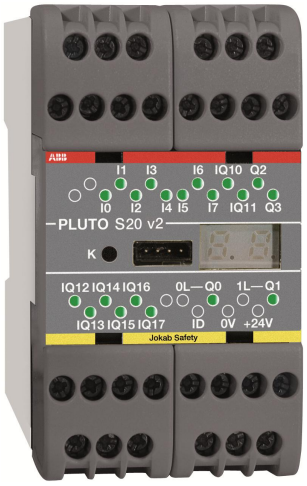

Pluto CP600 Panel Integration Manual

Table of contents:

1 General

Pluto is a programmable safety system intended for safety applications where it is not accepted that faults in the control system lead to loss of a safety function.

HMI panels are generally non-safe components. Keep this in mind when designing your safety application. Do not use the HMI for a reset of the machine.

See the “Pluto Hardware Manual” that is available in the tool “Pluto Manager” regarding how to connect and setup a Pluto unit.

2 Serial interface

Using a RS232 interface it’s possible to connect a Pluto unit with a CP600 panel by using Modbus ASCII. By using the Pluto unit’s programming port and a COM port on the HMI panel. All Pluto models can be connected to a CP600 HMI in this manner.

2.1 Cable

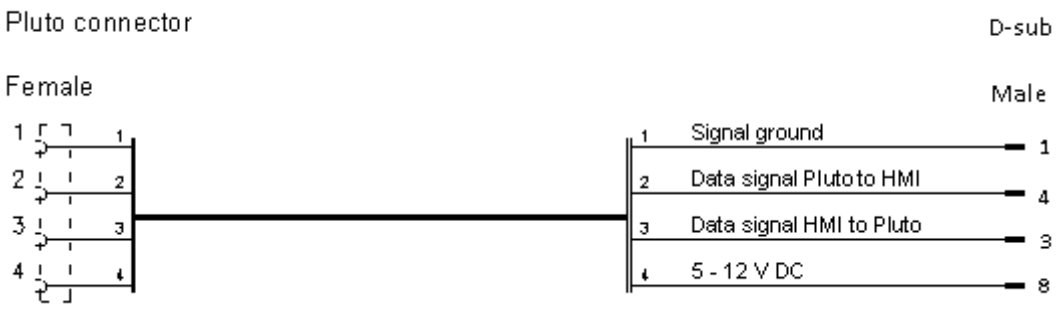

Figure 1

The total length of the cable, including connectors, should not exceed a maximum of 10 meters.

The HMI panels in the CP600 series can be connected to the Pluto by using the cable shown in the figure below.

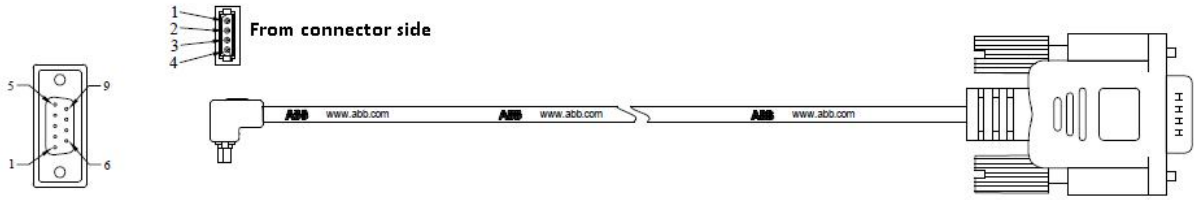

Figure 2

The male nine pole D-sub connector is connected to CP600 panel.

The female four pole connector is connected to the programming port (red arrow) of the Pluto unit.

The cable's length is 3 meters.

ABB article number **2TLA020070R6900**.

Figure 3

3 Pluto Manager

Pluto Manager is the tool used to create a Pluto project. In this document Pluto Manager version 2.30 is used.

A Pluto project contains local variables used only in the specific Pluto unit. It also has global variables which are read over the Pluto bus by all Pluto units.

It is also possible to export local variables to the Pluto bus. This type isn't the same as a global variable but has a similar function.

Only local variables and global variables can be read by the HMI panel, connected serially to the Pluto unit. A local variable in another Pluto unit has to be mapped to a global variable, or exported, and then mapped to a local variable in the unit that is connected to the HMI.

See the Pluto programming manual for an in-depth explanation of local, global, and export variables.

In the Pluto unit, to write variables from the Pluto to the panel when the panel performs a read request: Simply export the variables and import the resulting .csv-file into Panel Builder.

In the Pluto unit, to read variables from the Panel when the panel performs a write request, see the "Function Libraries" chapter below.

3.1 Data – Pluto to Panel, Export of variable names to be used in the panel

There are two ways of exporting variables in Pluto Manager. First, it's possible to menu-click on the complete project and choose the Pluto the panel is connected to. This way all variables will be included, including those that are global in other units. This means that the status of a global input in another Pluto unit than the unit the panel is connected to, is accessible.

Note that's it's important to choose the format "Comma separated CSV file for CP600/Modbus".

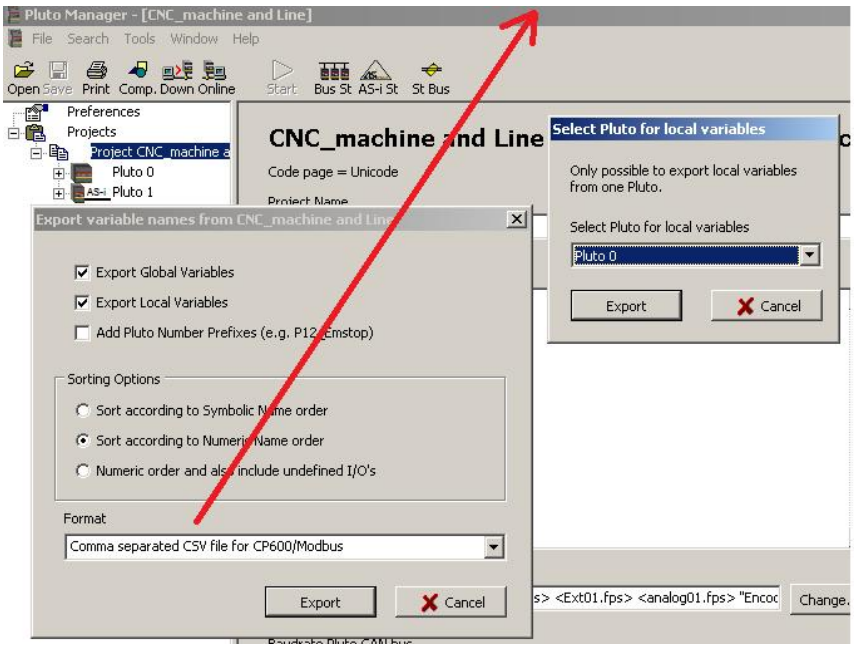

Figure 4

The other way is to menu-click on the “Variables” tab of the Pluto the serial cable will be connected to. Then only the variables in the Pluto unit will be exported, including those global in that unit only.

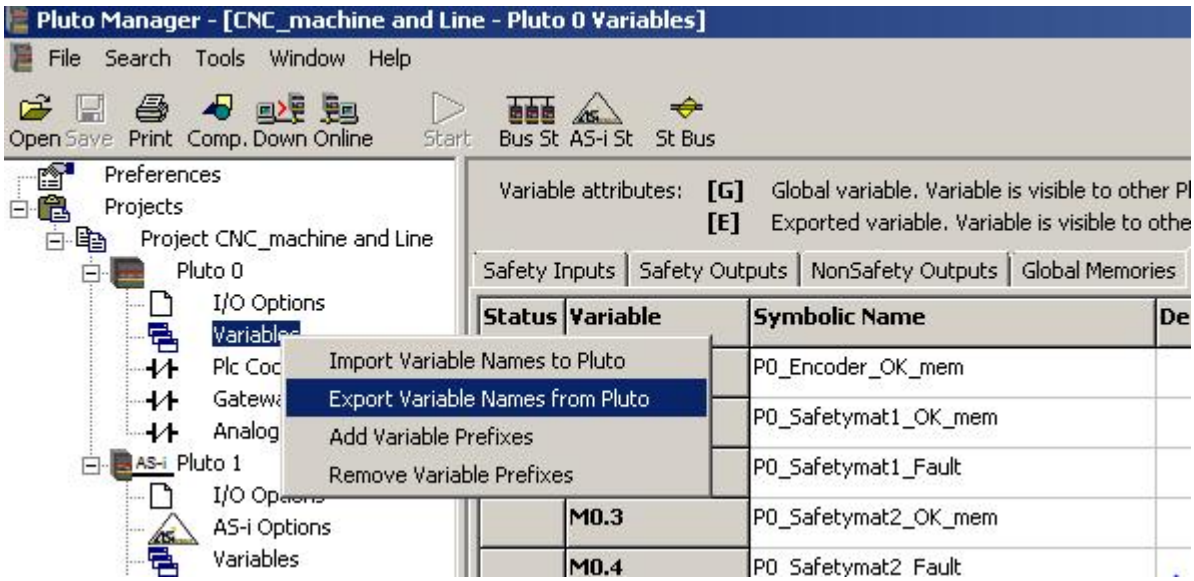

Figure 5

Export local and/or global variables as needed.

System variables contains, among others, error registers and logs which could be useful to show in the panel. However, they are not exported automatically. See below.

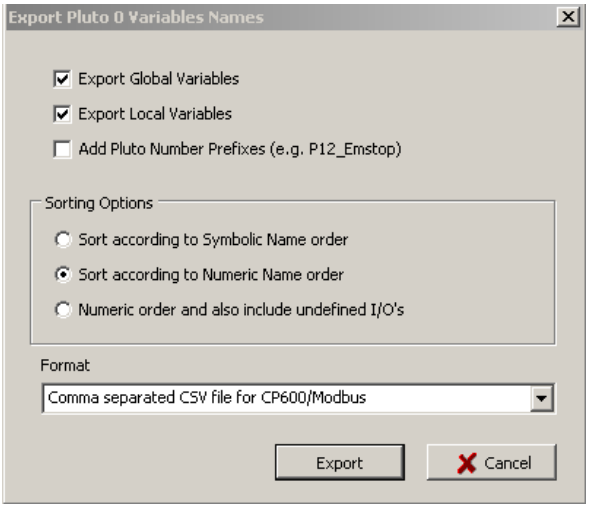

Figure 6

The resulting .csv-file ends up in the same directory as the Pluto project’s .sps-file.

The file can then be used to import variables into the Panel Builder 600 project.

3.1.1 Export of system variables

Currently automatic export of system variables is not supported. Some of them could be useful to show error states for example. Below is a listing of the useful ones. Please add the below to the CSV file if they are intended to be used.

```
1,SM_Flash,INP,101042,boolean,Flash 0.4s/0.6s (on/off)
1,SM_FastFlash,INP,101045,boolean,Flash 0.17s/0.33s (on/off)
1,SM_DoubleFlash,INP,101046,boolean,Flash 0.11/0.2/0.11/0.67ms (on/off/on/off)
1,SM_Pluto0_Present,INP,101140,boolean,Pluto #0 is present
1,SM_Pluto1_Present,INP,101141,boolean,Pluto #1 is present
1,SM_Pluto2_Present,INP,101142,boolean,Pluto #2 is present
1,SM_Pluto3_Present,INP,101143,boolean,Pluto #3 is present
1,SM_Pluto4_Present,INP,101144,boolean,Pluto #4 is present
1,SM_Pluto5_Present,INP,101145,boolean,Pluto #5 is present
1,SM_Pluto6_Present,INP,101146,boolean,Pluto #6 is present
1,SM_Pluto7_Present,INP,101147,boolean,Pluto #7 is present
1,SM_Pluto8_Present,INP,101148,boolean,Pluto #8 is present
1,SM_Pluto9_Present,INP,101149,boolean,Pluto #9 is present
1,SM_Pluto10_Present,INP,101150,boolean,Pluto #10 is present
1,SM_Pluto11_Present,INP,101151,boolean,Pluto #11 is present
1,SM_Pluto12_Present,INP,101152,boolean,Pluto #12 is present
1,SM_Pluto13_Present,INP,101153,boolean,Pluto #13 is present
1,SM_Pluto14_Present,INP,101154,boolean,Pluto #14 is present
1,SM_Pluto15_Present,INP,101155,boolean,Pluto #15 is present
1,SM_Pluto16_Present,INP,101156,boolean,Pluto #16 is present
1,SM_Pluto17_Present,INP,101157,boolean,Pluto #17 is present
1,SM_Pluto18_Present,INP,101158,boolean,Pluto #18 is present
1,SM_Pluto19_Present,INP,101159,boolean,Pluto #19 is present
1,SM_Pluto20_Present,INP,101160,boolean,Pluto #20 is present
1,SM_Pluto21_Present,INP,101161,boolean,Pluto #21 is present
1,SM_Pluto22_Present,INP,101162,boolean,Pluto #22 is present
1,SM_Pluto23_Present,INP,101163,boolean,Pluto #23 is present
1,SM_Pluto24_Present,INP,101164,boolean,Pluto #24 is present
1,SM_Pluto25_Present,INP,101165,boolean,Pluto #25 is present
1,SM_Pluto26_Present,INP,101166,boolean,Pluto #26 is present
1,SM_Pluto27_Present,INP,101167,boolean,Pluto #27 is present
1,SM_Pluto28_Present,INP,101168,boolean,Pluto #28 is present
1,SM_Pluto29_Present,INP,101169,boolean,Pluto #29 is present
1,SM_Pluto30_Present,INP,101170,boolean,Pluto #30 is present
1,SM_Pluto31_Present,INP,101171,boolean,Pluto #31 is present
1,SR_appCRC,IREG,300006,short,PLC application CRC
1,SR_ErrorCode,IREG,300011,short,Error code
1,SR_ErrorLog1,IREG,300012,short,Last error code
1,SR_ErrorLog2,IREG,300013,short,2:nd last error code
1,SR_ErrorLog3,IREG,300014,short,3:rd last error code
1,SR_SupplVolt,IREG,300040,short,Supply voltage (x10 volt)
```

3.2 Data – Panel to Pluto, External Communication

To configure the Pluto to be able to receive data from the HMI panel; use the “External communication” button (red rectangle in the figure below) in the Pluto unit that the serial cable is connected to. *This is non-safe communication*.

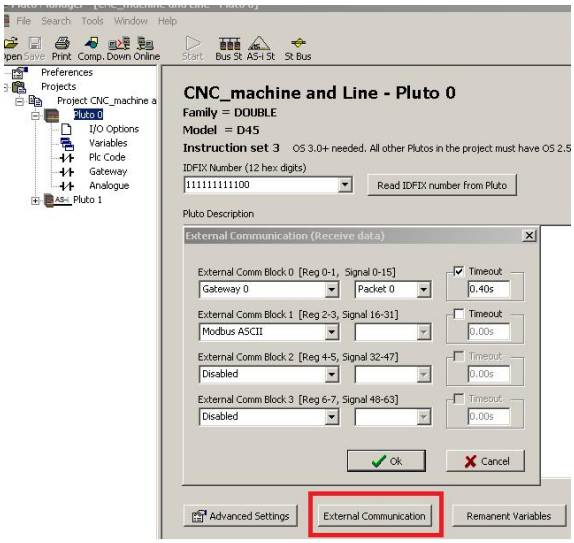

Figure 7

Please note that it is possible to mix both HMI communication and communication with a gateway.

In total it is possible to setup 64 bits and eight 16-bit registers, which can be used for receiving data from an external source.

The “Timeout” check box can be used to zero the receiving “External Comm block…” after a designated time, between 0 – 2.54 seconds, if no new data arrives within that time.

Use this timeout function if you use some sort of keep-alive signal.

3.2.1 Function library

To be able to send variables to the Pluto unit from the panel; the library “Ext01.fps” must be added. This library contains blocks used to receive the data from the panel.

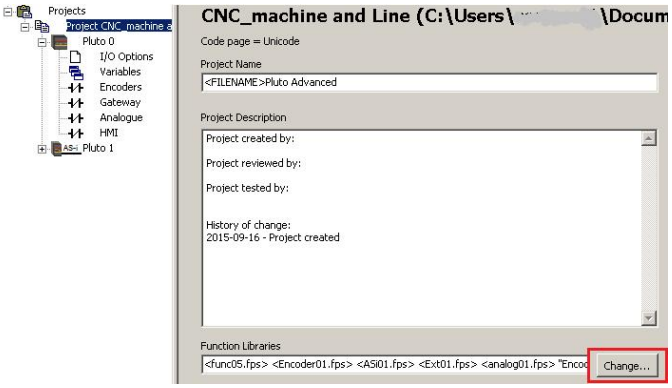

Figure 8

Press the “Change…” button and then the “Add Standard library…” button. Choose “Ext01.fps”.

The contents of the areas are shown in the below picture.

To Pluto Area Packet	Type	Data
0	Bit (16 bits)	Bit variables 0…15
	Register (16 bits)	Register 0
	Register (16 bits)	Register 1
1	Bit (16 bits)	Bit variables 0…15
	Register (16 bits)	Register 0
	Register (16 bits)	Register 1
2	Bit (16 bits)	Bit variables 0…15
	Register (16 bits)	Register 0
	Register (16 bits)	Register 1
3	Bit (16 bits)	Bit variables 0…15
	Register (16 bits)	Register 0
	Register (16 bits)	Register 1

Figure 9

The addressing of areas are shown in the below picture.

Data block	Data in Pluto
External Comm Block 0	Data bit 0…15
	Reg 0
	Reg 1
External Comm Block 1	Data bit 16…31
	Reg 2
	Reg 3
External Comm Block 2	Data bit 32…47
	Reg 4
	Reg 5
External Comm Block 3	Data bit 48…63
	Reg 6
	Reg 7

Figure 10

4 Panel builder 600

The software used to program and setup the CP600 panel series is called “Panel builder 600”. There is also a “Panel Builder 600 Basic”. “Basic” is used for the “eCo” -version CP600 panels.

Both software versions can be used since they both contain the driver needed. The version must be V2.0.0 or newer.

The example project used in this manual were created in the “Basic” version.

Figure 11

Figure 12

4.1 Driver & Software settings

The communication is realized by Modbus ASCII.

Driver settings:

- Baud rate 57600
- 8 Data bits
- 1 Stop bit
- No parity
- No flow control

4.2 Setup of the protocol

Setup the protocol according to the following pictures. In “ProjectView” open “Protocols” and then click the “+” in the tab “Protocols” to add a driver. Choose “ABB Pluto”.

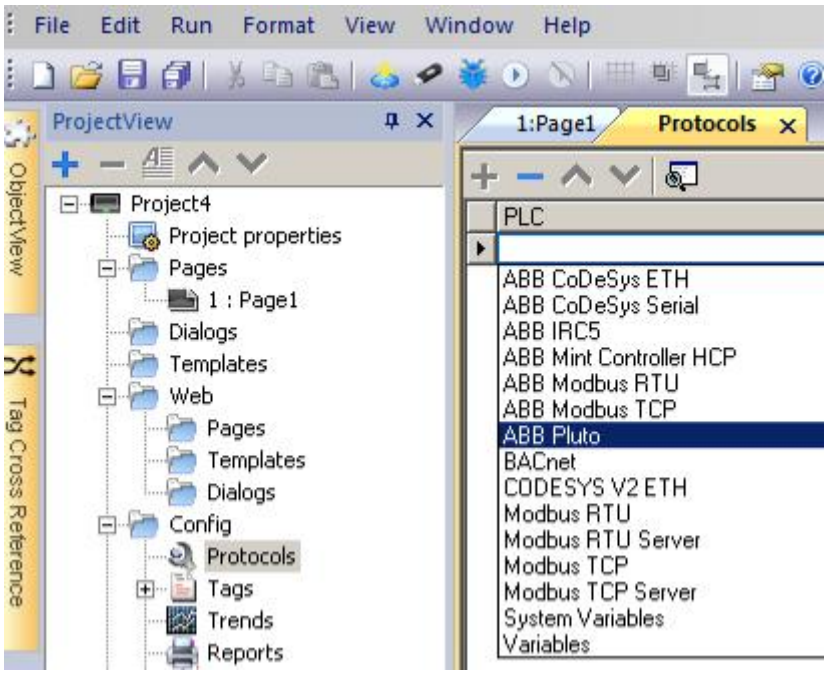

Figure 13

Communication settings can be changed by pressing the “Comm…” button. Normally these settings are not needed to be changed.

The default settings work as they are setup, press “OK”.

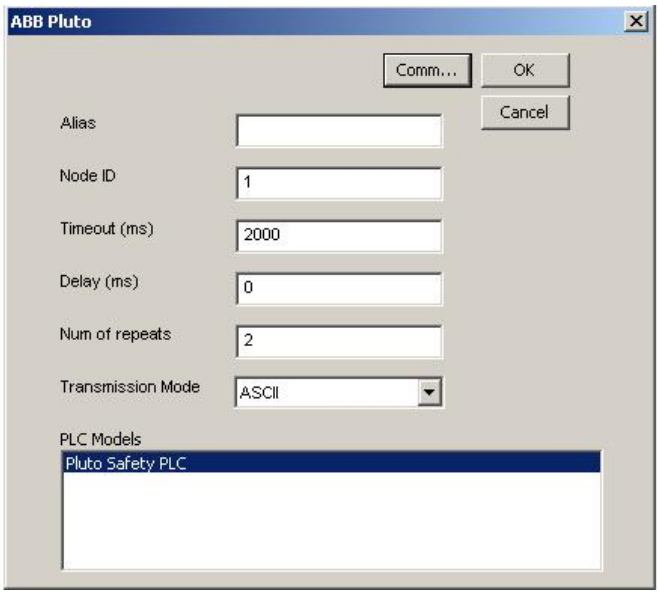

Figure 14

4.3 Importing tags

To import the exported .csv-file from the Pluto project open the “Tags” tab by clicking “Tags” in the “ProjectView”, and press the button that is marked by a red rectangle in the picture below.

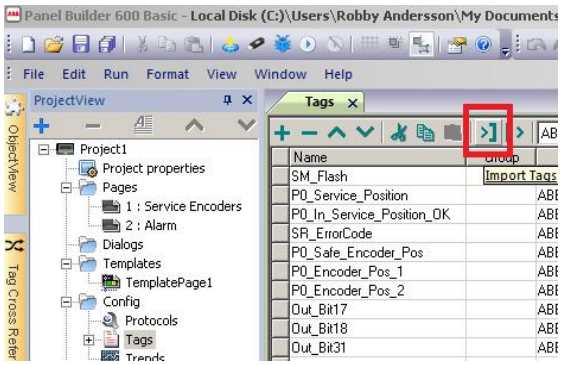

Figure 15

Choose the Linear type as shown in the picture below.

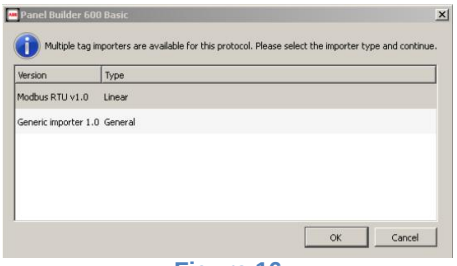

Figure 16

The number of tags imported will be shown. They are all placed in a “Dictionary”, see the “Dictionary” object in the “ProjectView”. It’s possible to choose which of them that are to be used in the panel project as tags. Highlight the tags and press the button marked in red in the picture below.

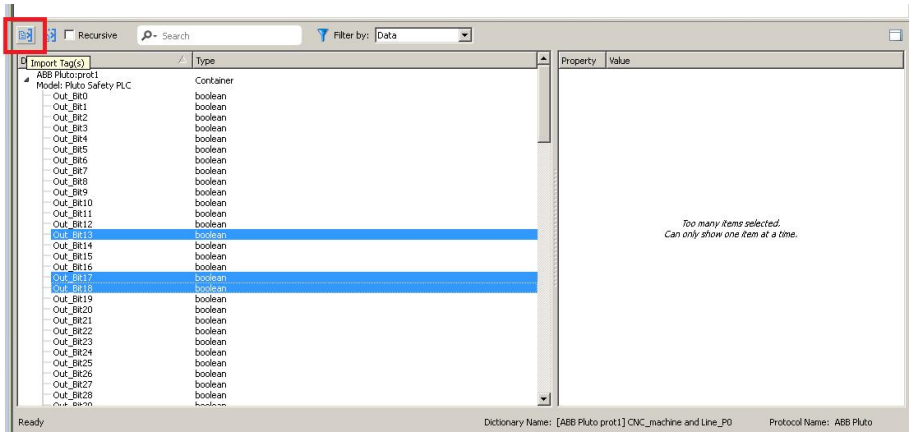

Figure 17

5 Application Examples

Below is a picture of a machine with two emergency stops, encoders, safety mats, and a process lock for the door.

This example will show how to map variables between the Pluto unit and the HMI panel.

This is an example and ABB takes no responsibility for any errors it may contain.

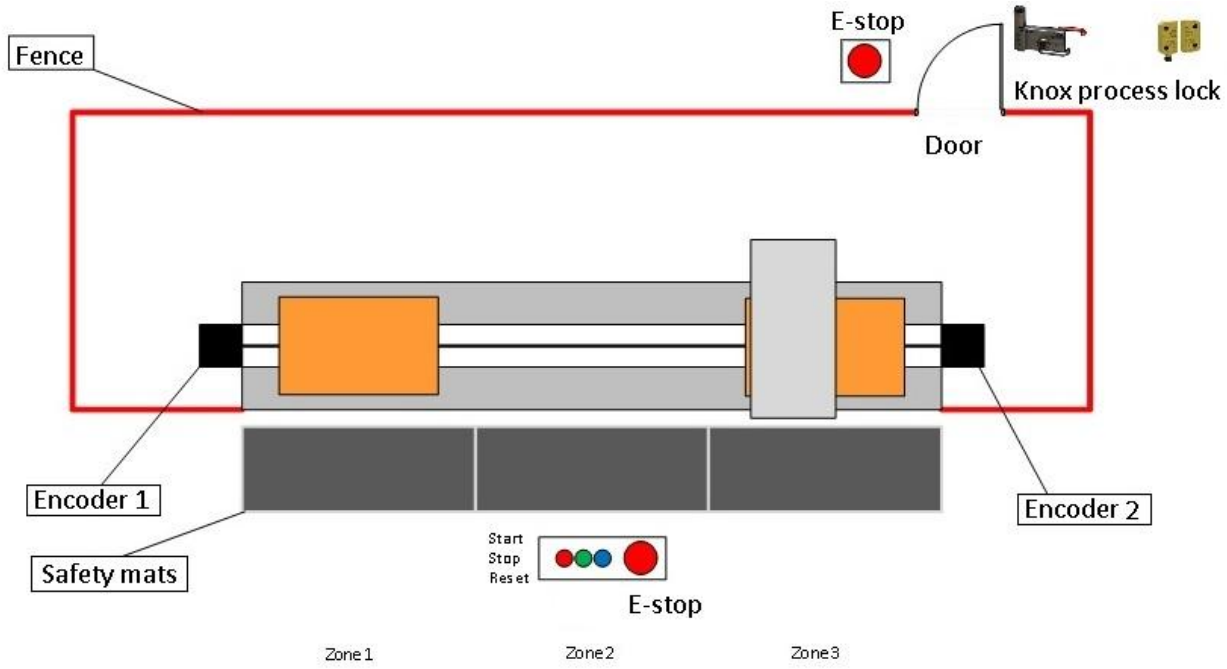

Figure 18

To the HMI panel from the Pluto unit:

- Show on the HMI; which emergency stop has been actuated?
- System errors to be shown in an alarm list in the HMI.
- Show a decimal value in the panel by scaling an integer from the Pluto unit.

To and from:

- What values the two encoders have, what the mean value between them is and during service; zero the position of the encoders.
- Enter a value for number of units produced and read back when the target is reached.

From the CP600 HMI panel to the Pluto unit:

- Control the Knox process lock in the above picture, and a time-out implementation if the panel is disconnected from the Pluto unit.

5.1 Emergency stop shown in panel – Boolean read by panel

In this example an overview of the machine will be shown. Then by using a light-indicator it's possible to see which emergency stop around the machine that has been pressed.

Export the variables intended to be used from the Pluto project as described above.

The picture below show which memory is used in the Pluto project.

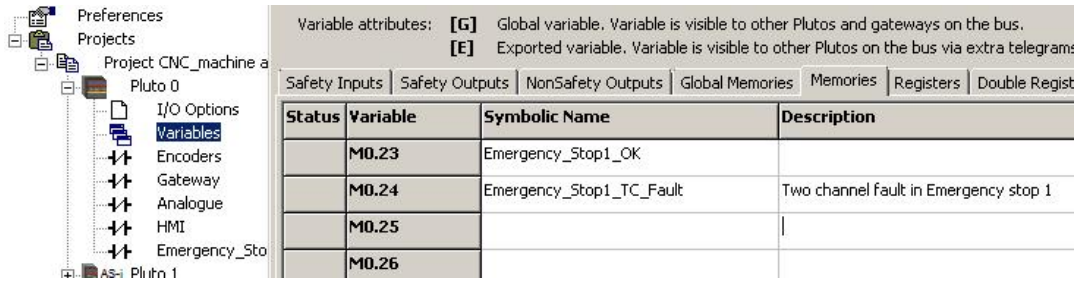

Figure 19

The resulting .csv-file will contain a line similar to this:

1,Emergency_Stop1_OK,INP,101263,boolean,

Import the .csv-file as described above. The result is shown in the picture below.

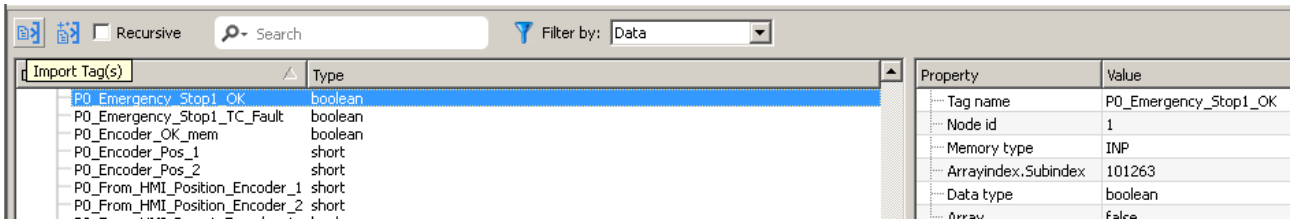

Figure 20

Add the tag to the panel's project as shown in the figure below.

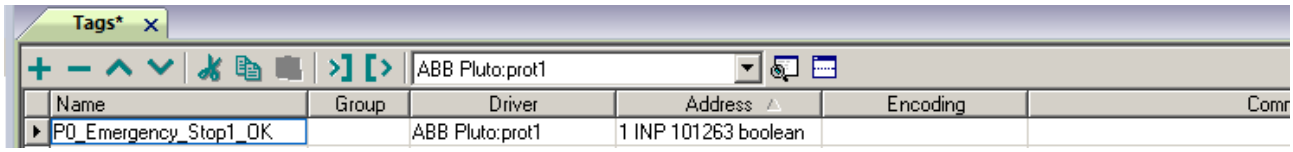

Figure 21

In the picture below an overview of the machine is shown in the panel. For simplicity's sake in this example; a single light-indicator has been added. Several others can easily be added in the same way. Use the "Widget Gallery" – "Lights" and drag and drop the light to the intended placement.

The overview picture was added in the same way but "Basic" – "Images" was used instead. The picture has been moved all the way back so the indicator lights can be placed on top of it and thereby visible.

The light is the red circle highlighted by four black squares. When it is marked in this way it is possible to change the object's properties.

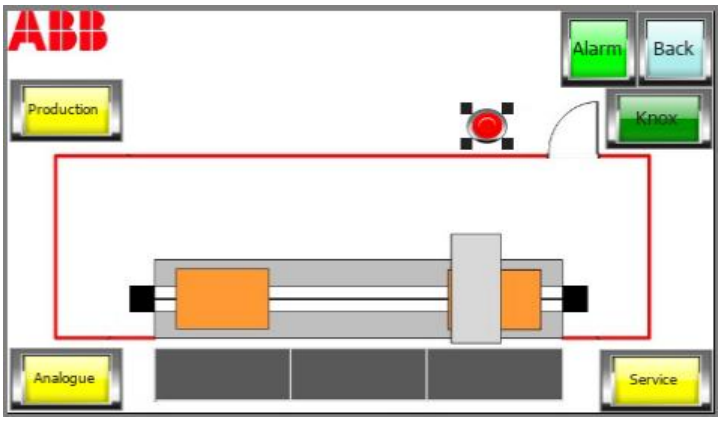

Figure 22

In the properties window for the light widget link the “Value” to the memory used in the Pluto by pressing the small “+”, marked by a red rectangle in the below picture.

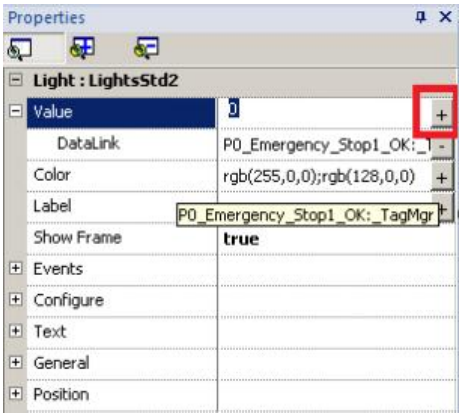

Figure 23

Choose the memory used in the Pluto from the tag-list.

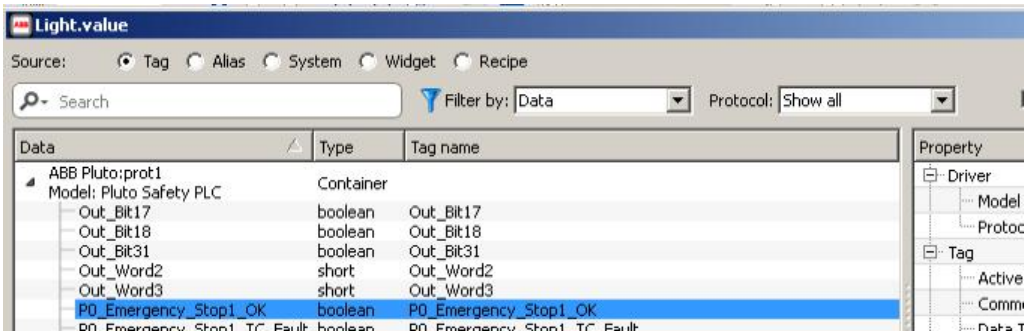

Figure 24

Change the colour of the button so that it's bright red when the tag value is *zero*, and dark red when it's *one*. Since the safe state is zero. See the "Pluto Safety manual" in Pluto Manager.

Perform the colour configuration by pressing the red rectangle in the picture below.

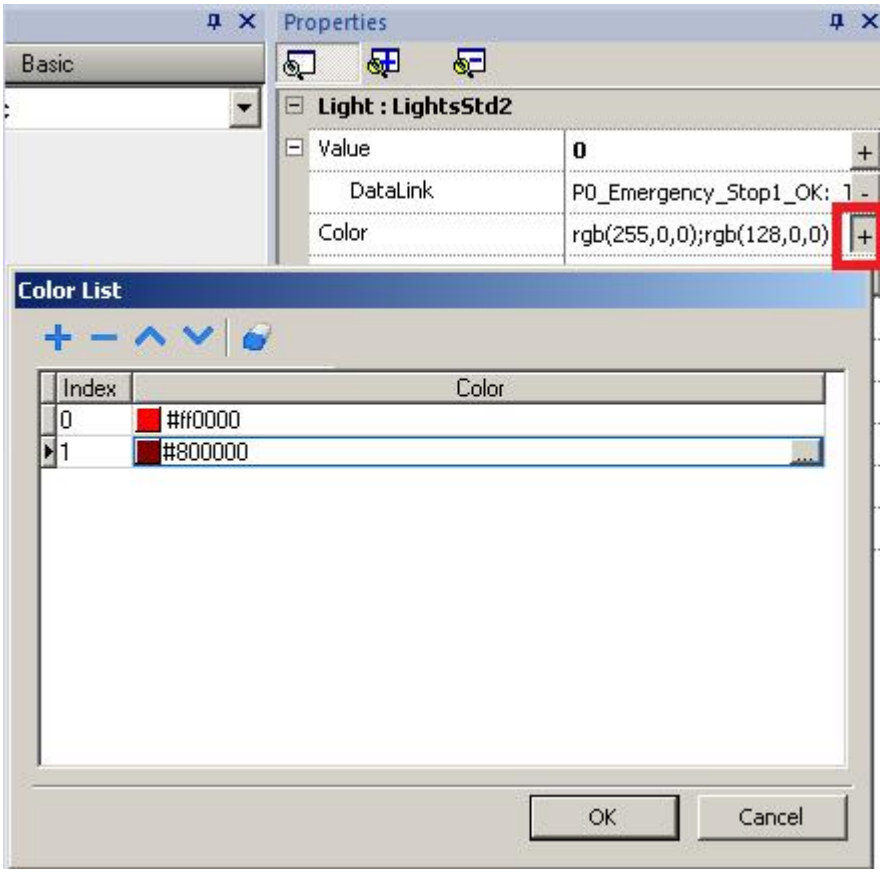

Figure 25

5.2 System error shown in the panel's "Alarm" widget – Register read by panel

In this example a system register in the Pluto unit will be used to show a system error in the panel's Alarm widget.

The System Register in the Pluto unit is called "SR_ErrorCode" and it reports different system errors of different severity. See the Pluto Hardware manual for a complete list of the meaning of the error codes.

Export the variables from the Pluto project as described above.

Since export of system variables and registers is currently not supported; manually add the following line to the resulting .csv-file.

1,SR_ErrorCode,IREG,300011,short,Error code

In the Pluto project, the below picture shows which system register it is that contains system errors in the Pluto. Errors could for example be a CAN-bus fault (Error 18) or an input error (Error 12).

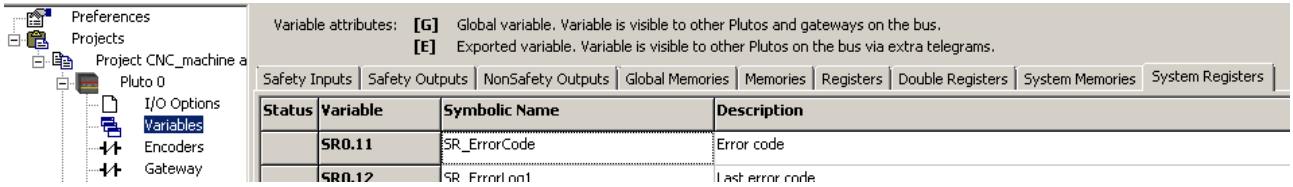

Figure 26

Import the .csv-file into the panel's project as described above.

Create a new page, in this example it will be called "Alarm". When the page is ready; add by drag and drop the "Active Alarms" widget to the page using the "Widget Gallery" – "Basic" and choosing "Alarms" in the drop-down menu. In the below picture the result of a detection of Error 18, CAN-bus fault, is shown.

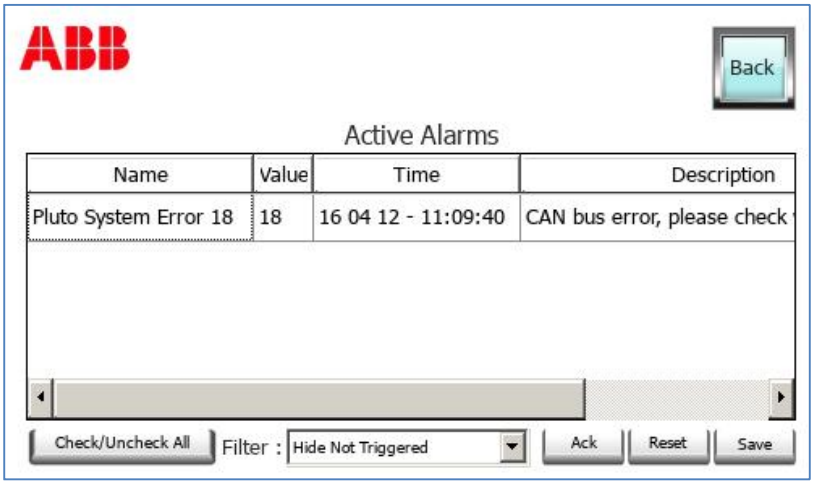

Figure 27

Highlight the widget in the page so that its properties can be changed. In this example some columns that won't be used has been hidden. Configure this by pressing the small "+" marked by a red triangle in the picture below.

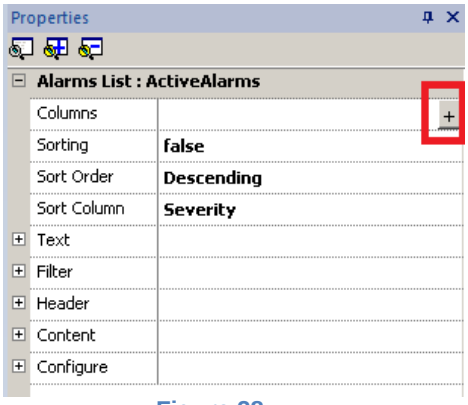

Figure 28

In the pop-up window mark the different columns and change the "Visible" setting from "true" to "false".

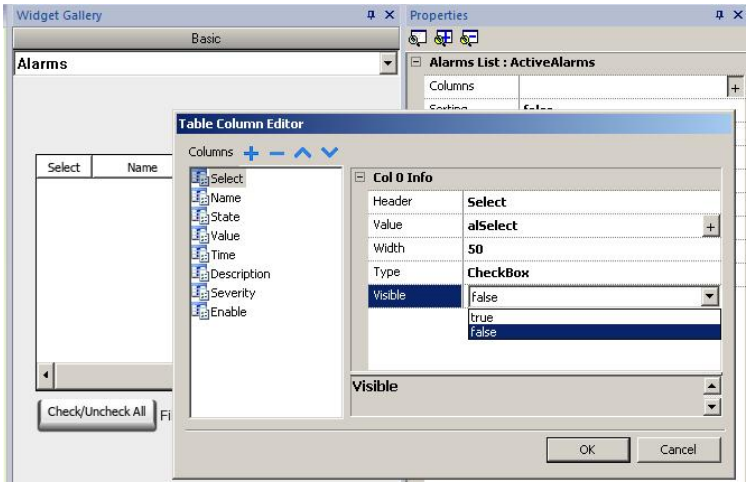

Figure 29

Open the "Alarms" tab, found in the "ProjectView" – "Config". Add a new alarm by pressing the "+" marked by a red rectangle in the picture below. In the "Name" column give it a name suitable for what the alarm will be used to indicate. In this example a CAN-bus error detected by the Pluto unit will be shown; so "Pluto System Error 18" is a good name. In the "Tags" column link it the "SR_ErrorCode" tag that was imported earlier.

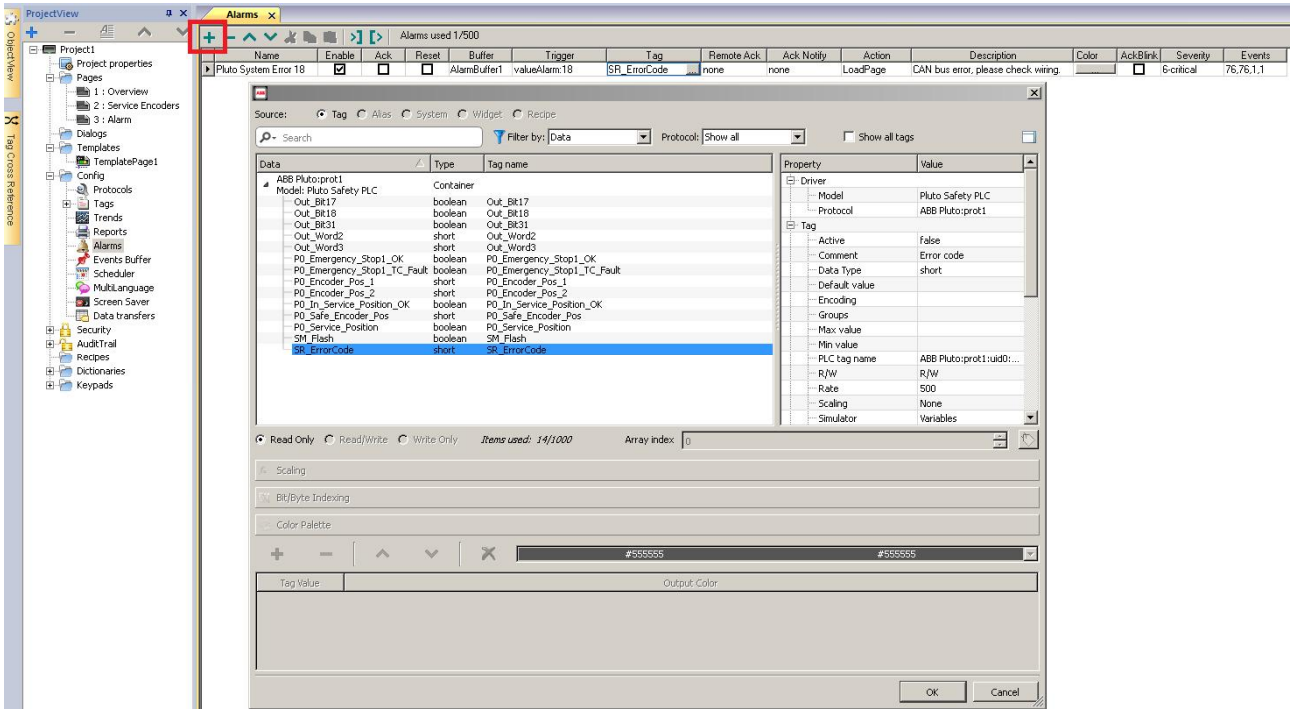

Figure 30

Add a description of the error so it's simple to understand what has happened. In this case "CAN bus error, please check wiring".

This is an example for adding an alarm for the CAN-bus error but adding an alarm for a two-channel fault in the above emergency stop example is done in the same way.

Add a trigger in the "Trigger" column. In this example a value will simply be evaluated. If the Pluto unit's system register is 18, there is a CAN-bus error. This can be changed to a range, deviation, limit, and bit-masking. See the "Panel Builder 600 "help.

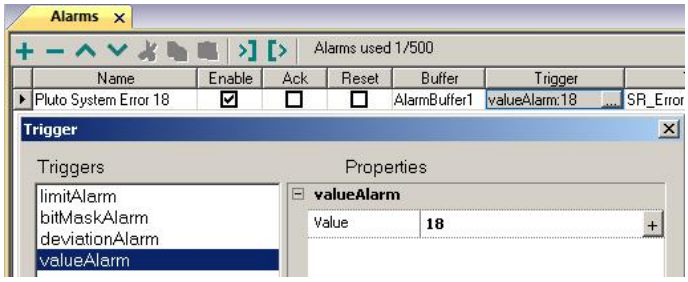

Figure 31

Add an action in the "Action" column – "Load Page". This will make a popup window appear on the panel's screen when a system error occurs in the Pluto. See the picture below. It's possible to add one or several actions happening at the same time.

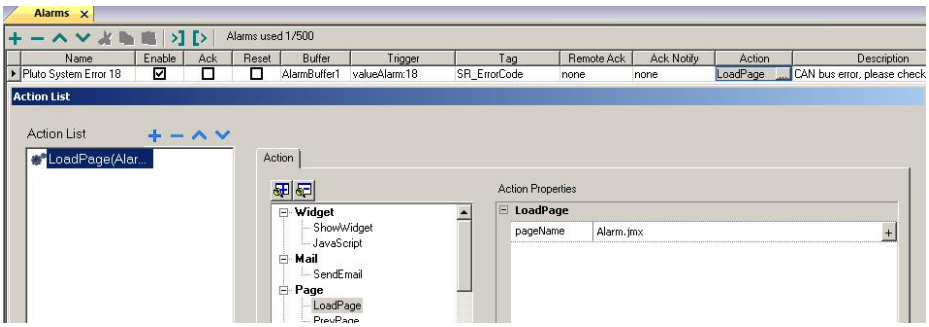

Figure 32

Add a button to those pages where the operator should be able to read alarms. Do so by using the “”Widget gallery” – “Buttons – Standard”. Drag and drop into place on the page. Mark it to be able to change its properties. Change its “Label” to something suitable like “Alarm”. As shown in the picture below.

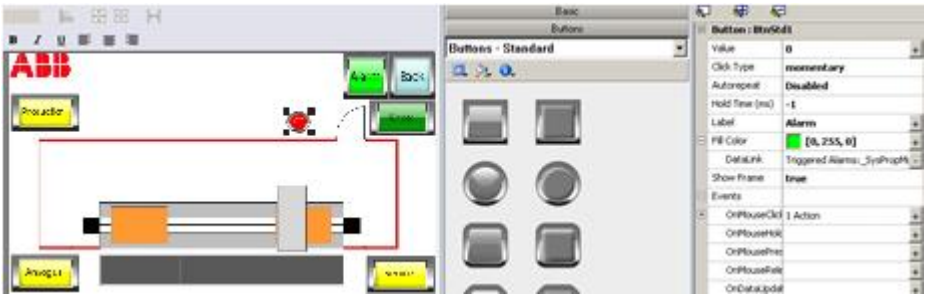

Figure 33

To make the green button appear red when an alarm is active; link the “Fill Colour” property to a system memory of the panel called “Triggered Alarms”. Change the colour depending on if there is an active alarm or not, Tag value “1”, to red, or green if it’s “0”. As shown in the picture below.

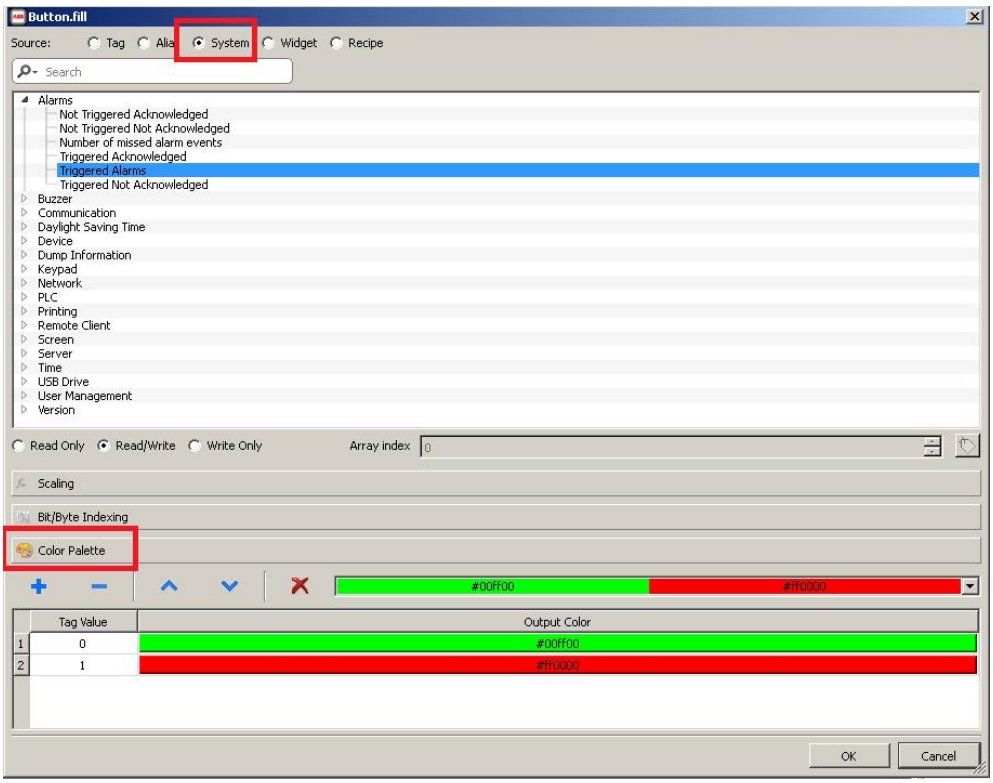

Figure 34

Finally by using “Properties” on the button add “Events” – “OnMouseClick” – “Action” – “Page” – “LoadPage”. Choose the “Alarm.jmx” page for the “pageName”. This will load the page when the button is pressed. See the picture below.

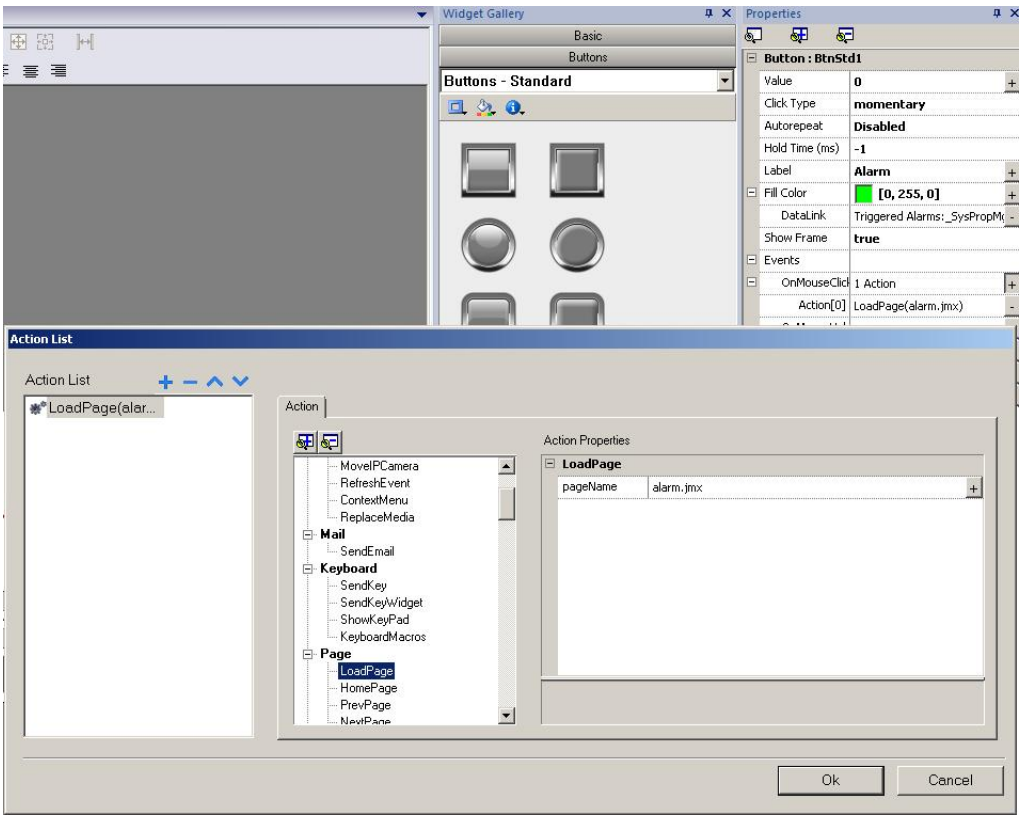

Figure 35

5.3 Service of encoders – Boolean read and write, register read

During operation the absolute encoders' position drift slightly over time. To reset them to a known position zero, the machine's moving part is run to a service position. This service position is monitored by a safe proximity sensor, Eden. The position is thus ensured safely so that the encoders can be zeroed.

This is done by the Pluto unit after receiving a "write" command from the HMI panel.

Add the "Ext01.fps" and "Encoder02.fps" libraries as described above.
The "Encoder02.fps" is available upon request from:
support.jokabsafety@se.abb.com

The data from the HMI in this example is received in the "External Comm block area 1", as shown in the picture. Please notice the numbers inside the brackets. They're used when mapping the receive data to the local variables in the Pluto unit by using blocks.

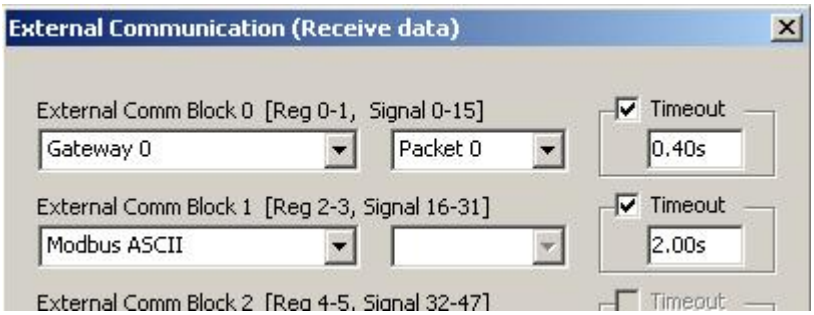

Figure 36

The "Ext01.fps" library contains a number of blocks to realize gateway and Modbus ASCII communication. See Pluto Manager for an in-depth description of these blocks.

In the below picture part of the project is shown. It contains block needed to map the receive data in the picture above to variables in the Pluto unit.

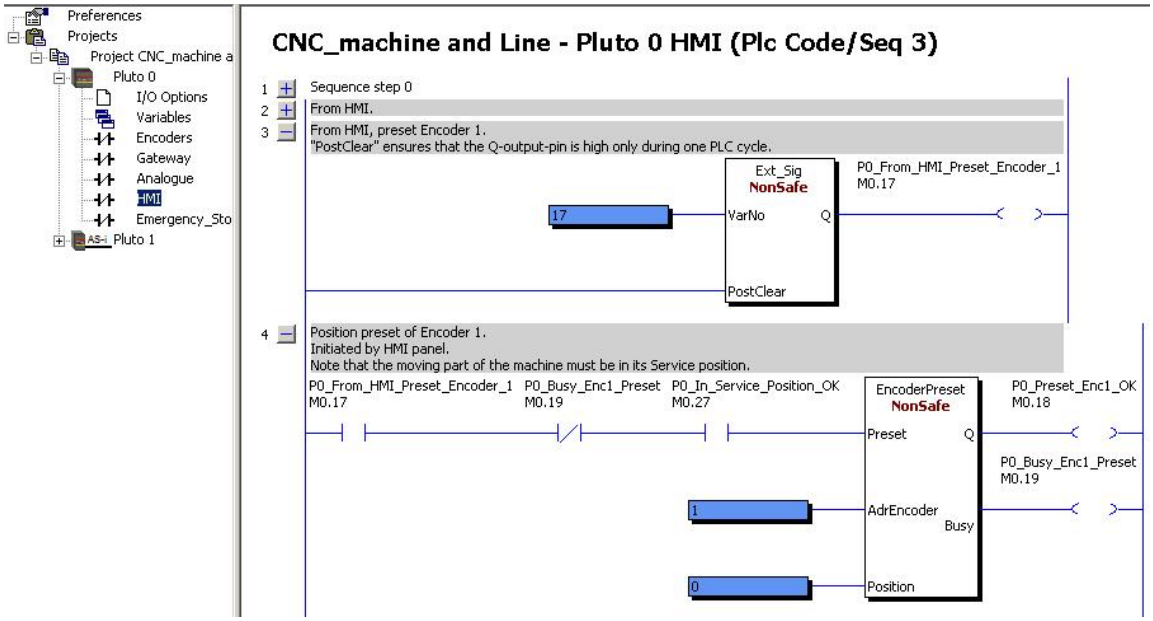

Figure 37

The Pluto project's PLC code is split up into different parts in this example by adding and naming sequences. See the Pluto programming manual how to do this.

The block "Ext_Sig" maps a Boolean variable to a local memory bit. In this example this bit will be used to initiate the write of a zero value to an absolute encoder addressed as 1.

The block "EncoderPreset" is the block that performs the preset of the absolute encoder's position. In this example it's initiated by a button in the panel, if the machine's moving part is detected to be in its safe service position.

Export the variables from the Pluto Manager project as described above. The resulting .csv-file will contain the following lines, among others. Import the .csv-file in the Panel Builder 600 and start using them as tags, see above.

1,P0_From_HMI_Preset_Encoder_1,INP,101257,boolean,
1,P0_In_Service_Position_OK,INP,101267,boolean,
1,P0_Encoder_Pos_1,IREG,300102,short,
1,Out_Bit17,OUTP,17,boolean,External Comm block 1 bit 1

In the Pluto project this corresponds in the same order as:
M0.17
M0.27
R0.2
Bit 17 in the receive area of the Pluto, see the picture above (External communication, inside the brackets).

In the panel's project a page is created called "Service Encoders". It contains buttons and indicator lights, and register values being shown.

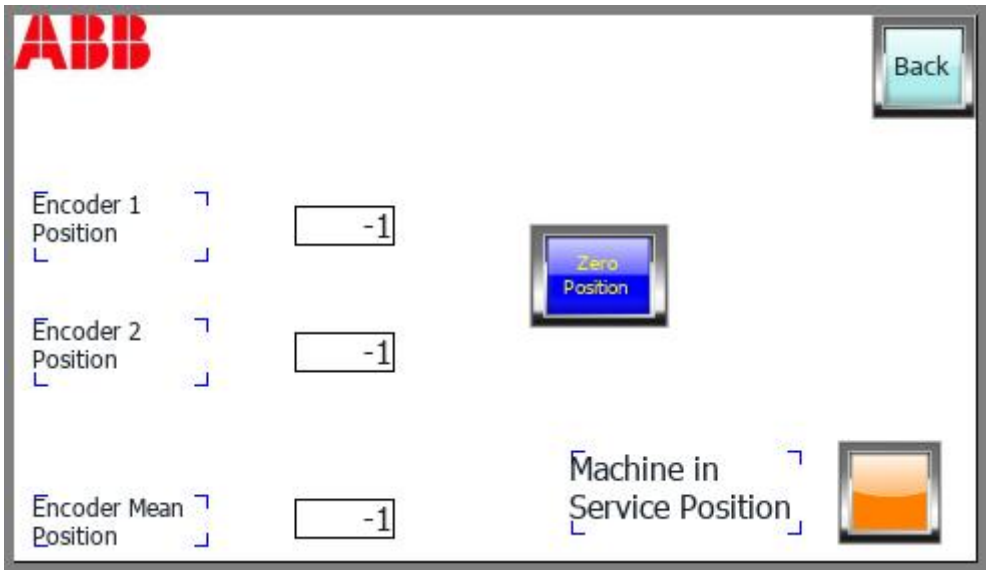

Figure 38

Use the "Widget Gallery" – "Basic" and in the drop-down menu choose "Text/Numeric" to drag and drop "Label" and "99999" to add text as information and, the decimal values intended to work with.

Add the orange indicator light as done in the “Emergency stop” example above. Link it to “1,P0_In_Service_Position_OK,INP,101267,boolean,”, M0.27 in the Pluto unit. It will simply indicate when it’s ok to attempt to enter a new position for the encoders. The Pluto project will block writes unless this condition is fulfilled, as can be seen above.

To link absolute encoder one’s position to a field, highlight the field and press the small “+” marked by a red rectangle in the below picture, select the tag. Check that it’s entered as “Read only”. The property “Keypad” is also changed to “None”.

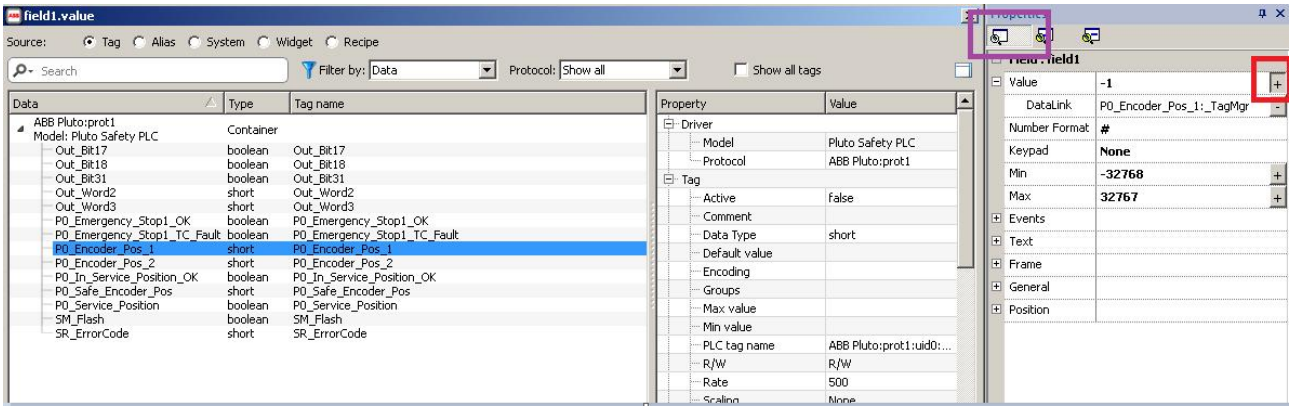

Figure 39

In this example the property “Value” was changed from “99999” to “-1”. Also, by using the button highlighted by a purple rectangle a frame was added by changing the variable “Frame” – “Show Frame” from false to true. “Keypad” is set to “None” to prevent input.

The blue button highlighted in the picture below is then used to write a command to zero the two encoders’ position in the Pluto unit.

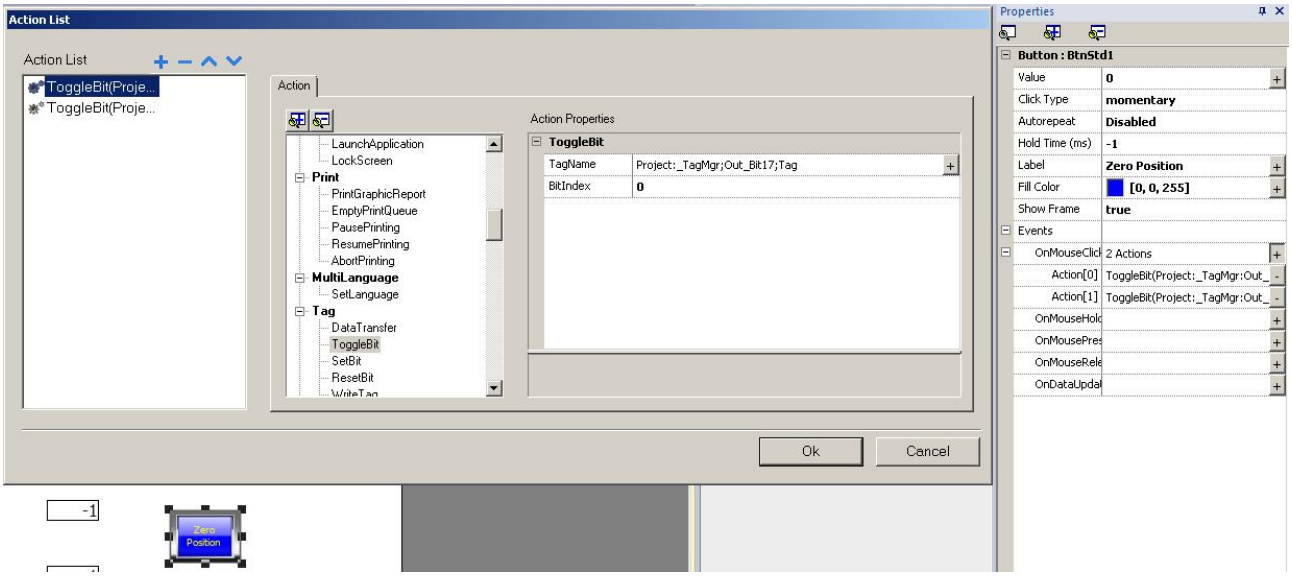

Figure 40

This is done by working with the “Events” property of the button. By clicking on the “+” of the “Events” – “OnMouseClick” four “Actions” are added. The actions are two “Tag” – “ToggleBit”. In this example only one encoder’s preset operations tag is shown but the same procedure is used for both.

To perform a Boolean write to the Pluto; “ToggleBit” is used to toggle a bit in the Pluto unit used to initiate the zeroing of the absolute encoder’s position.

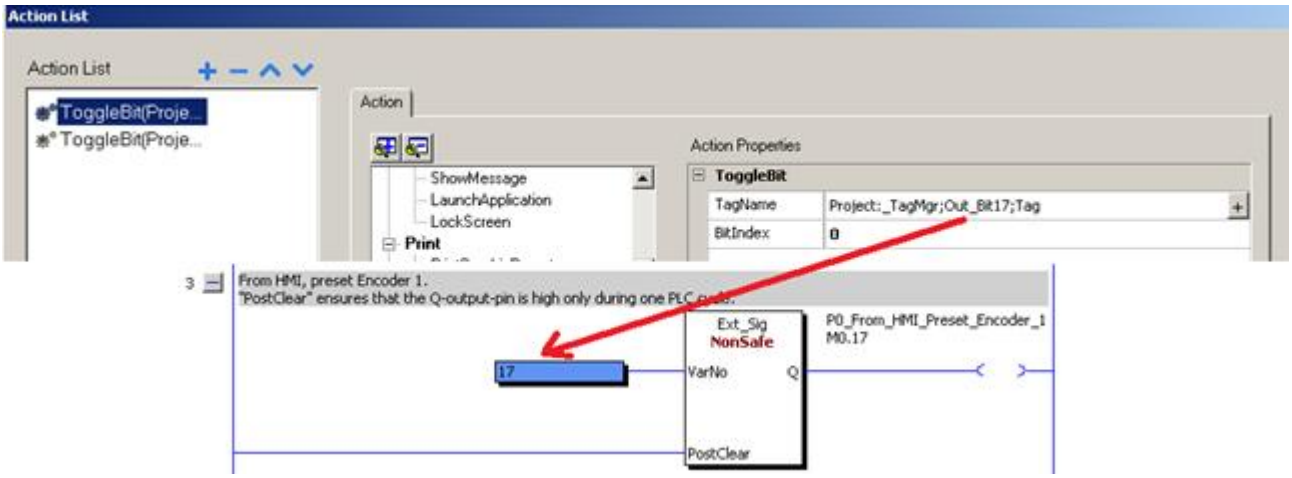

Figure 41

In the “Ext_Sig” block the “PostClear” pin is connected and always true. This means that the next PLC cycle the “Q” output is zeroed. In this example it ensures that the zeroing-operation isn’t spammed.

5.4 Production control – Register read and write

The panel will be used to set a number of units that the operator wish to produce. The Pluto will count how many units is has produced, and when it reaches the number the operator has entered it will stop production. The operator can see in the panel how many has been produced so far. It is also possible to zero the number of units produced counter to start a new series.

How this is handled in the Pluto project is not shown here, only the communication between the panel and the Pluto unit.

See the above example of Service of Encoders for how to add buttons, label and numeric fields, and how to attach tags to them. It will use the same External communication area as well.

The following exported variables are used.

1,P0_Units_produced,IREG,300109,short,
1,Out_Bit19,OUTP,19,boolean,External Comm block 1 bit 3
1,Out_Word2,HREG,400002,short,External Comm block 1 reg 0

In the Pluto project this corresponds in the same order as:
R0.9 Attached to "Produced number of units" field in the below picture.
Bit 19 in the receive area of the Pluto, see the picture in the above example (External communication, inside the brackets).
Reg 2 in the receive area of the Pluto, see the picture in the above example (External communication, inside the brackets).

The new addition in this example is the "Wanted number of units" field. It has been enabled for input and is connected to "Out_Word2". Highlighted by the black squares in the picture below.

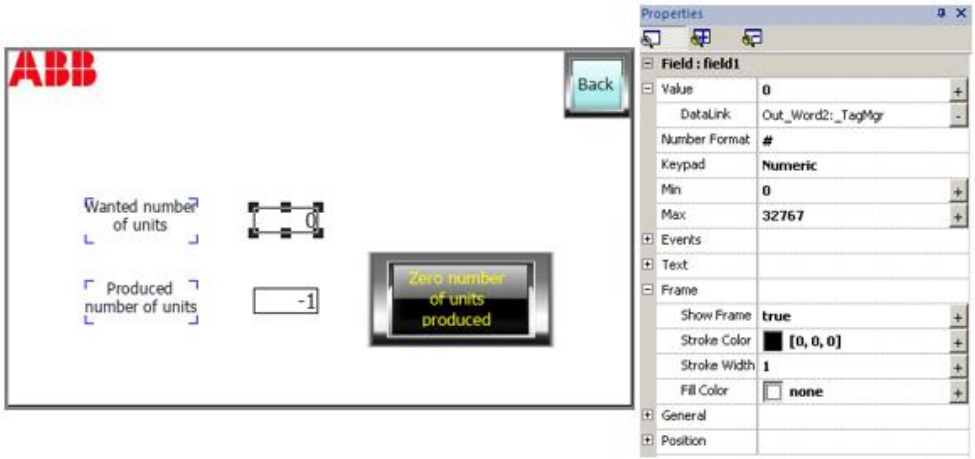

Figure 42

Please remember to make the tag "Read/Write" (red rectangle) when attaching it.

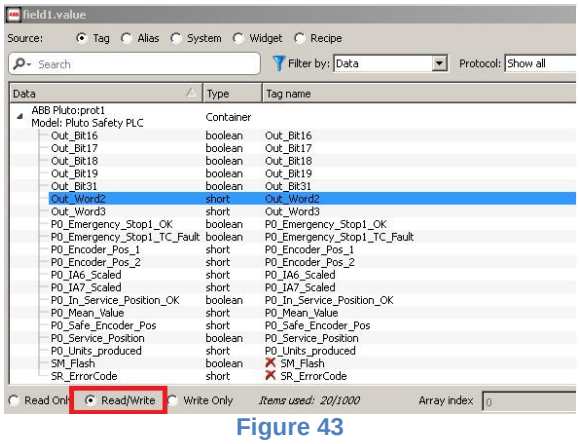

Figure 43

The field is linked to “Out_Word2”. It has a numeric keypad and a min and max value. The number format (#) ensure only integers will be the result when entering a number.

In the below picture it is shown how the variables are linked in the Pluto project.

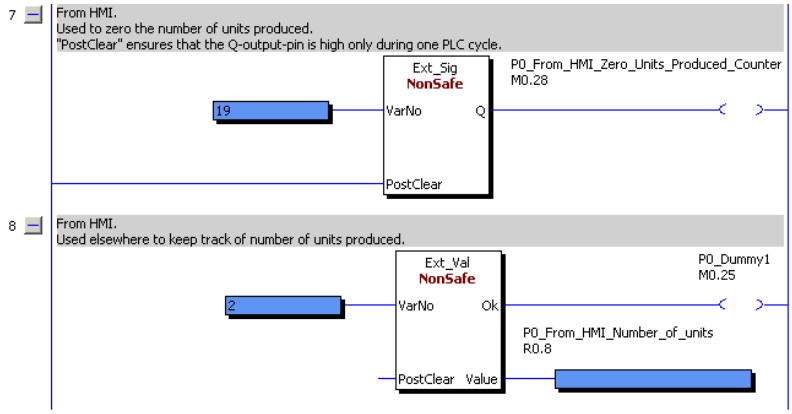

Figure 44

Please note how the “PostClear” input is used differently between the Boolean signal (Ext_Sig) and the register (Ext_Val).

In the Boolean it used to clear the Q-output the following PLC cycle so that it doesn’t spam the zero units produced signal.

In the register it is left unconnected, this means the value will be kept until an update is sent or if a time-out occurs. See the following example.

5.5 Lock function of the Knox process lock – Boolean write & Time-out

There is a Knox process lock which will be controlled by the panel. It works like so that as long an ON-signal is maintained it will be locked. If the locking signal goes to OFF it opens. However, it's bi-stable, so during a power failure it will stay locked. See its manual for an in-depth description.

Export the variables from the Pluto project and import it into the panel project as described above.

As in the earlier examples "Comm Block Area 1" will be used for the panel communication.

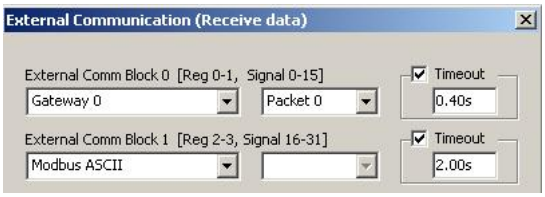

Figure 45

Create a button and link "Out_Bit16" to it. Do so by drag and drop it into place from the "Widget Gallery". Then highlight it and use "Properties". Click the small "+" on the "Value" property and choose "Out_Bit16.

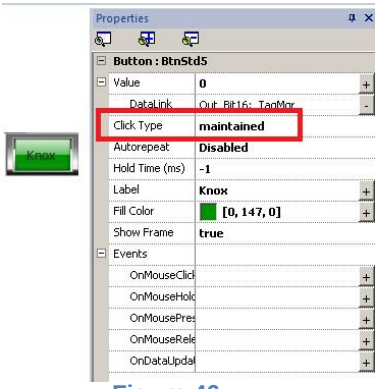

Figure 46

Then change "Click Type" to "maintained". This means the button is shown as pressed/activated until pressed again.

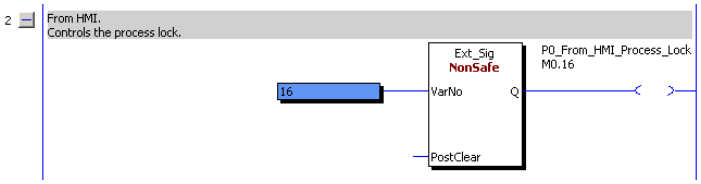

Figure 47

In the Pluto project the "Ext_Sig" block will map to a local memory that is then used to control the Knox process lock elsewhere in the PLC program.

If the signal is somehow interrupted, for example either by zeroing it by using the "PostClear" input of the block shown in the picture above, or by a time-out described above, the button will be returned to its non-pressed state. The panel will detect the change-of-state. This only works if they are in the same "External Comm Block…" area.

By ticking the time-out box, and setting a time in the “External Comm block”, it’s possible to zero the receive data if no new data arrives within the time set.

Figure 48

To realize this function a system memory in the Pluto unit is used, called SM_Flash. See the picture below where it’s located in the Pluto unit.

Figure 49

Export variables from the Pluto project as described above. Since export of system memories isn’t supported; manually add the following line to the resulting .csv-file.

1,SM_Flash,INP,101042,boolean,Flash 0.4s/0.6s (on/off)

Import the .csv-file and add the tag to the panel’s project as described above.

Since the default update rate is too slow to read the “SM_Flash” bit reliably the poll-rate must be changed from the default 500 ms to 100 ms. Compare the rates of the system register “SM_ErrorCode” and “SM_Flash” shown by the purple rectangle in the picture below. Change the rate by pressing the button marked by the red rectangle and enter 100 for “SM_Flash”.

Figure 50

Open the “Data transfers” tab that is found in the “ProjectView” of the panel’s project. Map the “SM_Flash” to “Out_Bit31” in this example as shown in the below picture.

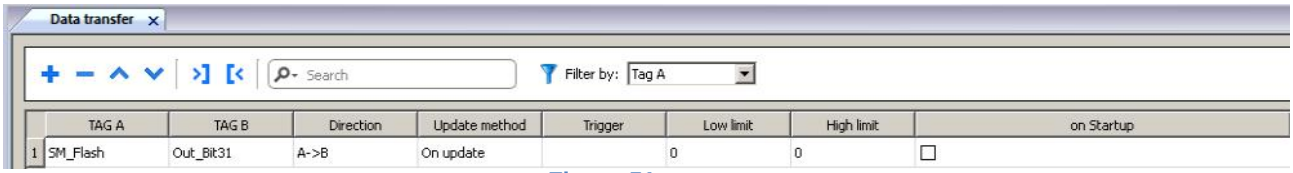

Figure 51

This will simply mirror the status of the “SM_Flash” back to the Pluto in a “keep-alive” function. If the cable between the Pluto and the panel is disconnected, the Pluto unit’s reception area will be zeroed. Please note that bit 31 of the “External Comm Block 1” here is spent, it can’t be used for anything else, but it doesn’t need to be mapped to a variable in the Pluto unit.

5.6 Analogue decimal value shown in panel – Register read and scaling

In the machine a temperature is monitored by two analogue sensors and evaluated by the Pluto unit. This temperature will be shown on a page in the panel. Similar to the two absolute encoder example above the two analogue sensors values will be shown, and a safe mean value of them. What's different in this example is that a decimal value will be presented according to how the value is scaled by the Pluto unit.

Below is a picture of the Pluto PLC code shown as being monitored online. The two sensors are each evaluated by a block which perform scaling of the input signal. The last network takes the two sensors scaled values and calculates a mean of them safely. See Pluto Manager for an in-depth explanation of the blocks.

Please note that the "ReadCurrent" blocks does read values down to zero. This is because some sensors produce a current below 4 mA but above 0 mA when in a fault state.

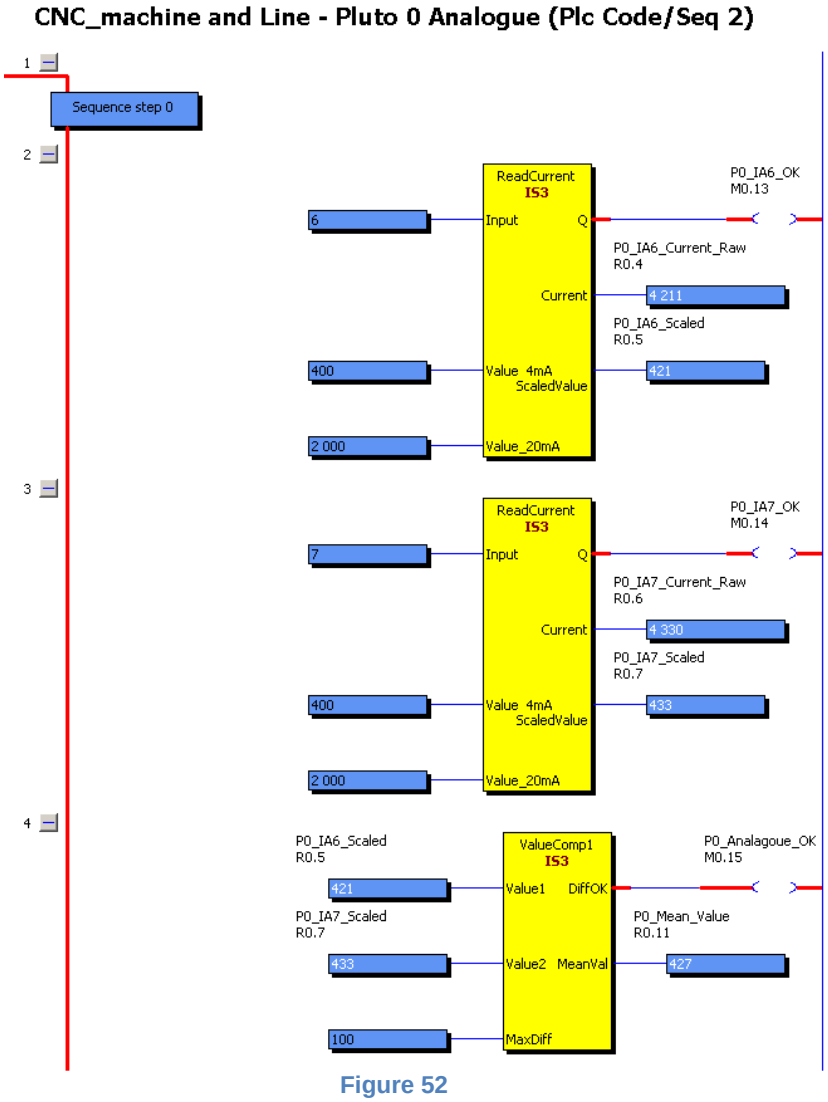

Figure 52

Export the variables from the Pluto project as described above. Import the resulting .csv-file as described above.

The following tags will be used in this example:

1,P0_IA6_Scaled,IREG,300105,short,
1,P0_IA7_Scaled,IREG,300107,short,
1,P0_Mean_Value,IREG,300111,short,

A new page is created called “Analogue”. Drag and drop from the “Widget Gallery” – “Basic” – “Text/numeric” three labels and three fields, as shown in the picture below.

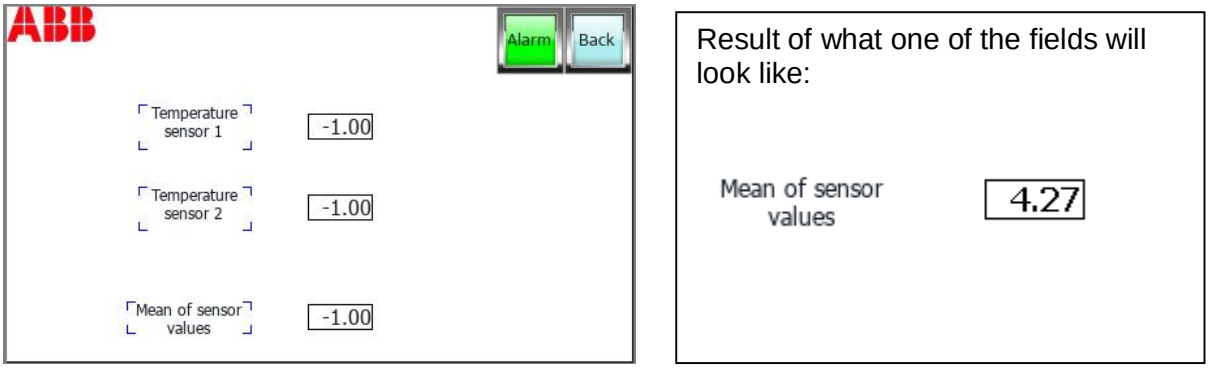

Figure 53

Link the fields with the tags above. Each numeric field is the highlighted and its properties are changed. The properties in the below picture are shown for the highlighted field in the picture above.

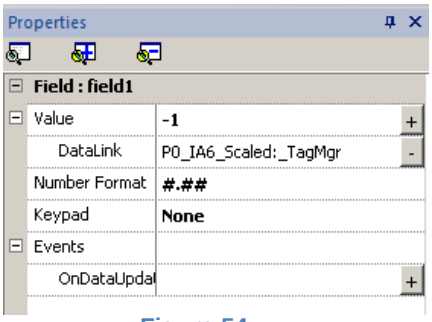

Figure 54

To show the value as a decimal value instead of the 0 to 2000 scaling by range is used when the “DataLink” property is added by pressing the small “+” symbol. See the below picture for the popup input window.

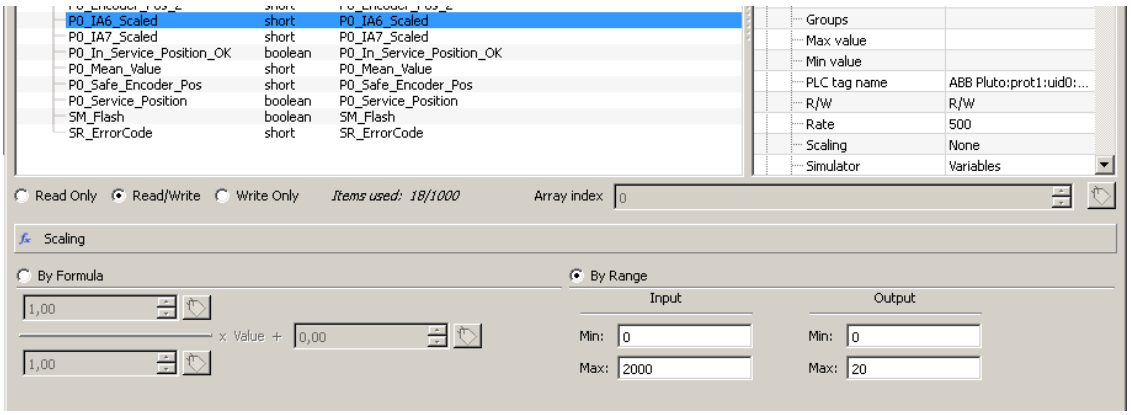

Figure 55