



PLCNext Engineer

eHMI Functions

Creating and Using Symbols



Table of Contents

1. What this Document addresses	3
2. Pre-Requisites	3
3. Create a Custom Data Type (UDT)	4
4. Define a PLC Variable using the newly Defined UDT	6
5. Create a basic PLC Program using the Array of UDTs	7
6. Create a Symbol in eHMI	8
7. Creating UDT in eHMI	11
8. Assigning UDT to the eHMI Symbol	12
9. Creating a eHMI Page with Multiple Symbols.....	16
10. Runtime Example with Multiple Symbols	17
11. Conclusion.....	18



1. What this Document addresses

Consider a project that has many of the Same Pieces of Equipment – 10 Motors, 10 Gauges or 10 Conveyors, etc. In Programming the Equipment, you created a Structure – and an Array of those Structures to easily hold and control all of the Data Associated with it. This is great for the PLC, but when you go to create the HMI Objects, you must create and assign each Object separately. You can use Grouping, Cut and Paste and other methods to assign each of the Tags to the HMI objects easier. However, after you pull your hair out, you realize that was a lot work to create and now you will have to maintain each object and Tag connection. This is obviously tedious and opens a potential for mistakes.

“Symbols” in our eHMI is a Solution. Symbols allow you to create individual Functions (Pushbuttons, Checkboxes, any eHMI Object you may need) combine into a single Element and Pass Individual Tags or an entire Structure of Data to that Graphic. This creates synergy between the PLC and eHMI and removes the tedious chore of having to create and assign each item, one by one.

In this Document, we create a custom Array of Structures in the PLC and a simple Program. We then Create a Symbol in the eHMI and pass an entire Structure to that Symbol, demonstrating the power and simplicity.

NOTE: It is not Required to use UDTs in Symbols. PLCNext Engineering allows you to address and assign each Object in a Symbol individually. However, this document was written to take advantage of using the same UDTs in a PLC and eHMI to demonstrate the flexibility and efficiency.

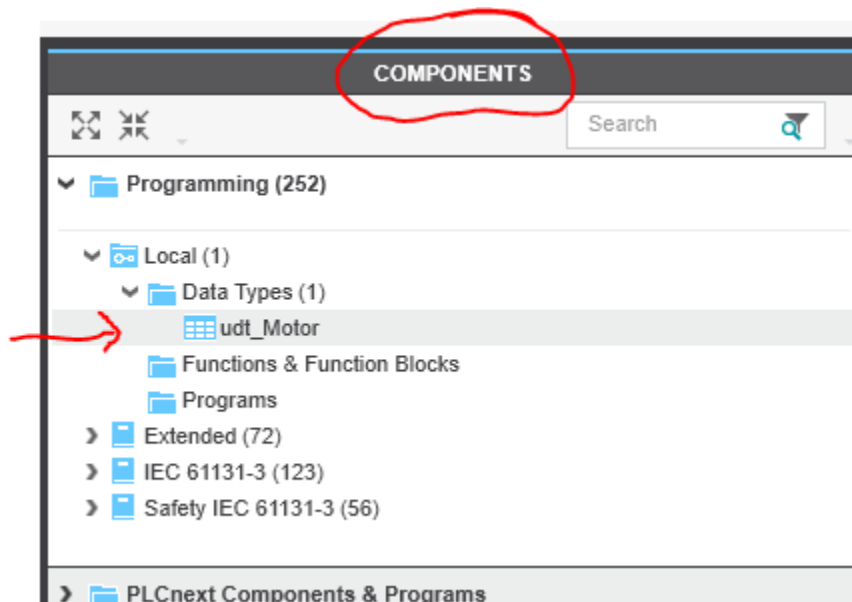
2. Pre-Requisites

Basic PLCNext Navigation and Program Construction is a Must. You must know how to Create and Use Data Types and Arrays and be able to Navigate the HMI Webserver Screens and Functions.

3. Create a Custom Data Type (UDT)

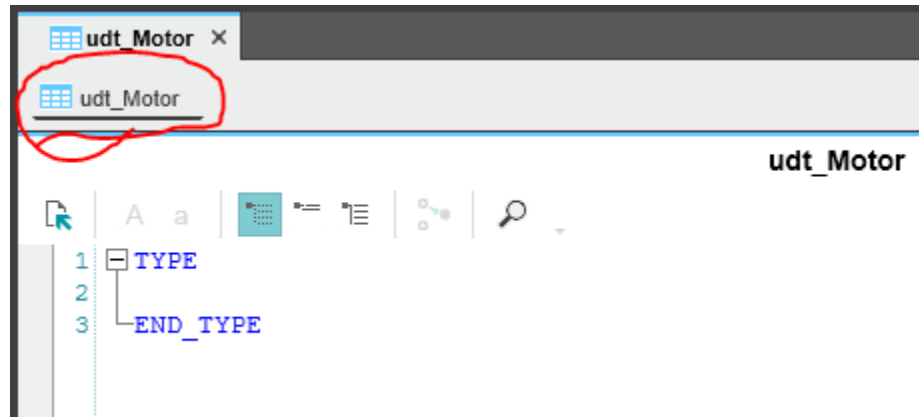
The first thing you must do is to create a UDT (User Defined DataType) Structure that will hold a list of the variables that we can use in our PLC Program and eHMI.

Right-Click on Data Types to create a DataType WorkSheet. In this example, I renamed the Worksheet to `udt_Motor`. This Name can be anything you want, but should be descriptive to the type you are creating for easy reference later. This name has no bearing on execution or performance... it is just a container for our newly defined UDT.



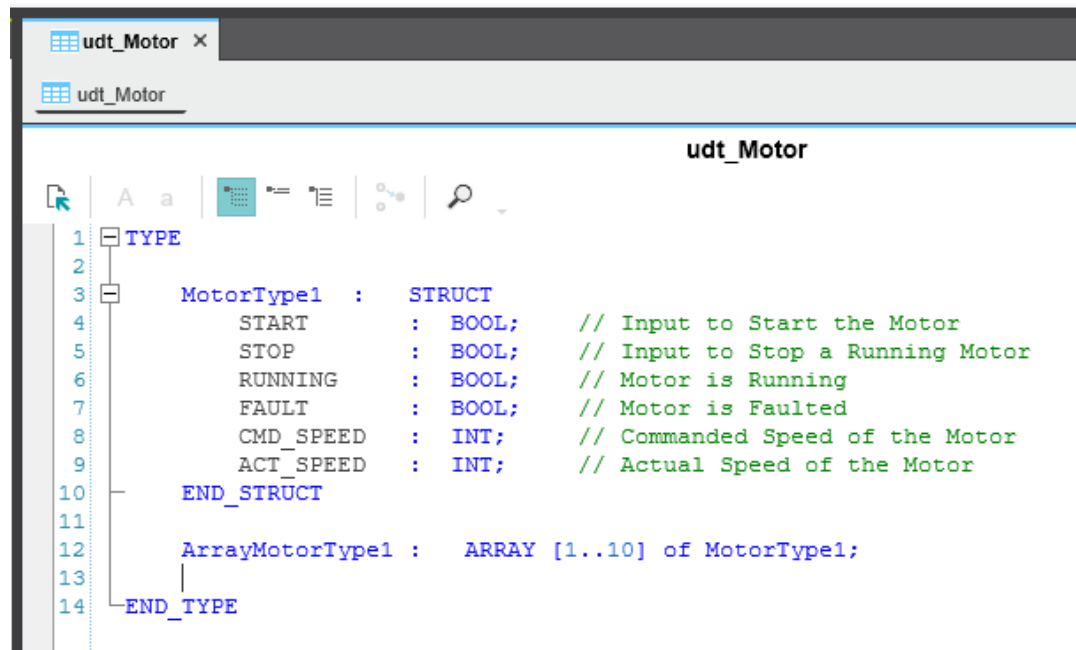


Double-Click on the DataType You just created above (in this example, I double-clicked on “udt_Motor”..) The following will appear.



This is where your custom Data Types are Defined for use in your programs.

In this example, I defined a Structure with a few Tags that are typically used when I want to Monitor and Control a basic Motor. I then Created an ARRAY of 10 of the MotorType1, as below:



Close the udt_Motor Tab.



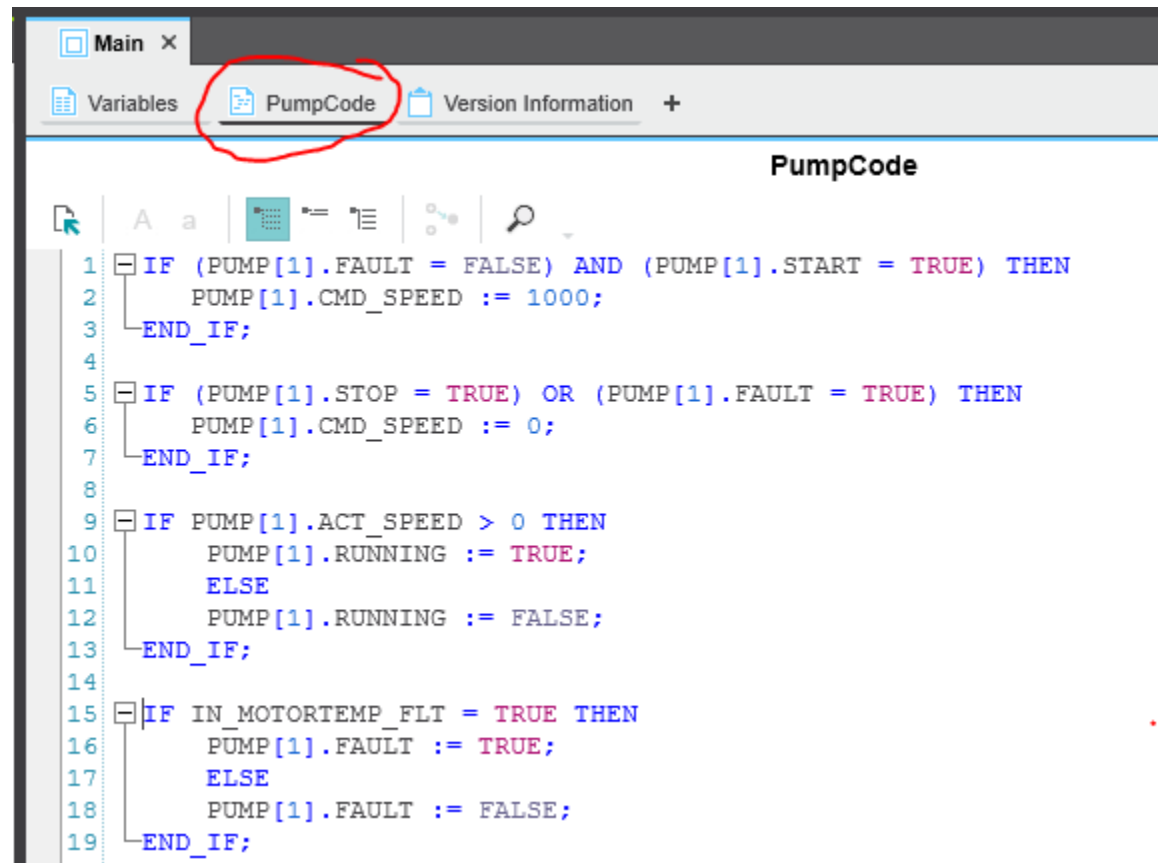
4. Define a PLC Variable using the newly Defined UDT

In the Programs Section, open the 'Main' program. Create a Variable named 'PUMP', using the Arrays of UDTs "ArrayMotorType1". We will be using this Array to Monitor and Control upto 10 Pumps with a PLC Program. NOTE: the Variable IN_MOTORTEMP_FLT could be Tied in a Physical Input Card that would go 'High' when a Motor Overtemperature Fault occurred on the connected Motor (and can be used for simulation purposes as well).

	Name	Type	Usage
▼ Default			
	PUMP	ArrayMotorType1	External
	IN_MOTORTEMP_FLT	BOOL	External
	Enter variable name here		

5. Create a basic PLC Program using the Array of UDTs

Create a small Program to Control Each of the Ten Motors I defined. Here is a simple program to run the first Motor (Pump[1]).

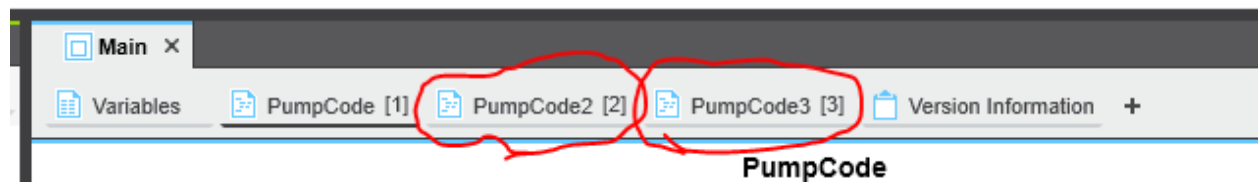


```

1 IF (PUMP[1].FAULT = FALSE) AND (PUMP[1].START = TRUE) THEN
2   PUMP[1].CMD_SPEED := 1000;
3 END_IF;
4
5 IF (PUMP[1].STOP = TRUE) OR (PUMP[1].FAULT = TRUE) THEN
6   PUMP[1].CMD_SPEED := 0;
7 END_IF;
8
9 IF PUMP[1].ACT_SPEED > 0 THEN
10  PUMP[1].RUNNING := TRUE;
11 ELSE
12  PUMP[1].RUNNING := FALSE;
13 END_IF;
14
15 IF IN_MOTORTEMP_FLT = TRUE THEN
16  PUMP[1].FAULT := TRUE;
17 ELSE
18  PUMP[1].FAULT := FALSE;
19 END_IF;
  
```

In the Above Program, I am using PUMP[1] Array to control and Monitor the First Pump. It is important to Realize that all of the Control and Monitoring associated with this Motor is included in this Array.

I created additional Sheets for PUMP[2] and PUMP[3] so that I can show multiple Motors Functioning on a eHMI Page that we create later. I copied the above program and just changed [1] to a [2] for Motor 2 and [3] for Motor 3.

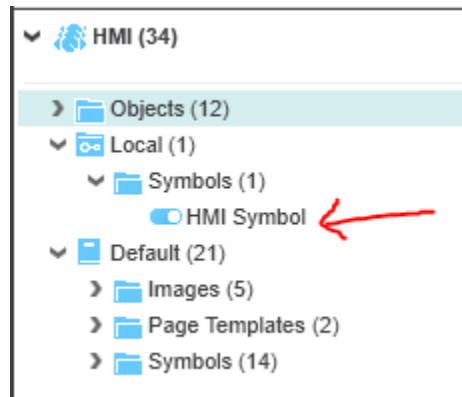
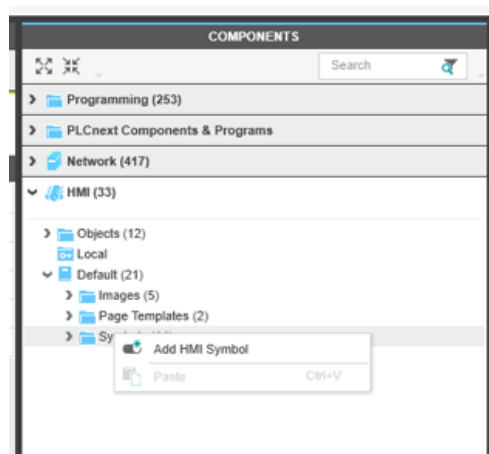


6. Create a Symbol in eHMI

Since we have the UDT Defined and are using an Array of them to control many systems, we can associate eHMI “Symbols” to the same UDT and make them easily reusable. In this way, we only need to pass the Array Number to each of the Symbols, since they all

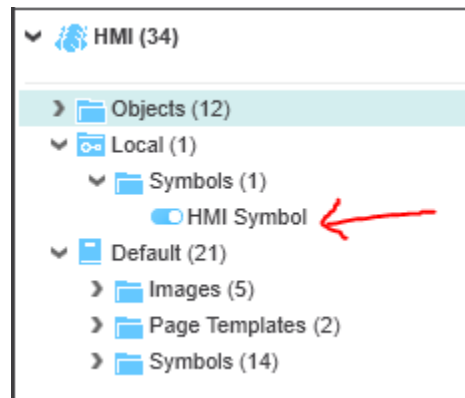
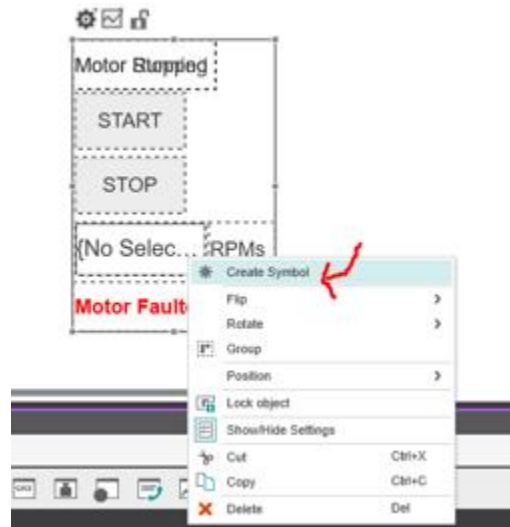
There are two ways to Create an eHMI Symbol:

1. Right Click on Symbol <Add HMI Symbol>



Change the Name to Motor1_Symbol, by Right-Mouse Click.

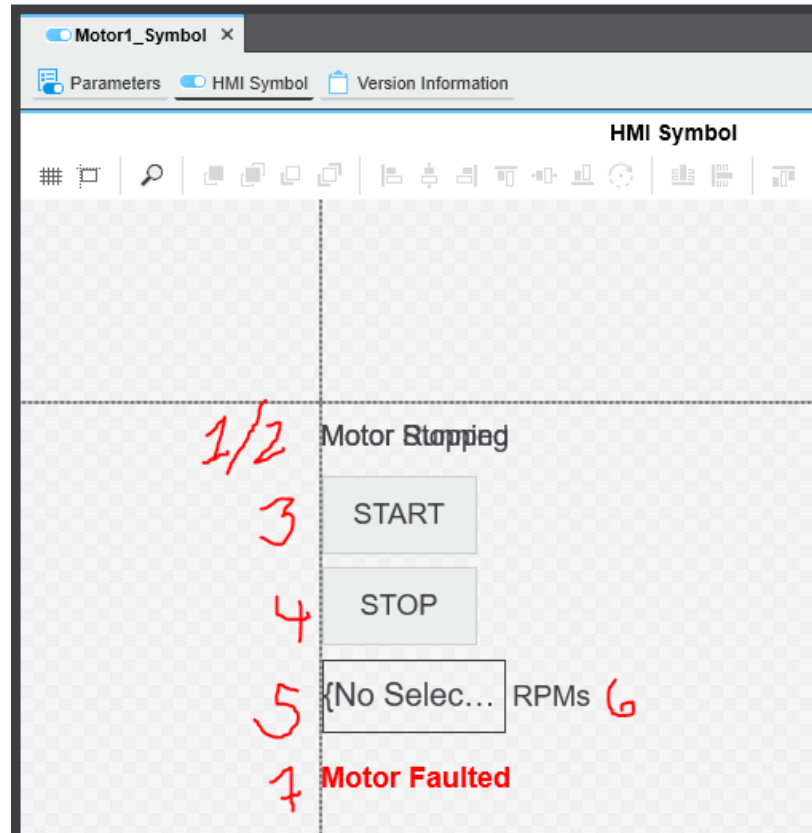
2. Create all the Objects how and where you like them on a Page. Then, Select all of the Items you want to be included in the Symbol. Right Mouse Click and Select 'Create Symbol'.



You can then change the Name to Motor1_Symbol, by Right-Mouse Click.

To Edit the Symbol, Double-Click on Motor1_Symbol.

I Created the Following:



Item #1 and #2 are Static Text (on Top of Each Other). "Motor Running" and "Motor Stopped".

Both have the Dynamic 'Visibility' activated. We will use the '.Running' Variable to determine which to Display. They are 'Overlaid' on top of each other. You will have to separate them, Modify the controls, then Overlay them again.

Item #3 and #4 are Push Buttons with the appropriate Text per Function.

Item #5 is Text with Dynamic 'Text' Activated.

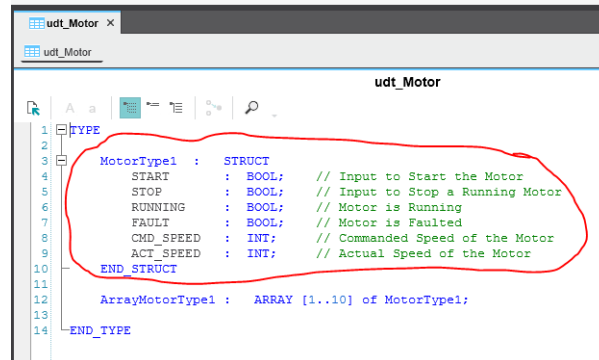
Items #6 is Static Text (and will not Change)

Item #7 is Text with Dynamic 'Visibility'.

7. Creating UDT in eHMI

Now that we have the Symbol Graphic Completed, we need to create a 'Temporary' Variable that the eHMI will use to Pass the UDT from the PLC to the Symbol. The Symbol will use the same UDT Structure as the Arrays, so that there is a 1:1 correlation of PLC Program to eHMI Variables.

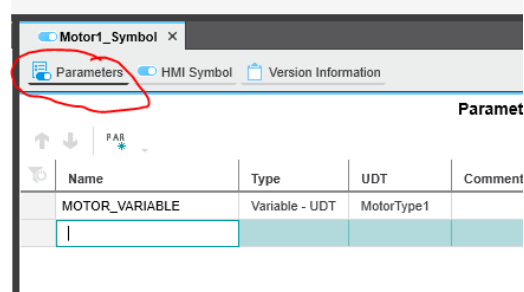
Remember that we will be passing a Structure (not Array) of Motor_Type1 to each Symbol.



Therefore, when we define the UDT in the eHMI, we use the same Data Type Structure that we used in the Array – in this Case, MotorType1.

To do this, Select the Symbol Created. Go to Parameters and create a Temporary Name that we will use in the Symbol to Pass the Data from the Array to the Symbol. I used a Temporary tag, called "MOTOR_VARIABLE" and defined it as DataType "MotorType1".

In doing so, I now have the same Structure as the Arrays that are used in my PLC Program, but assigned to a 'Temporary Tag':



MOTOR_VARIABLE.START
 MOTOR_VARIABLE.STOP
 MOTOR_VARIABLE.RUNNING
 MOTOR_VARIABLE.FAULT
 MOTOR_VARIABLE.CMD_SPEED
 MOTOR_VARIABLE.ACT_SPEED



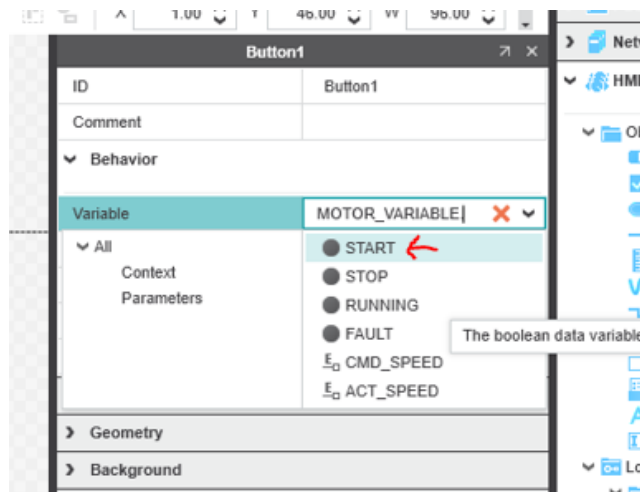
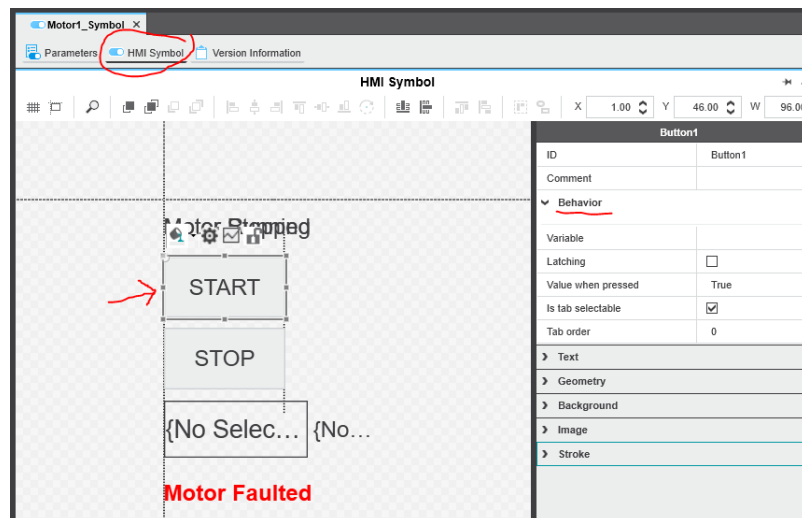
8. Assigning UDT to the eHMI Symbol

We now go to each Item in the Symbol we created and Pass the Values of the UDT to each of them so that they graphics respond correctly to the Data Passed to it. Since we want them all to look and perform the same, we only need to do this 1 time, which is why a 'Symbol' is used.

We need to associate the Tags and Performance of each Object with the Symbol Parameters so that the Data will Properly Populate. To do so,

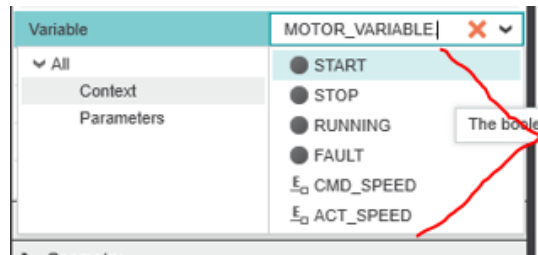
Select "HMI Symbol" Tab, Select Each Item and Tie it to the Proper Behavior or Dynamic Control for Each Object that was Created. Below, we are Editing the Push Button 'Start' and Tying to the UDT Parameter Object... Which is MOTOR_VARIABLE.xxxxxxxx

When you want to use a Parameter from the UDT, when you see a Variable input, select the Box, Start Typing MOTOR_VARIABLE, then Put a <Period>... all of the Available Tags will appear, as below.





Notice that the Variables Available for the Symbol are the same as UDT

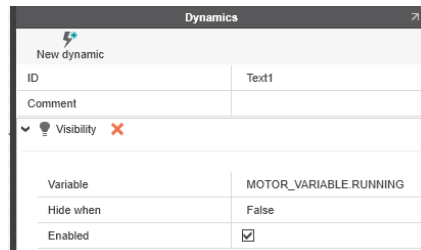


```
TYPE
MotorType1 : STRUCT
    START   : BOOL;
    STOP    : BOOL;
    RUNNING  : BOOL;
    FAULT    : BOOL;
    CMD_SPEED : INT;
    ACT_SPEED : INT;
END_STRUCT
```

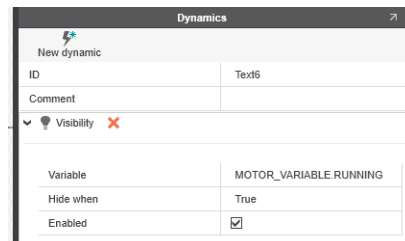
Now, Finish Arranging all the Data for each of the Objects in the Symbol and how you want them each to Perform.

I did the Following (I only show the Dynamics or basic Definition).

Motor Running:



Motor Stopped:





START:

Button1	
ID	Button1
Comment	
▼ Behavior	
Variable	MOTOR_VARIABLE.START
Latching	<input type="checkbox"/>
Value when pressed	True
Is tab selectable	<input checked="" type="checkbox"/>
Tab order	0
▼ Text	
Text	START
Text when down	
Format	Arial 18 Normal Center Middle
Margin	0
Direction	Left to right

STOP:

Button2	
ID	Button2
Comment	
▼ Behavior	
Variable	MOTOR_VARIABLE.STOP
Latching	<input type="checkbox"/>
Value when pressed	True
Is tab selectable	<input checked="" type="checkbox"/>
Tab order	0
▼ Text	
Text	STOP
Text when down	
Format	Arial 18 Normal Center Middle
Margin	0
Direction	Left to right

MotorSpeed:

Dynamics	
New dynamic	
ID	Text3
Comment	
▼ A Text ✖	
Variable	MOTOR_VARIABLE.ACT_SPEED
Format	0 digits after decimal
Enabled	<input checked="" type="checkbox"/>



MotorFault:

A screenshot of a software interface titled "Dynamics". It shows a configuration window for a new dynamic. The window has a header bar with a lightning bolt icon and the text "New dynamic". Below this is a table with three rows: "ID" with value "Text2", "Comment" (empty), and "Visibility" with a red "X" icon. Below the table is another section with three rows: "Variable" with value "MOTOR_VARIABLE.FAULT", "Hide when" with value "False", and "Enabled" with a checked checkbox.

New dynamic	
ID	Text2
Comment	
Visibility	✗

Variable	MOTOR_VARIABLE.FAULT
Hide when	False
Enabled	<input checked="" type="checkbox"/>

The Symbol Definitions in the eHMI are now complete.

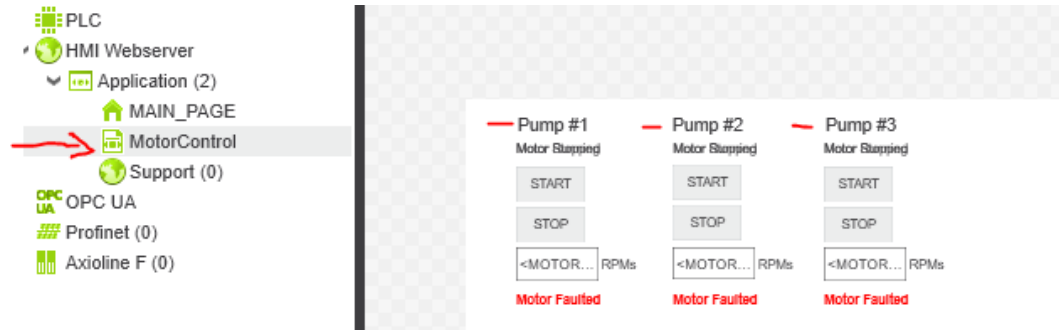
We can now put this Symbol in our eHMI Project and use it Multiple Times.

We simply assign one of Array Tags to each of the Motor1_Symbol Instances. This keeps our Look, Feel and Programming Simple.

9. Creating a eHMI Page with Multiple Symbols

In this example, I have Created a New HMI Page and inserted 3 Copies of the Motor1_Symbol we created. I assigned PUMP[1] to the First, PUMP[2] to the Second and PUMP[3] to the Third.

Note: I added the Text Pump#x to the Top of each of the New Symbols.

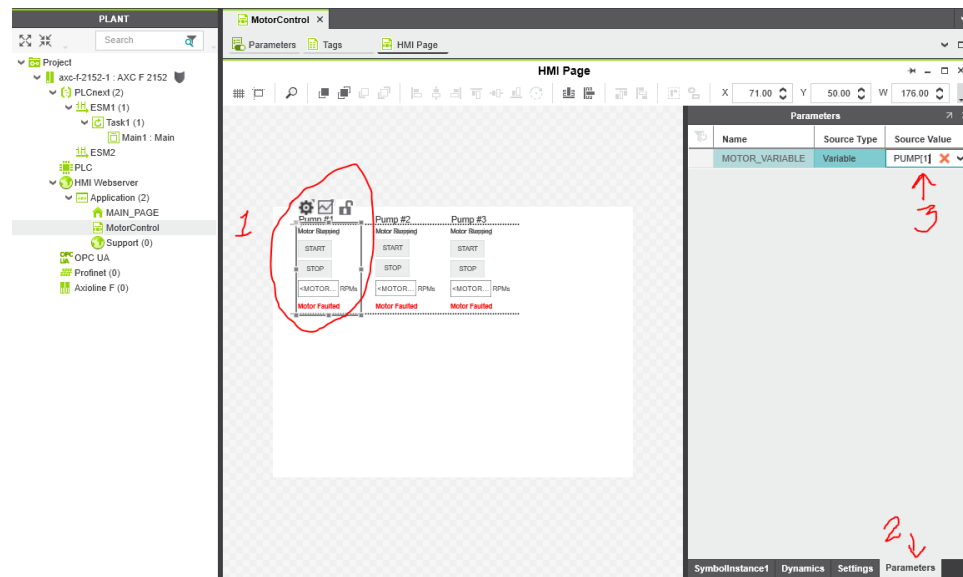


Once Created, we can now pass the Entire Structure to each of the Symbols.

To Do this:

1. Select the Symbol and Drag Multiple Instances of Motor1_Symbol onto an eHMI Page
2. Select 1 of the Instances of the Symbol
3. Select the "Parameters" Tab and Enter the Array "PUMP[1]" into the Source Value.

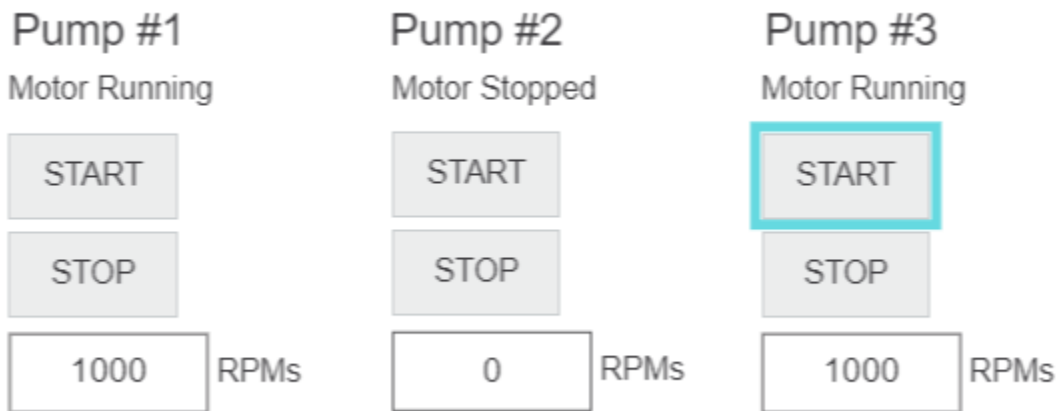
This correlates all of the values for PUMP[1] Structures to the eHMI Structure.



4. Do the Same for the Remaining PUMP[2] and PUMP[3].
5. Verify there are no Errors, then Download and Run the Program and the eHMI.

10. Runtime Example with Multiple Symbols

When Running, it will look something like the below:





11. Conclusion

To better understand and build upon this example. Try to add different Attributes or Colors to the eHMI Symbol we created or try to change the Program to force a fault to see the result. You could, for example, you could add a "NAME" (STRING) variable to the "Motor_Type1" Structure. You can then edit the NAME Variable in the PLC Program and Create a New Object in the eHMI Structure to display this name.

There are many powerful things that can be done with Structures and Symbols. Work Smart, not Hard ! Symbols should be a part of your projects in the future.