Add 5 parameters – FeederData (ST_HmiFeederInfo), PosX and PosY (numbers), VeloX and VeloY (numbers)



3. Bind nested UC parameters to parent UC