Griffith School of Engineering

Project 1

Drink Filling Process

Contents

1. Introduction

Aim

The aim of this project is to become familiar with the SCADA hardware and software package. This will be achieved by implementing a simplified scenario of a soft drink filling process.

Background

Supervisory control and data acquisition (**SCADA**) is a control system architecture that uses computers, networked data communications and graphical user interfaces for high-level process supervisory management. It can also utilise other peripheral devices such as a programmable logic controller (PLC) and discrete PID controllers to interface with the process plant or machinery. The operator interface is used for monitoring and the issuing of process commands, such as controller setpoint changes. Through the monitoring of operations, faults or bottlenecks in the supply chain are easily identifiable allowing for plant operators to make appropriate changes to maximise production efficiency and increase profitability. The utilisation of a SCADA system also allows for accurate tracking of factory products, this is particularly important when product needs to be recalled.

In this project, you will utilize two Schneider Electric programs.

- RemoteConnect: This is an all-in-one software tool used to configure and program the SCADAPackx70 range. It allows the user to write a program that can then be transferred to the SCADA pack hardware either locally, using any communication port or remotely through communication devices like serial *modems*, Ethernet routers or serial/Ethernet radio units.
- ClearSCADA this program allows the user to set up a user interface or mimic for viewing values and implementing commands.

2. SCADA Board Inputs and Outputs

Digitals Inputs (DI) & Outputs (DO)

A digital signal is used to denote items that have only two different logic states, ON (binary 1) or OFF (binary 0). In a SCADA system, the digital input port allows for the detection of information from the monitored environment. Similarly, the output port is used to produce signals that can control aspects of the said environment. For example, a digital input signal could be used to indicate whether a valve is open or closed. Whilst, the output signal could be used to open or close that same valve.

The SCADA pack that will be used in this project has 24 digital ports. 16 of which are inputs, B1-B8 are buttons and S1 to S8 are switches, and 10 outputs connected to LED's, D1 to D10. It is important to note that the buttons can be set to trigger on the rising edge, falling edge or both (see Figure 1 for image).

Analogue Inputs (AI) & Outputs (AO)

Analogue signals are variable, they have many states. Analogue input signals can be used to detect things like flow rate, temperature or in the case of our valve example it could be used to indicate the position of the valve i.e. half open, quarter open etc.

Analogue output signals are much the same except they are used to control machinery instead of measure. For instance, a variable voltage signal could be used to control how fast an electric motor rotates or in the case of the valve how quickly it opens and closes.

The SCADA pack that will be used in this project has eight analogue ports, six variable input ports (three sliders and three pots), plus two output ports connected to LED ladders (see Figure 1 for image).

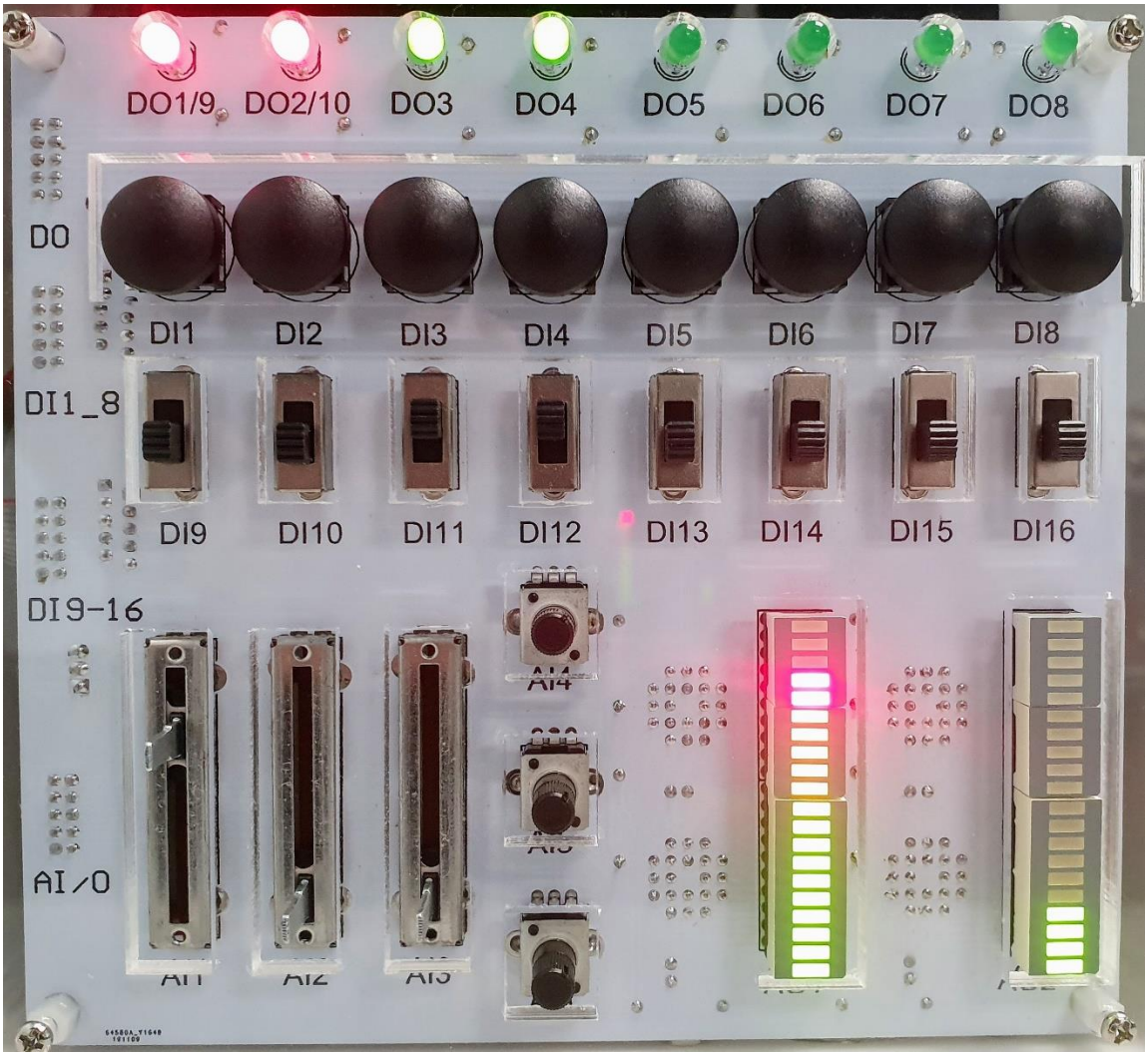


Figure 1: SCADA in/out board

In our daily lives, we mainly use analogue signals. However, all analogue signals include a digital component, like a valve is 0%=closed or 100%=open. An analogue signal can only be read by an analogue I/O, but a digital one can use either analogue or digital I/O's. But in practice, you never connect a digital signal to an analogue I/O because the analogue I/O's are more expensive than digital one. In some situations, it is possible to convert an analogue signal into a series of digital signals. This is done because digital signals are more suited to computers (PC/PLC, etc.)

As rule of thumb,

- Whenever possible use a digital I/O, since it's cheap.
- Use an analogue I/O if the option to convert the signal in a series of digital signals isn't possible.

2.1 - Default Factory Setup Instructions: Set SCADAPack x70

Reset factory setting on the SCADAPack x70 controller using SELECT button

1- Ensure that the SCADAPack x70 controller is powered down (unplug the device).
2- Press and hold the SELECT button.

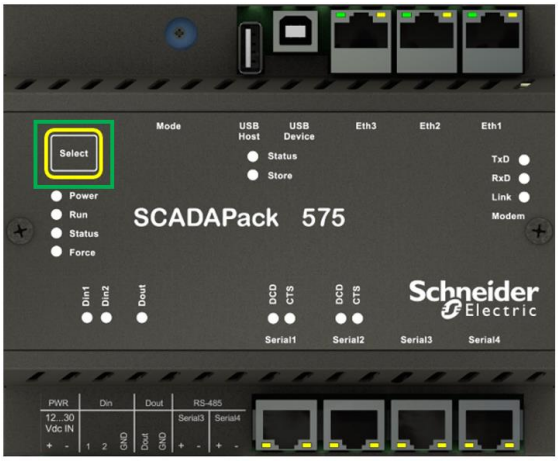


Figure 2. Top view of the SCADApack 575.

3- While holding down the SELECT button, power up the SCADAPACK X70 controller and observe the STATUS LED.
4- The STATUS LED will go through the following sequence while the SELECT button is held down.

Time Approx.	STATUS LED	Description
0 Sec	OFF	Run Mode
10+ Sec	ON	Service Mode
20+ Sec	Blink	Cold Boot Mode
30+ Sec	ON	Factory Boot Mode

Note: Hold it for a few seconds after the second set of the Status LED Solid on, to make certain of the reset being activated.

5- After releasing the SELECT button, observe the RUN LED.

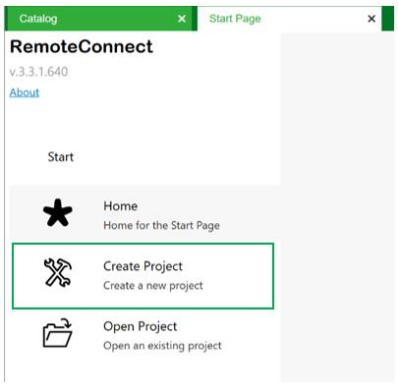
The RUN Led will go through a flashing sequence and once the SCADAPack x70 controller is ready it will flash 1.5sec ON and 1.5 sec OFF to indicate that the controller is in run mode.

2.2 -Remote Connect

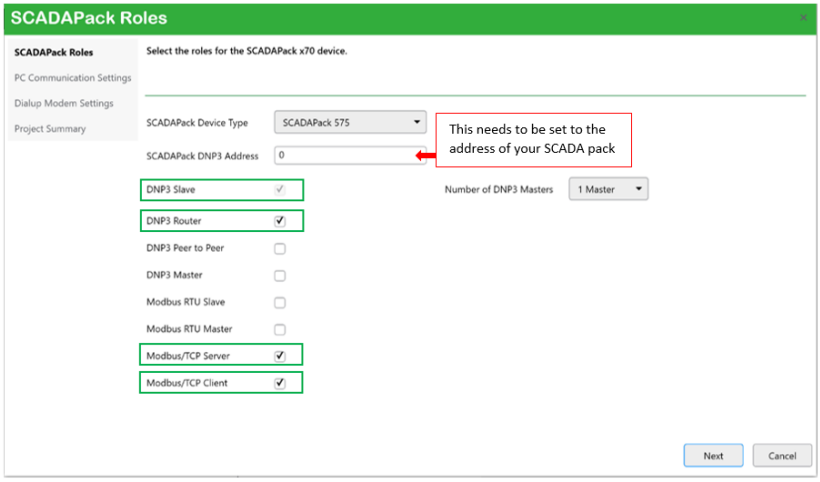
Create New SCADAPack x70 Project

NOTE: Do not have USB connection cable plugged into computer for Initial project creation.

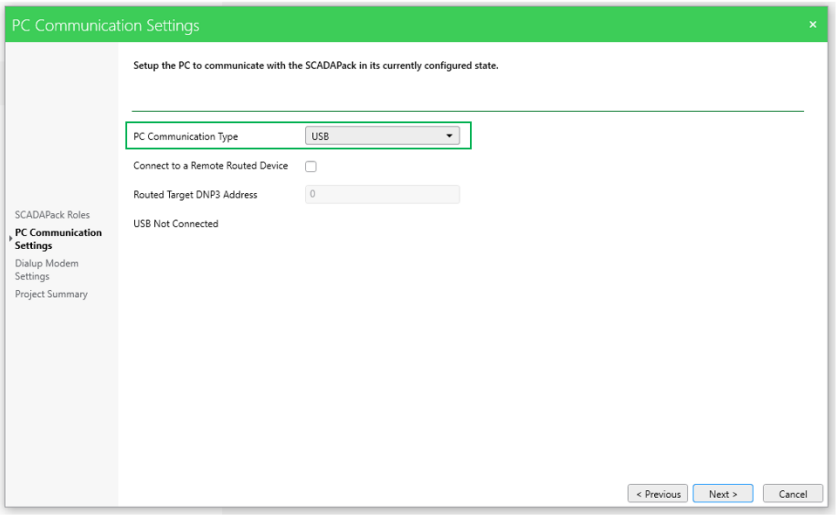
1- Launch Remote Connect from the windows start menu or desktop shortcut.
2- From the Start Page Tab click Create Project.



3- Complete the SCADAPack x70 roles page as follows: DNP3 Router ticked, Modbus RTU Slave unticked, MODBUS/TCP server and client ticked.
4- Set the SCADAPack DNP3 Address: This should match the RTU's address written on the unit. In this example it was 1 (each station will be different). Click Next



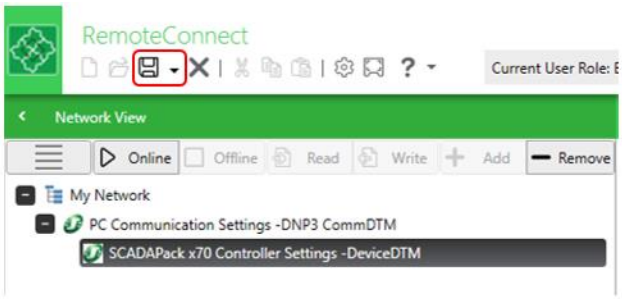
5- Complete the Communication Settings page as follows: Set to USB, Click next.



6- The project summary page is displayed. Important information displayed will include PC Communication Settings, and Controller Role Settings. Click Finish

7- Creating the project message will appear while the project is getting created. After the project is created, a new section called network view is displayed on the left-hand side.

   Network view contains the following objects:

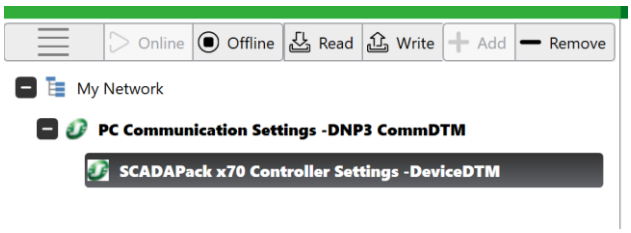


8- Save the project in a new folder. The save icon can be found in the top left corner.

Note: If using University computers do not save to desktop, save it to your Student drive.

Configuring USB Connection.

1- Connect the SCADAPack. Plug in USB cable to USB device on the SCADAPack x70 and connect to the PC.
2- From the Network View Tab right click the SCADAPack x70 Settings- ControllerDTM and select Go Online. Alternatively, select SCADAPack x70 Settings- ControllerDTM and click the online button in the above ribbon.
3- Once connected online both the 'PC Communication Settings- DNP3 CommDTM' and 'SCADAPack x70 Controller Settings – DeviceDTM' text will turn bold.



Note: If there are any issues going online load CMD prompt and use the command of Ping 10.2.3.4 (default IP address given by the SCADAPack USB plug) to verify there is a response from the SCADAPack.

Online Status- SPx70 Controller

1- Once connection is established, the Online Diagnostics will open (If it doesn't right click on SCADAPack x70 Settings- ControllerDTM and select online diagnostics).
2- The SCADAPack x70 Diagnostics Tab has 3 sub-Tabs. (Status, Logic and Objects). As this is the first time the device has been connected, the Tabs will show no information. To retrieve the information and confirm connection.
   a. Select the Status Tab: Click Refresh.
   b. Select Logic Tab: Click Refresh.
   c. Object Tab: Object Browser -> SCADAPack I/O -> Refresh. Notice objects already exist which are linked to the Physical IO of the SCADAPack x70.

Note: Object page may be unable to read IO from factory reset SCADAPack x70 controller because a factory reset controller does not have any IO configuration.

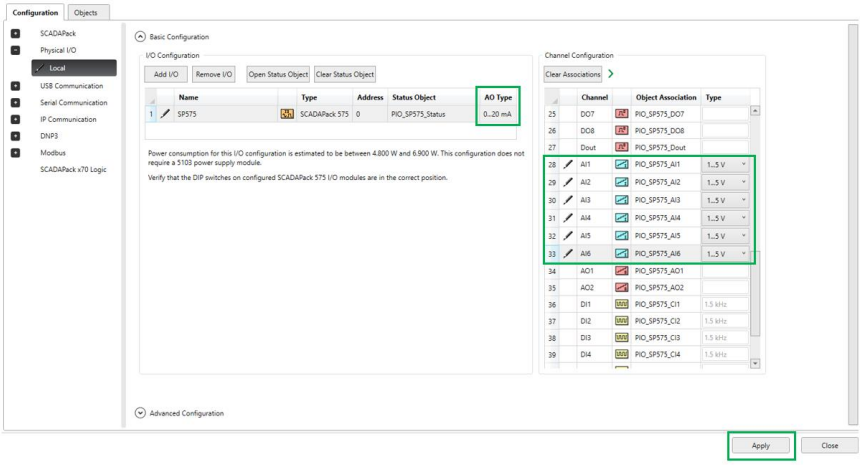
IMPORTANT!!

To protect the board from overheating after factory settings have been reset the analogue I/O's need to be configured.

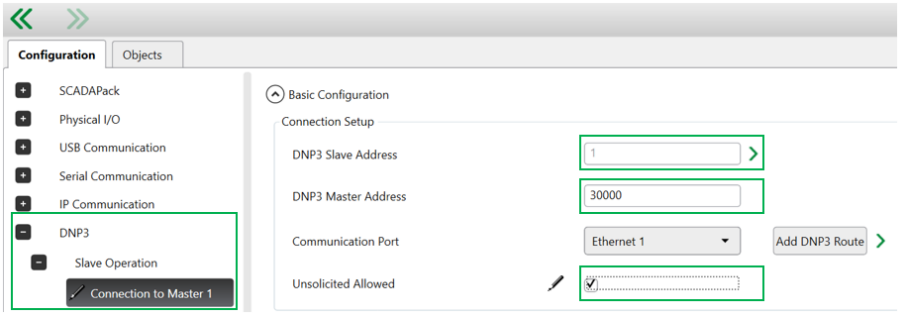
a. In Remote Connect -> Configuration Tab -> Physical I/O -> Local
b. Double click SP575 icon, this will bring up a dialog box -> Set AO Output Type to 0…5V, click ok.
c. Next in the Configuration Tab -> Channel Configuration -> Scroll down to AI -> In the Type column use the drop-down arrow to change the type associated with each AI to 0…5V. Click Apply.

Write the file to the device. This will reset the AO's (LED Bar1 and Bar2) to off until programs use them and AI's input range, this will prevent board burnout.

Note: The 'Write' to device button is located in the top left corner of Remote Connect.



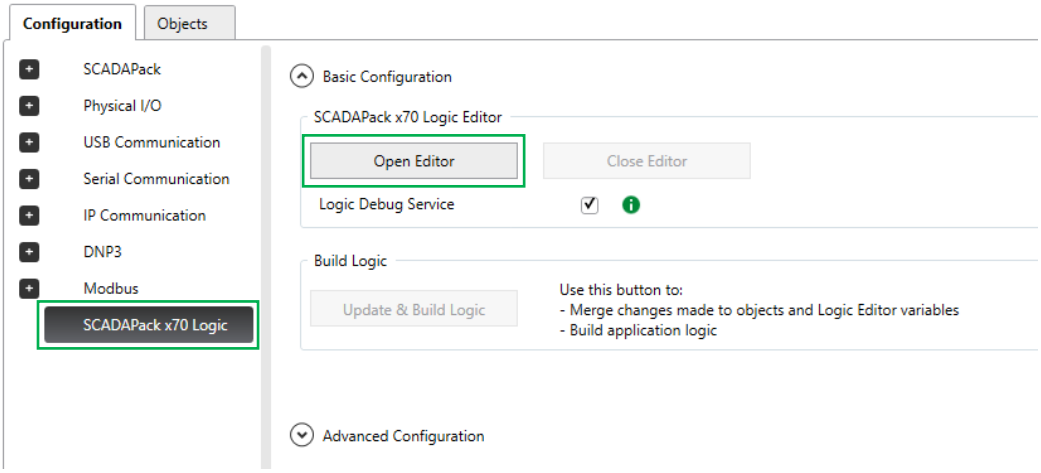
3- To finish off USB configuration settings -> Expand DNP3 -> Slave -> Connection to Master 1. Confirm the following information.
   a. DNP3 Slave Address: This should match the RTU's address that was set in new project setup Click apply.
   b. The Master address is set to 30000.
   c. Tick the box -> Unsolicited Allowed



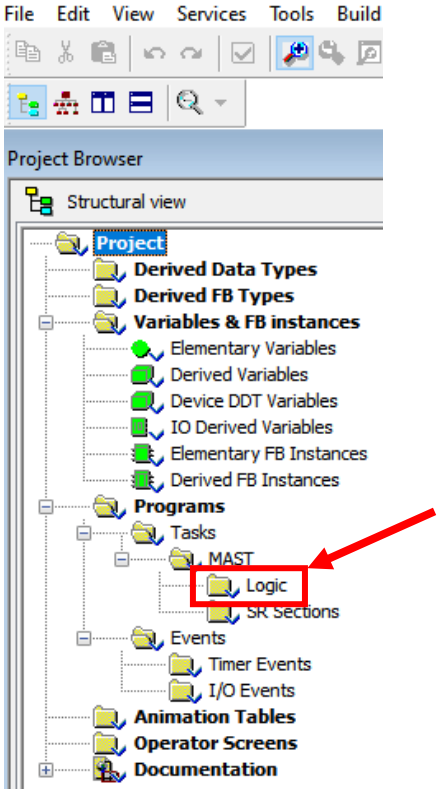
4- Write configuration to the RTU (the devices need to be online to do this)
5- Select SCADAPack x70 controller settings and bring up the Online Diagnostics Tab.
6- Refresh the status, logic and object Tabs as done previously
7- Right Click '**PC Communication settings – DNP3 CommDTM'** and select go offline. Notice text will no longer be bold.

2.3 Logic Editor

After you finish the SCADApack factory restart and the Remote Connect configuration, go the Configuration tab inside Remote Connect, access the SCADAPack x70 Logic, then click on the link “Open Editor”.

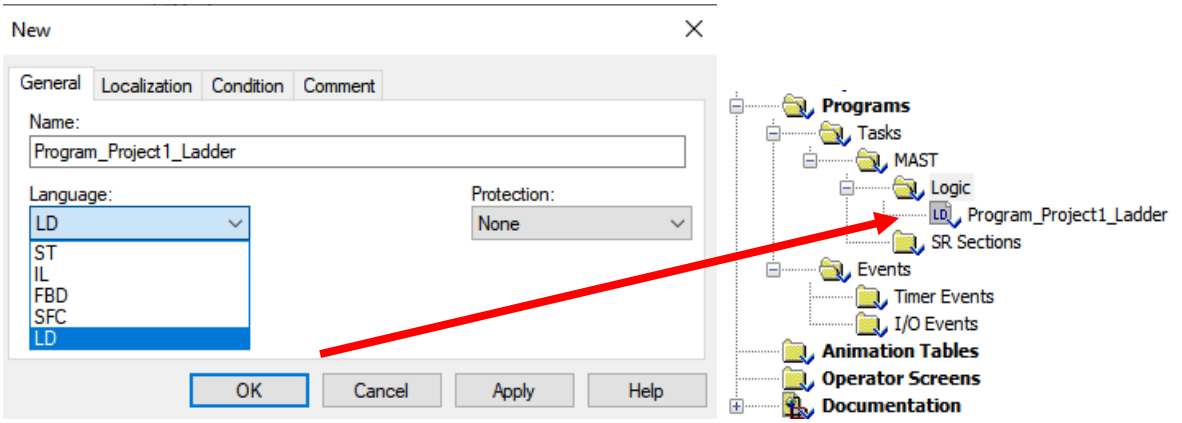


In the Logic Editor, the SCADA program code is written, analysed, build and debugged. First of all, a program code must be written. In the Logic Editor, go to Program-> Taks -> MAST -> Logic, then right-click in Sections and select New Section.



When the New Section window will pop up, follow the steps:

1- Define a name for your program code file, e.g., Program_Project1_Ladder.
2- To define the language of your program as LADDER, select LD in the language drop box.
3- Press OK to insert the program code file.



The ladder code window is divided between columns and rows. Note that the number of columns is limited to 11, however the initial number of rows is 100 and rows can be added if necessary (right click on the row number to insert or delete a row). To insert a variable, utilize the toolbar displayed in the top ribbon. The figure below illustrates this tollbar. Note that the contacts or inputs (normally open, normally closed, positive transition-sensing and negative transition-sensing) are inside the green box, the coils (open, negated, set and reset) in the blue box, the connections (horizontal and vertical) in the red box and the comment box in the yellow box. Single click on the object to select it and then click to place in the desired place in the logic editor window.

OPTIONAL: The table below contains the short cuts for all variables described previously. Remember to press ESC to use the mouse arrow again, after inserting the variables desired.

Variable Class	Variable Name	Short cut keys
Contacts (Input)	Normally Open	F3
	Normally Closed	SHIFT + F3
	Positive Transition-sensing	ALT + F3
	Negative Transition-sensing	SHIFT + ALT + F3
Coils (Output)	Coil	F5
	Negated Coil	SHIFT + F5
	Set	ALT + F5
	Reset	SHIFT + ALT + F5
Connections	Horizontal	F7
	Vertical	SHIFT + F7
Comment	Comment Box	F8

3. Project Description – Ladder Diagram

Drink Filling Process

A soft drink factory consists of many stages, one of these is the drink filling process, in which an empty can arrives to be filled with a soft drink. After the filling process, the can is closed and labelled. At the end of the conveyor belt, a visual sensor detects any faults of the final product. Figure 3 illustrates this process in detail, observe the position of sensors and machines. The factory applies a SCADA system to control all components illustrated in Figure 3. Consider that this progress has three stages:

- Stage 1: Can arrives at the conveyor belt 1 where it is filled with the soft drink.
- Stage 2: The full can goes to the second robotic arm to be labelled.
- Stage 3: A vision sensor analyses if the can has any fault.

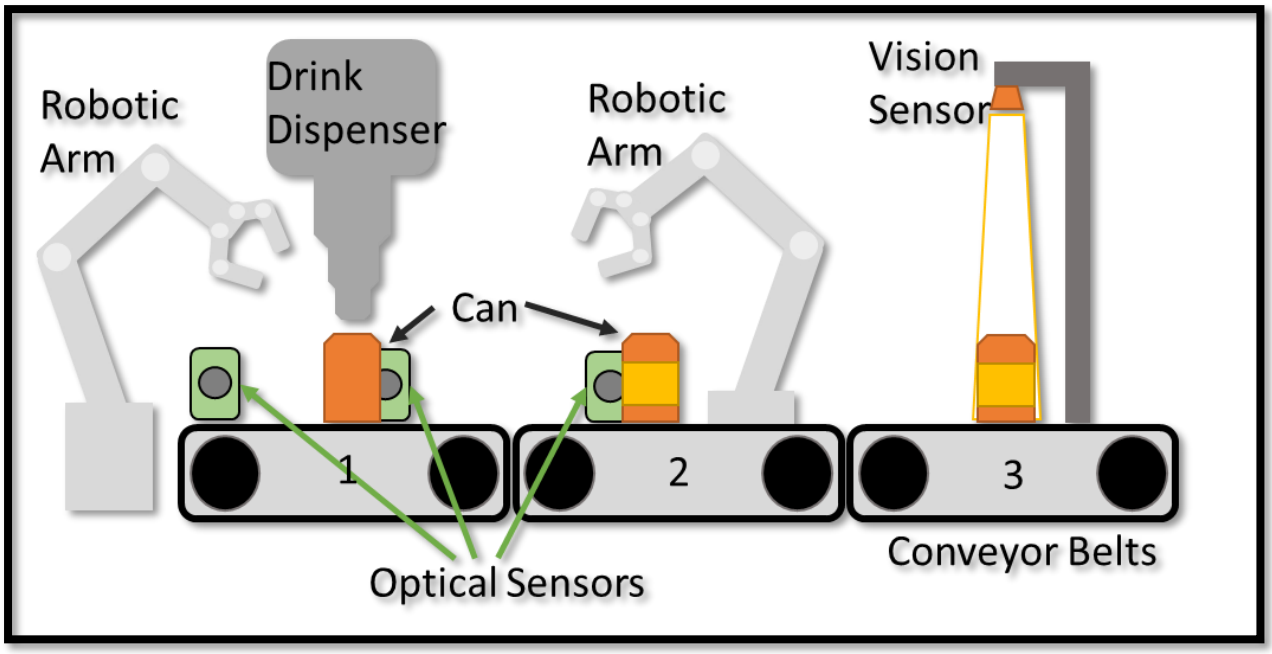


Figure 3. Drink filling process schematic.

Project Steps

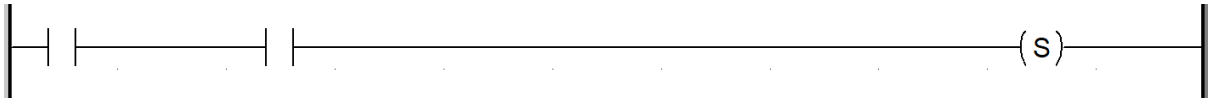
In this project, you are required to implement a SCADA system to control the drink filling process. The first task is to be completed in Ladder Diagram (LD) and is divided into three stages. However, all stages depend on a single input signal to operate: A manual ON/OFF switch.

Stage One: Drink Filling

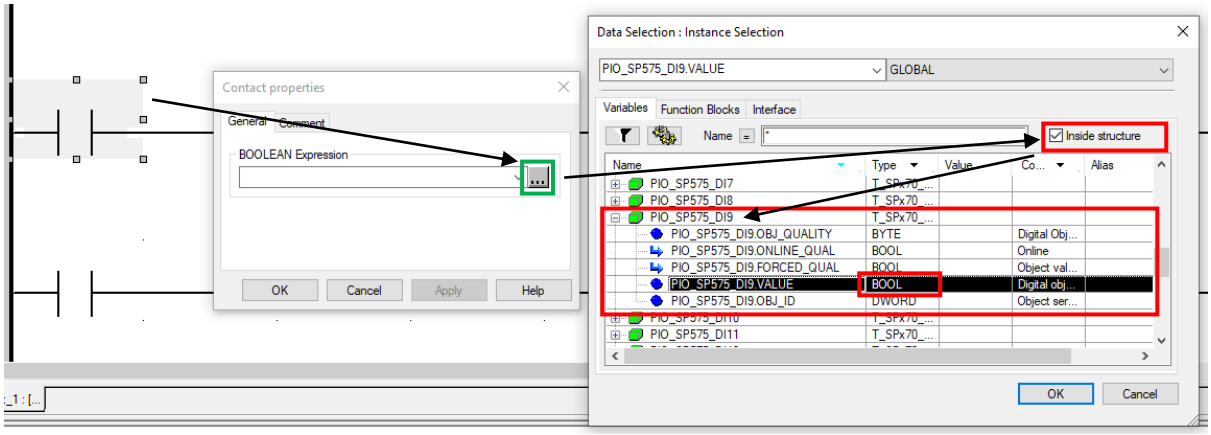
In stage one, you are required to implement a code that describes the *conveyor belt 1* in Figure 3. To begin any operation of belt, robotic arm and drink dispenser, the ON/OFF switch MUST be active (ON mode). Therefore, associate the first switch (Digital Input 9 – DI9) of your SCADA board to represent the ON/OFF switch. This switch is connected to the logic as an AND input because it MUST be active for the output (conveyor belt) to be active.

In the first stage, the can is placed at the conveyor belt 1 by the robotic arm. At this position, there is an optical sensor to detect the can. Once, the sensor is triggered AND the ON/OFF switch is activated, the conveyor belt 1 motor starts to operate. Associate the Digital Input 1 (DI1 – First Button) as the first optical sensor and the conveyor belt motor as Digital Output 1 (DO1 – First LED).

In the first line of your LADDER logic, place a normally open contact in series with another normally open contact and a set coil, as shown in the figure below:



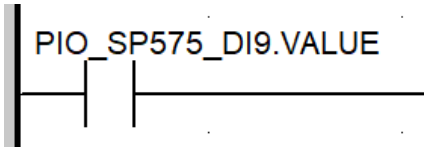
After placing the variables, double click on first contact to open Contact Properties window. In the General tab, at the right end of BOOLEAN Expression text field, click on the "…" (triple dot) to open the Data Selection window. If the Variables tab is empty, check the box "Inside Structure" at the right to see the list of objects from the SCADAPack. In this list, search for PIO_SP575_DI9 object and open the list of variables inside the object, then select the PIO_SP575_DI9.VALUE, as shown in the figure below.



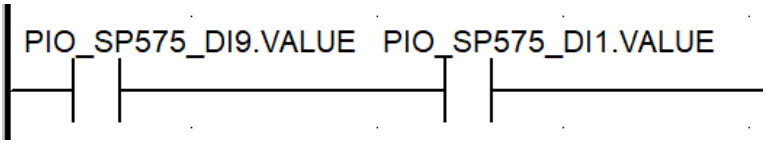
After selecting the object, the variable is associated with this object. Observe that the name of the object appears on the variable top.

Note: if there is an error of association between variable and object, a waved red line is going to appear bellow the object name. Those errors of association may occur because:

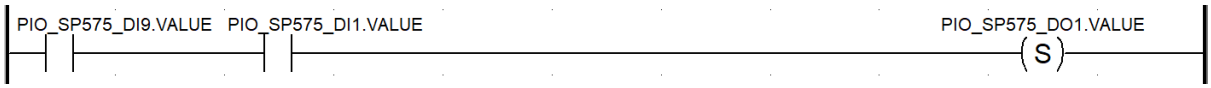
- Type of variable and object are different. Example: Digital variable is always a BOOLEAN.
- The object name is incorrect (when you enter the name manually) or that object does not exist.



Repeat the same process to the second contact, associating with the Digital Input 1 (DI1), the first button.



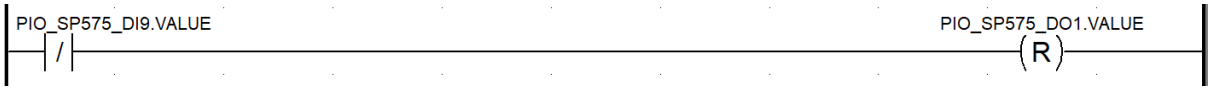
Repeat the process one last time, associating the set coil with the Digital Output 1 (DO1), first LED.



The first line of logic is finished. This line represents that, when the ON/OFF switch is ON (activated) and the first sensor detects, a can is located at the beginning of production line and thus, the conveyor belt 1 motor is set to operate.

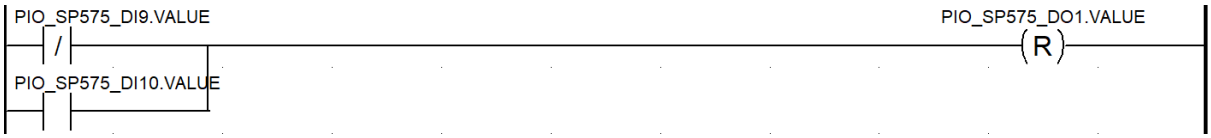
The next line in your logic implements the reset logic for the conveyor belt motor. Leave one or two empty rows between the previous logic and the next one, as the program code will be incremented step by step. Now add another row, with one normally closed contact and one reset coil. Associate the contact to Digital Input 9 (DI9 - PIO_SP575_DI9.VALUE) and the reset coil with Digital Output 1 (DO1 - PIO_SP575_DO1.VALUE). Consequently, the motor stops when the ON/OFF switch is deactivated.

Whenever the ON/OFF switch is deactivated, the conveyor belt 1 motor stops immediately.

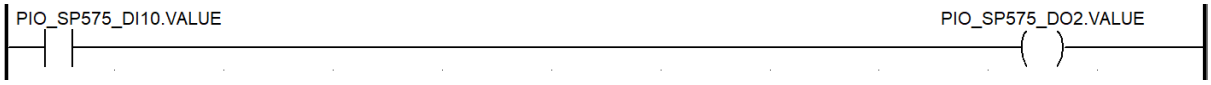


Once the conveyor belt 1 motor is operating, the conveyor belt 1 moves the can until it reaches the second optical sensor. When the second sensor is triggered, the conveyor belt 1 stops and the drink dispenser is activated.

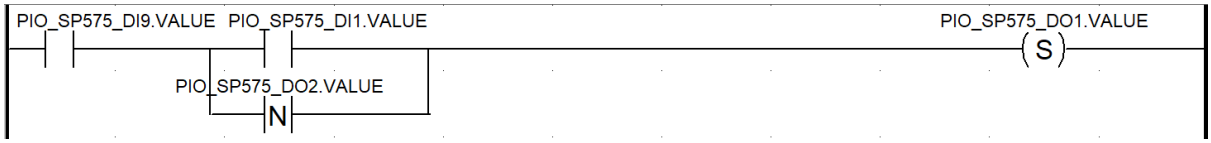
Therefore, the ON/OFF switch OR the optical sensor 2 must be able to stop the conveyor belt 1. Add one normally open contact in parallel to the Digital Input 9 (DI9) contact. Associate the Digital Input 10 (DI10 - PIO_SP575_DI10.VALUE) to the normally open contact. This connection represents an OR gate.



Besides stopping the conveyor belt 1, the optical sensor 2 also enables the drink dispenser. Therefore, another row of code is required to connect optical sensor 2 (DI10) to the drink dispenser. Add in this row a normally open contact and a normal coil. Associate the normally open contact with DI2 and the normal coil with Digital Output 2 (DO2 - PIO_SP575_DO2.VALUE).

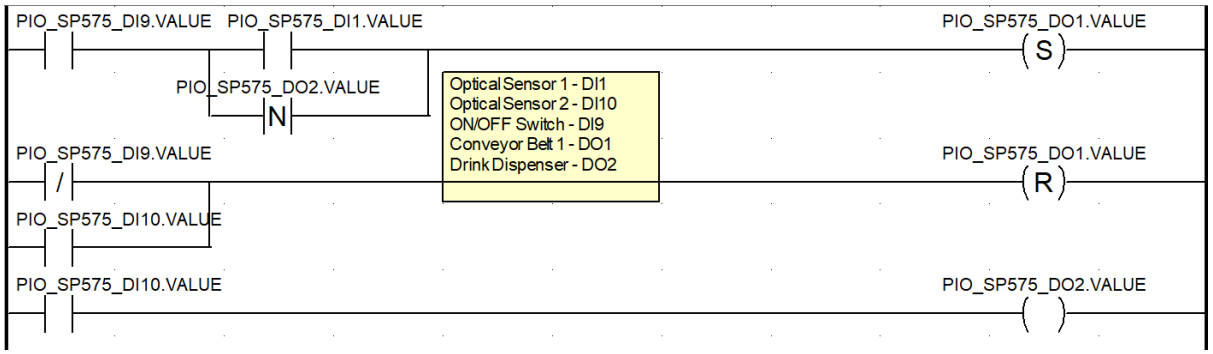


Immediately after filling the can with the beverage, the dispenser is deactivated, and the conveyor belt 1 motor reassumes operation. Hence, the trigger of this logic is the falling edge of the dispenser (DO2 – Digital Output 2). For the motor of conveyor belt 1 to be activated again, place one Negative Transition-sensing contact in parallel with the optical sensor 1 (DI1). Associate this contact with PIO_SP575_DO2.VALUE (Digital Output 2 _ DO2). Whenever the DO2 transit from positive to negative state, the contact is triggered and the DO1 is activated again.

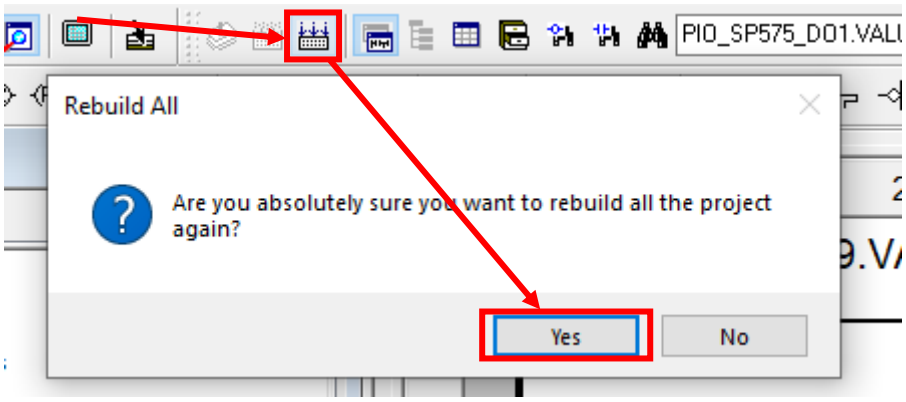


The stage 1 code is finished, however, a good programming code requires one last step: COMMENTS. Remote Connect allows the user to add Comment Boxes, on the tool bar and using the short key F8. Add one comment box and write it down the name of the digital I/Os that have been implemented and the object that it is represented, as shown below. Place the comment where it is clear to see and read.

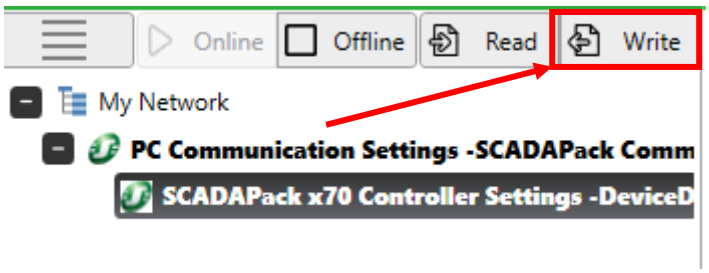
Note: Comment boxes must be implemented because any other user or programmer that has not work in this code, will not understand what each variable represents.

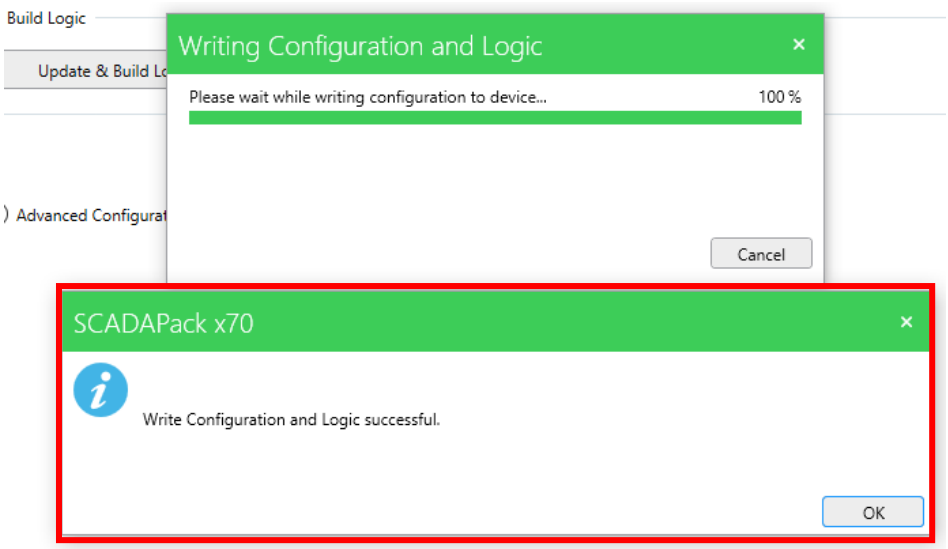


Now that the Stage 1 code is done and comments were added, go to the tool bar, and click on "*Rebuild all"* Project or in the tab *Build -> Rebuild all Project*.

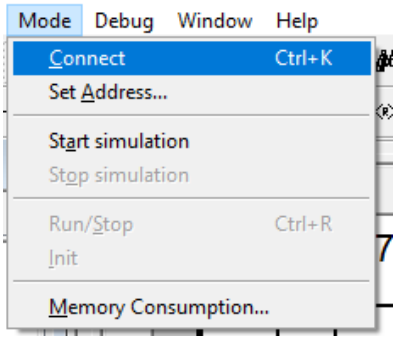


After successfully building the project inside the Logic Editor, go back to Remote Connect and at the left side above the SCADApack device and Communication, and click on *Write*. Remote Connect is going to transfer the code into the SCADApack. Wait until the window with the following message "Write Configuration and Logic successful" pop up. Example images below:

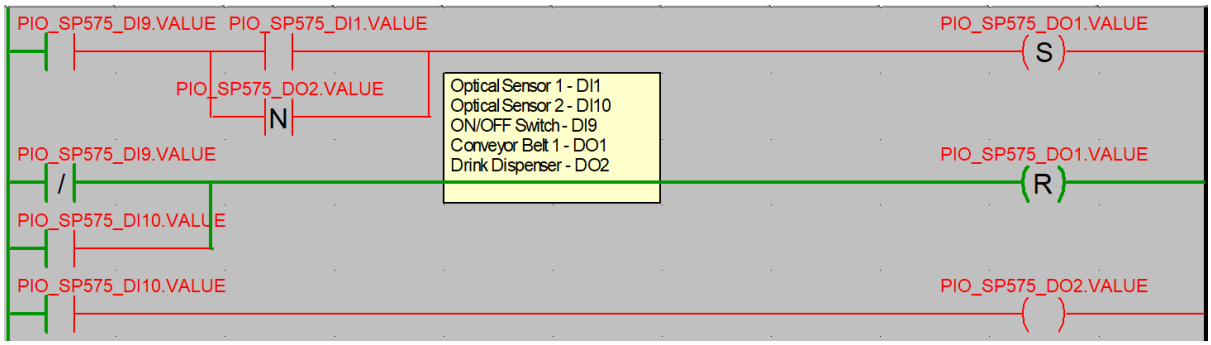




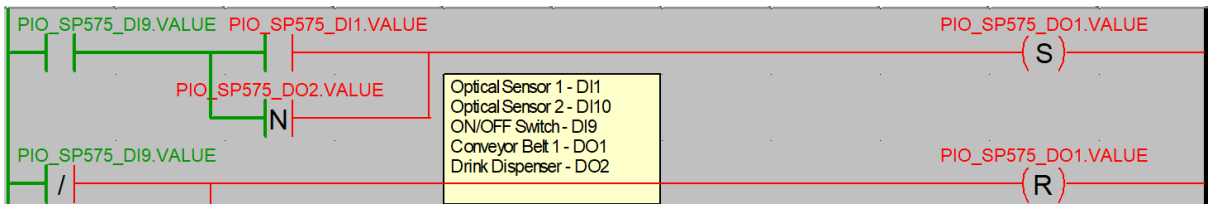
Click on OK to dismiss the window. To visualize and test the code at the same time, go back to the Logic Editor, go to the tab Mode -> Connect (short key: CTRL + K).



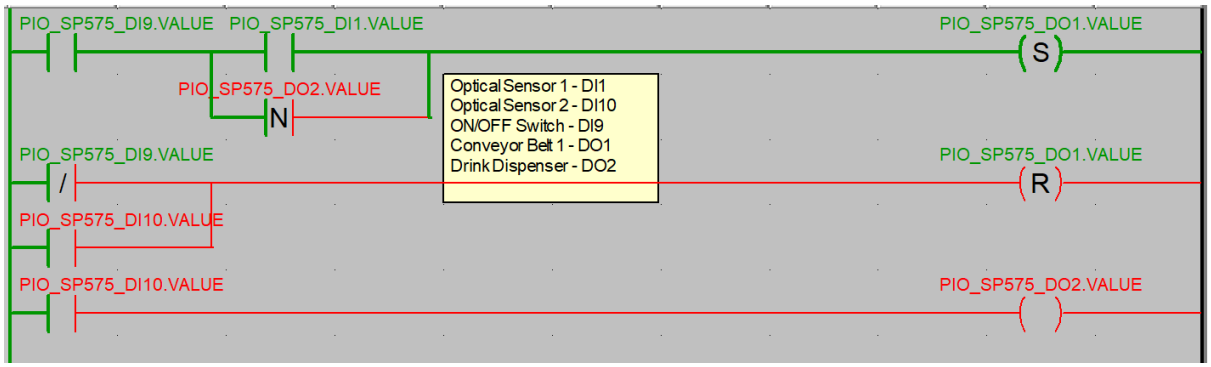
When the Logic Editor is connected to the SCADApack, the background turns to dark grey and the variables, object and connection lines, turn to red or green, which represent the logic states: FALSE (red) and TRUE (green).



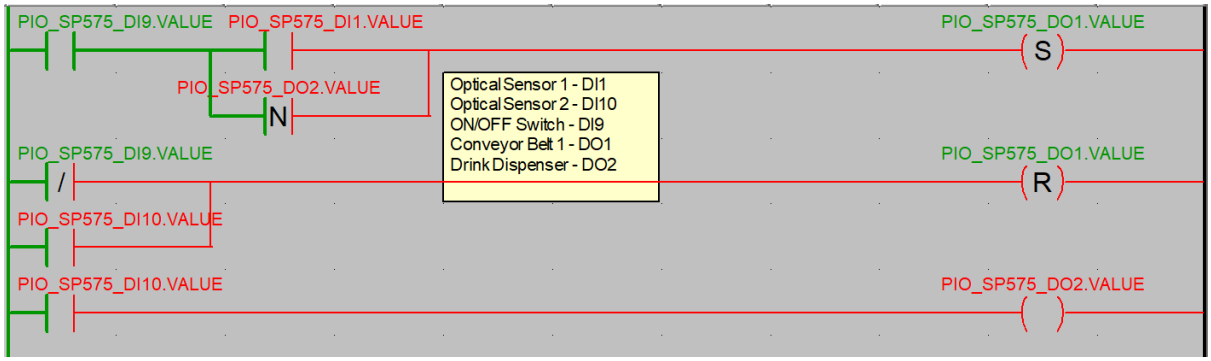
Activate the first switch, Digital Input 9 (DI9), observe that the normally open contact line turns green at the right side.



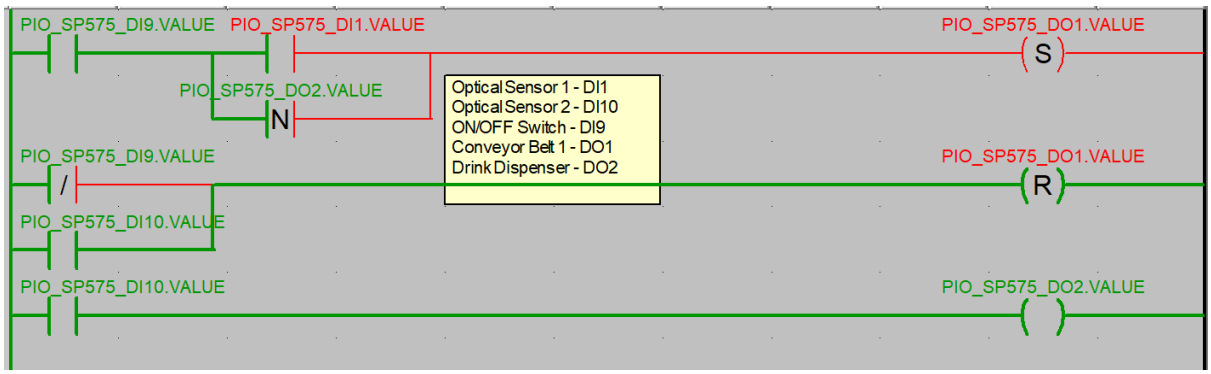
Next, press and hold the first button, Digital Input 1 (DI1). Observe the first row becomes green and that the first LED on the SCADA board (Digital Output 1 – DO1) turns ON. Also, the variable name inside the Logic Editor becomes green.



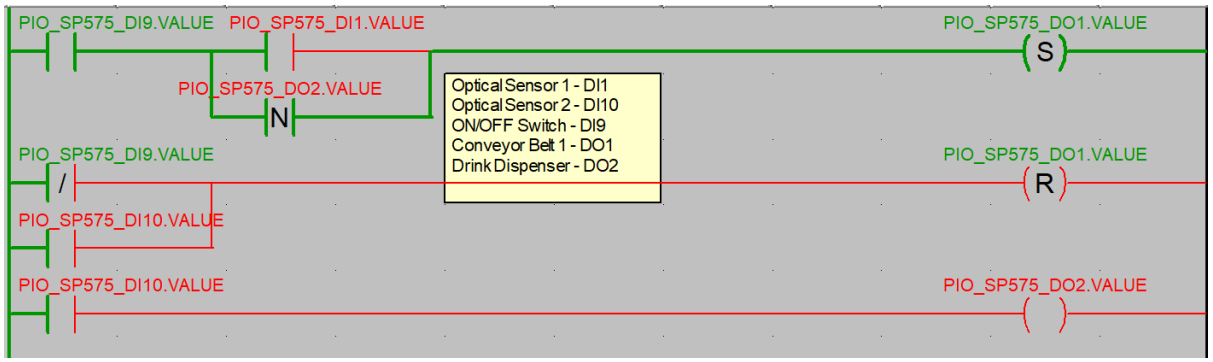
If the DI1 is released the contact line goes back to the original state FALSE, but DO2 variable name continues to be green, as it represents that the conveyor belt 1 motor is active and operational.

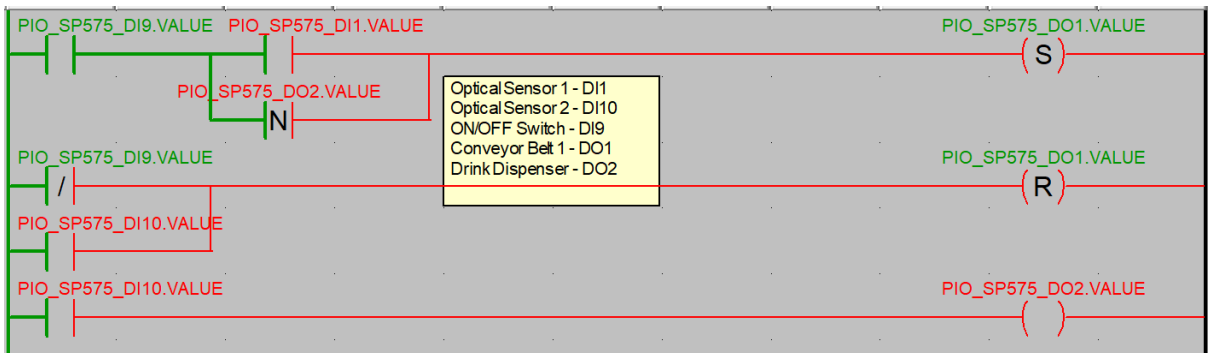


When activating the second switch (Digital Input 10 - DI10), the second and third row line becomes active, resetting DO1 and activating DO2.



Turning the second switch OFF, DO2 is deactivated and DO1 is activated again. The DI10 transition from ON to OFF can be seen in a fraction of seconds, therefore the visualization below was taken precisely and might not be seen. However, observe that the DO2 contact is activated, and triggers DO1, reassuming “normal” operation of the conveyor belt 1 motor.





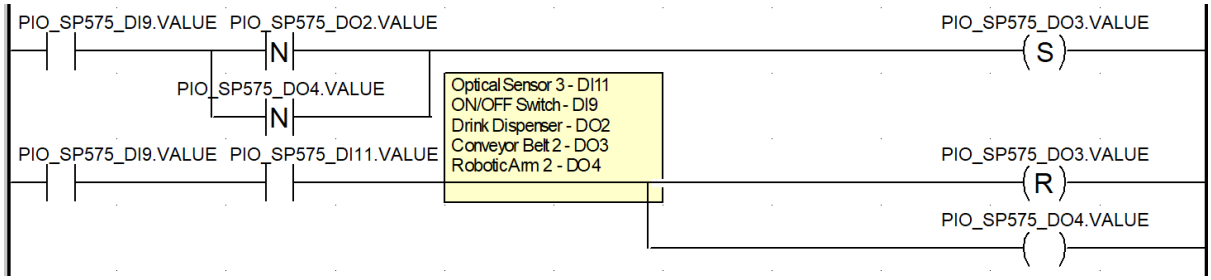
Stage Two: Labelling

In stage 2, **the conveyor belt 2 motor is automatically activated after the drink filling stage is over. The can is transported to a second station where the robotic arm 2 is placed. An optical sensor is located at this robotic arm. When the can arrives at the station, the conveyor belt 2 stops, then the robotic places label from soft drink company on the can. After executing the labelling task, conveyor belt 2 is reactivated.**

This scenario is similar to stage 1, whereas the conveyor belt motor is turn ON/OFF based on soft drink can location. Note that there are two Negative Transition-sensing contacts in the first row, because this stage starts when previous stage process is finished (negative transition from the drink dispenser). For representing the stage 2 objects implement the variables:

- Optical sensor -> Digital Input 11 – switch: PIO_SP575_DI11.VALUE
- Conveyor Belt 2 Motor -> Digital Output 3 – LED: PIO_SP575_DO3.VALUE
- Robotic Arm 2 -> Digital Output 4 – LED: PIO_SP575_DO4.VALUE

Remember to insert another comment box and write the names of variables/objects utilized in this stage.



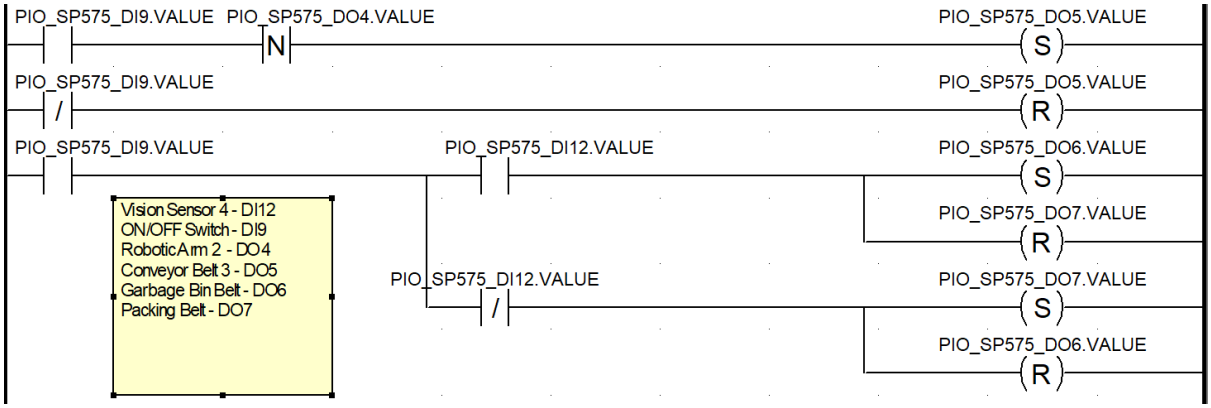
Stage Three: Fault Check

In stage 3, **the conveyor belt 3 motor is automatically activated after the robotic arm 2 stage finishes labelling the can. The can is transported to a third station where a vision sensor analyses the soft drink can. At this stage, the conveyor belt 3 motor is not required to stop, as the vision sensor has enough range to detect any faults from the can. The vision sensor checks for faults. If a fault is detected, the can goes to garbage bin. If the can has no faults, it goes into the packing station.**

This code requires a decision-making logic for the output of the vision sensor. **Suppose that there are two conveyor belts after the vision, one goes until the garbage bin direction and the other to the packing station.** Therefore, implement two digital outputs for the final conveyor belts and a single digital input for the vision sensor. For representing the stage 3 objects implement the variables:

- Vision sensor -> Digital Input 12 – switch: PIO_SP575_DI12.VALUE
- Conveyor Belt 3 Motor -> Digital Output 5 – LED: PIO_SP575_DO3.VALUE
- Garbage Bin Station -> Digital Output 6 – LED: PIO_SP575_DO6.VALUE
- Packing Station -> Digital Output 7 – LED: PIO_SP575_DO7.VALUE

The following code is an example but try to build your own code before looking at the answer!



4. Project Description – Function Block Diagram

The scenario made in LADDER Diagram is a simplification of reality, focused on Digital IOs, whereas the filling process has many variables, digital and analogue. For this part of Project 1, the scenario is more complex and has two types of objects: can and bottle. The objective is to implement an automated production line, adding also the analogue IOs. Observe that Figure 4 has both bottle and can (but only one item is the production line), in addition to previous scenario, there is another station to place the lid (stage 3), added between labelling and fault check stages. Consider that the process has four stages now:

- Stage 1: Can/bottle arrives at the conveyor belt 1 where it is filled with the soft drink.
- Stage 2: The full can/bottle goes to the second robotic arm to be labelled.
- Stage 3: The full can/bottle goes to the third robotic arm to be closed (placing the lid).
- Stage 4: A vision sensor analyse if the can/bottle has any fault.

Stage 1 is made step by step to provide guidance with Function Block Diagram language, while the other stages are exercises, although there an example, try to implement the code on your own. All code shown in this project is a suggestion, as the problem can be solved in different logics, however, be careful to fulfill every requirement described.

All stages are activated ONLY when the ON/OFF switch is ON, otherwise the system is shutdown.

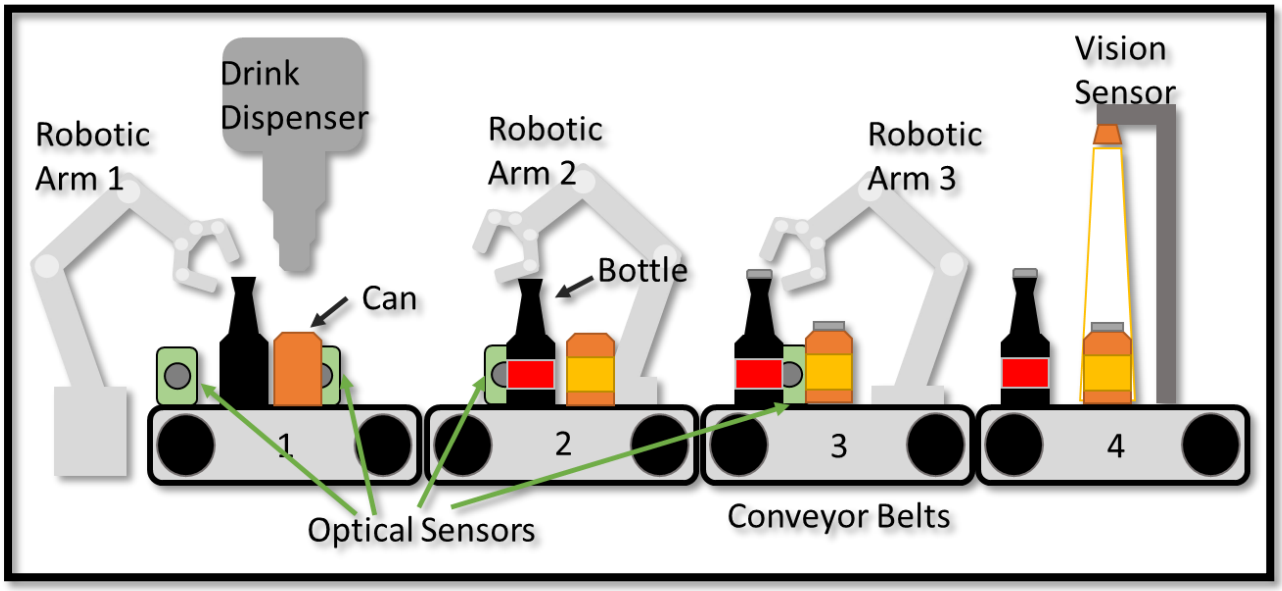


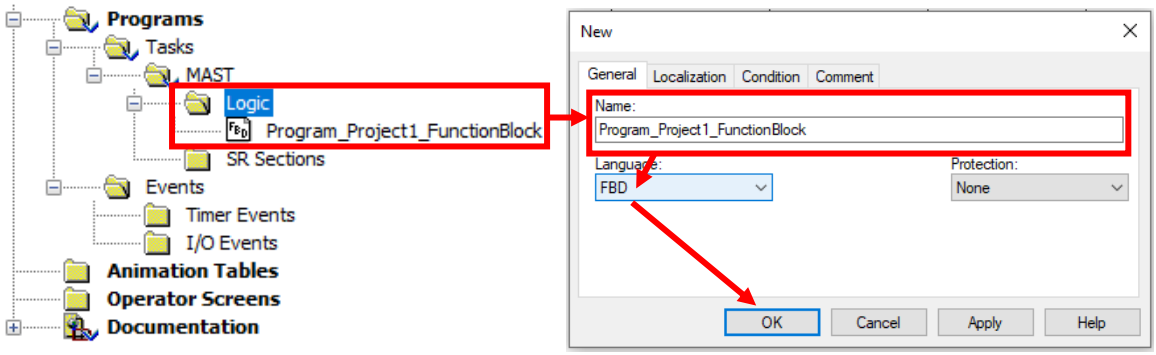
Figure 4. Production line of soft drink with bottle and can.

Stage One: Drink Filling Process

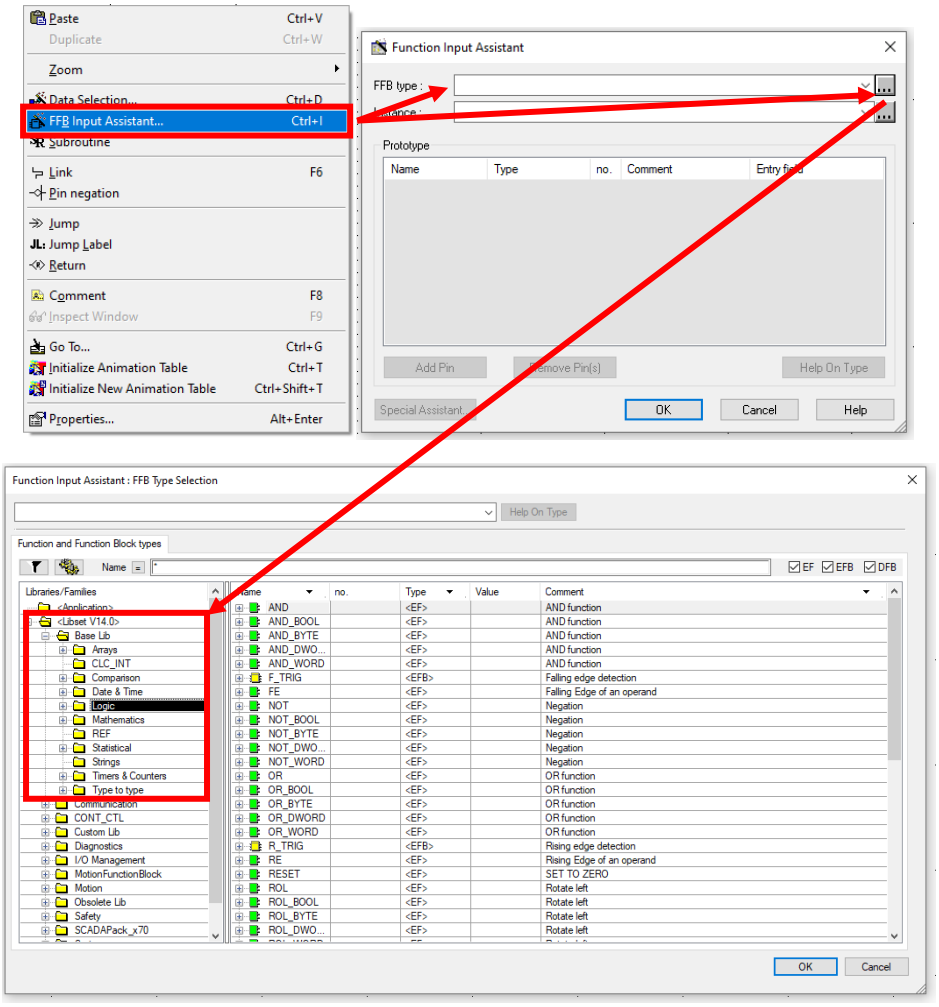
Considering that the system is operational (power switch is ON), in first stage, the robotic arm 1 places a can/bottle at the beginning of conveyor belt 1. The optical sensor detects the object presence and activates the conveyor belt 1 motor. The can/bottle moves until reaching the second optical sensor at the drink dispenser station. This optical sensor stops the conveyor belt 1 motor and the filling process starts.

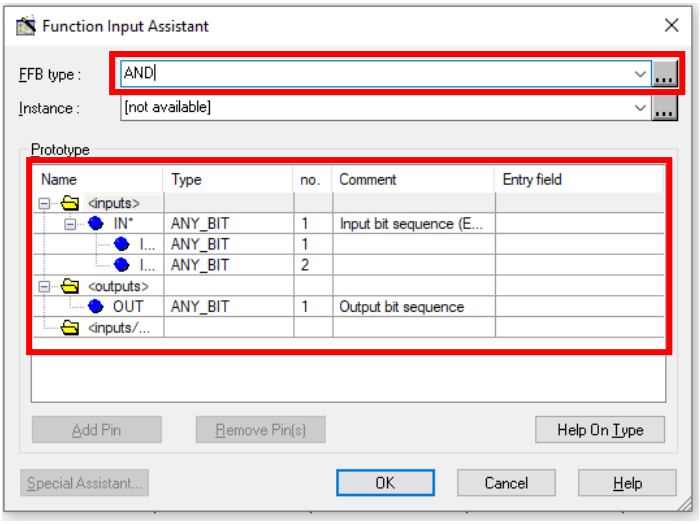
Similar to the LADDER Diagram code developed previously, this part of stage 1 is fully digital and has the same logic. Save the LADDER project, close it (Logic Editor and Remote Connect) and create a new

project. Open the Logic Editor, create a New Section inside the Logic folder, name your program code (example: *Program_Project1_FunctionBlock*) and define the language as *Function Block Diagram* - FBD.



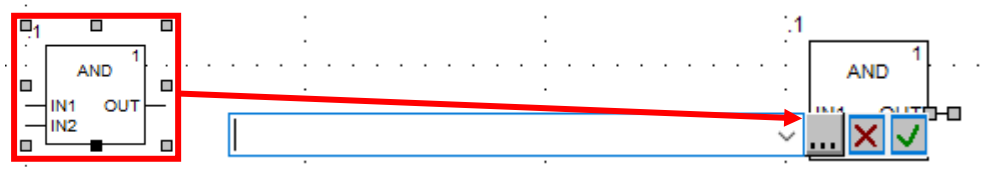
To insert a Function Block, go to the tab Edit -> FFB Input Assistant, or Right-click inside the code area and select FFB Input Assistant. The shortcut keys are CTRL + I. You see the Function Input Assistant now. Go to the FFB type text field and click on "…" (three dots) to open the FFB Type Selection window. All functions required in this project are inside the folder Libset V14.0 -> Base Lib. The digital IOs, such as AND, OR, XOR, Set/Reset, Falling Edge and Rising Edge, are inside the folder Logic as shown below. Search for "AND" gate and click on OK. Observe that the Function Input Assistant has the name of the function block selected and the IOs related to this function. After becoming familiar with the function blocks name, write it on the text field FFB type to automatically select the desired function. Place the AND in the logic editor.





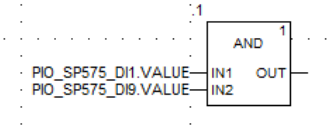
Another option is to use the tool bar for inserting the variables into the code area. Observe in the figure below, the Function Input Assistant is inside the red box. The green box has the LINK and TOGGLE PIN-NEGATION, the LINK connects function blocks IOs and the TOGGLE PIN-NEGATION switch the IOs pins from normally open to normally close, or vice-versa. In the blue box, there are the JUMP and JUMP LABEL variables, this part is described and demonstrated in Project 2. For the comment box click in the symbol inside the purple box.

Insert the AND gate function block into the code area, observe that there are two inputs pins at the left side, IN1 and IN2, and one output pin at the right side, OUT. To associate a function pin with an object from the SCADApack, double-click on the pin to see the text field bar, go to the "…" symbol to access the object selection, like has been done with LADDER Diagram part.



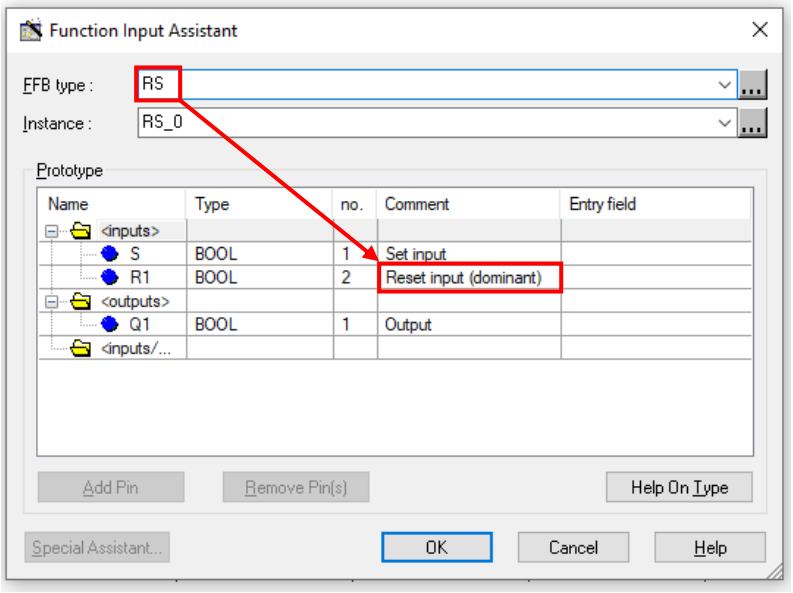
Select PIO_SP575_DI1.VALUE (first optical sensor) and PIO_SP575_DI9.VALUE (ON/OFF switch) for pins IN1 and IN2, as figure below. The order of objects and pins does not matter for function blocks like AND, OR and XOR.

Note: These function blocks have up to 32 input pins. To increase the number of input pins, click on the function block, observe the black square at the lower bottom (picture above), click and drag down this black square to create more input pins.



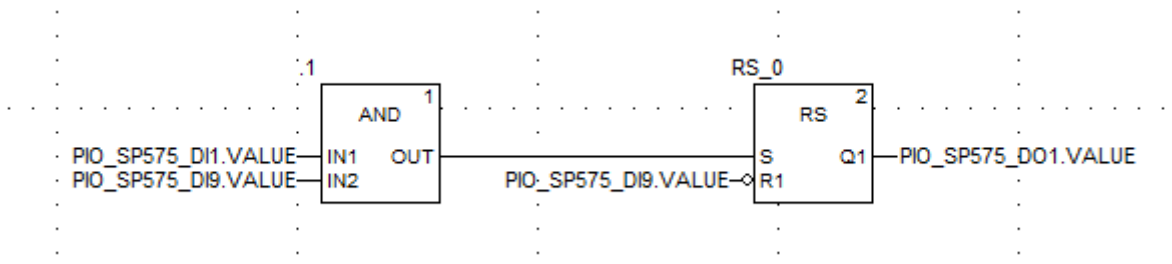
Access the Function Input Assistant again, search for the Set/Reset function called RS. Be careful to select the function block described in the picture below, as there are two types of SET/RESET in FBD. On the RS function block, the RESET is dominant, which means that if both SET and RESET are TRUE, then the output is FALSE, as RESET has priority. The other function block is called SR, where SET is dominant and therefore, if both inputs are TRUE, the output is TRUE as well.

Note: observe that the Function Input Assistant has the dominant comment for the input that is dominant. Figure below.



Place the RS function block in the code area at the right side of the AND block. LINK the AND *OUT pin* to the RS *S pin*, as illustrated below. Access the LINK in the tool bar as shown previously or use the short cut key F6. Then, select the TOGGLE PIN-Negation to change the R1 pin from normally open to normally close. Observe the small circle at the pin, symbolizing the negation pin.

Associate PIO_SP575_DI9.VALUE to *R1 pin* and PIO_SP575_DO1.VALUE to *Q1 pin*. This code is equivalent to the first code developed in LADDER. Build and write this program in the SCADApack. Go to ONLINE mode to observe the same pattern, GREEN colour for digital components that are TRUE and RED colour for FALSE.

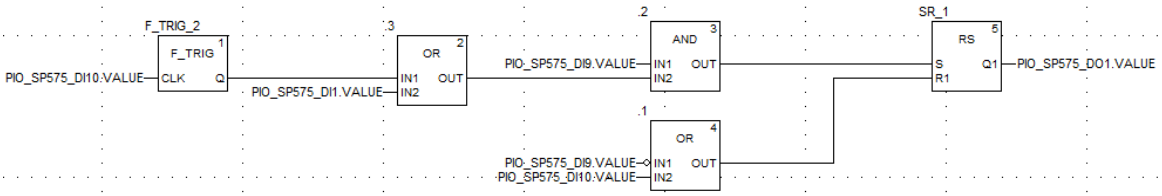


Next step is to modify this code for the drink dispenser to stop when the can/bottle reaches the second optical sensor and for resuming operation when this sensor is off (filling process is finished). Remember that:

- DO1 is turned down when second optical sensor - DI10 (second switch) is ON.

- DO1 returns to TRUE state when DI10 is turned off. Therefore, there is an OR gate for DI1 and DI10 (negative transition). To detect the falling edge of DI10 search for the function block: F_TRIG.

An example of code that implement these changes is illustrated below:



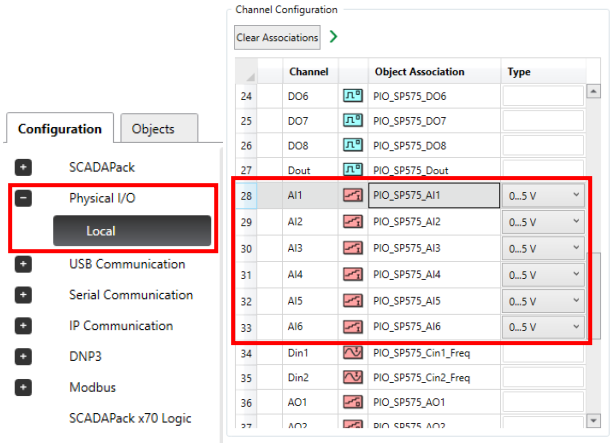
The analogue input 1 (AI1) controls the drink dispenser. When the second optical sensor detects the can/bottle and the conveyor belt 1 motor is stopped, the drink dispenser (analogue input) must be utilized to fill the object. The amount of soft drink differs from the can to the bottle. Consider that the can is fulfil at 50% of AI1, while the bottle is at 80%. Additionally, the can must stay still for 2 seconds for the drink to settle, while the bottle requires 3 seconds. After this time, a LED lights up indicating that the process has finished, and the object is ready for the next station.

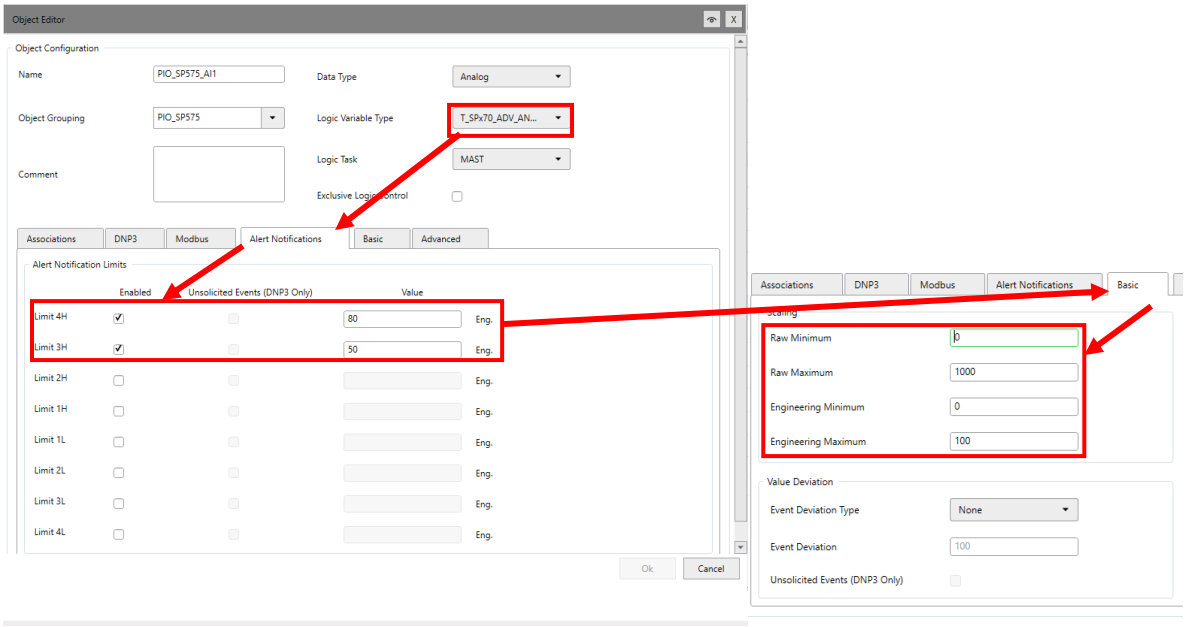
Note 1: This output (LED) is the same for both can and bottle.

Note 2: Implement the last switch (digital input 16 – DI16) to represent the object in the production line, consider that when the switch is OFF, it represents a can, otherwise, when ON, it represents a bottle.

Before implementing the code for the drink dispenser applying the analogue IOs, go back to Remote Connect -> Configuration – SP x70 Controller (tab) -> Configuration -> Physical I/O -> Local. Look for the right column “Channel Configuration”, scroll down the tab to find the analogue input channels AI1 to AI6. Modify the Column “Type” to “0…5V”, then double-click on AI1. The Object Editor window pops up, go to the Logic Variable Type and change it to T_SPx70_ADV_ANALOG, then go to the tab Alert Notifications, enable Limit 4H and Limit 3H, insert the respectively values, 80 and 50. Later, go to the tab BASIC, change the Scaling values, Raw Maximum to 1000 and Engineering Maximum to 100.

Once all configurations are done, click on OK and on the right bottom side at Remote Connect, click on APPLY. The changes are applied to the SCADApack configuration and to the Logic Editor. When adding the Analog Input 1, there are more options of variable since the variable type is Advance Analog. The raw and engineering values are scales that represent the Analog Input. Analog IOs are measured in terms of current and voltage. When applying the scale, the user can see it in the configurated range. For this project, the engineering value is utilized because it represents the IOs from 0% to 100%.





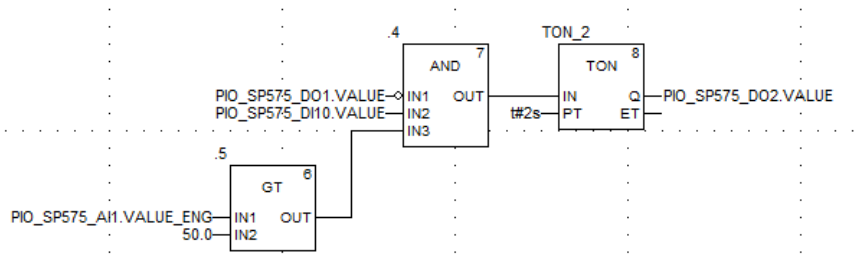
Now add an AND gate that has three input pins, associate two pins with PIO_SP575_DO1.VALUE (negated) and PIO_SP575_DI10.VALUE (second optical sensor, second switch). Add the function block GT (Greater than), linking the output with the third input from AND block. The GT function compares IN1 pin with IN2 pin, allowing the equation IN1>IN2. The output is a BOOLEAN value, TRUE or FALSE. Therefore, the pins order makes difference in this logic. Associated the PIO_SP575_AI1.VALUE_ENG to IN1 and enter the value 50 in the pin text field. The GT block compares the Analogue Engineering Value (from 0 to 100) to 50, symbolizing the 50% of soft drink that is poured inside the can.

For the 2 seconds that the can needs to stay still, the function block TON (Timer ON Delay) delays the signal accordingly to the pre-set time (PT input pin). To define the timer, use the notation:

t#2s

Where t# means time, the number 2 represents the time quantity and s represents the time unit, seconds. As a result, the pre-set timer is 2 seconds. Link the AND output pin to the input PIN from TON. Associate the TON output Q with PIO_SP575_DO2.VALUE.

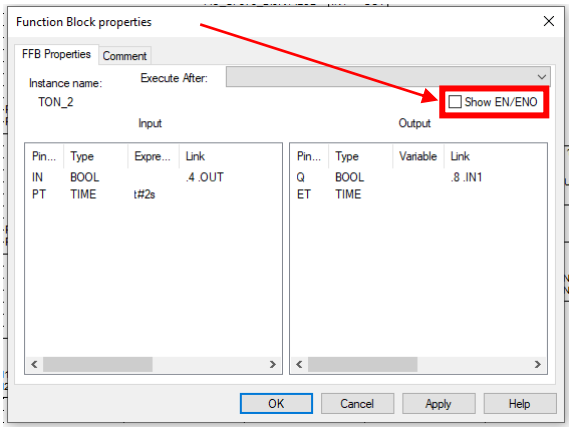
The described code can be seen here:



Build the changes, write it to the SCADApack and tested the code while observing the changes inside the Logic Editor (Mode -> Connect).

Consider that the last switch (DI16 – Digital Input 16) represent the object on the conveyor belt. Whenever DI16 state is FALSE, there is a can. On the other hand, when it is TRUE, a bottle has been detected. This way, the program code can differ the scenario and execute different routines. Copy and paste the previous code, changing the value in GT block to 80 and the pre-set timer to 3 seconds.

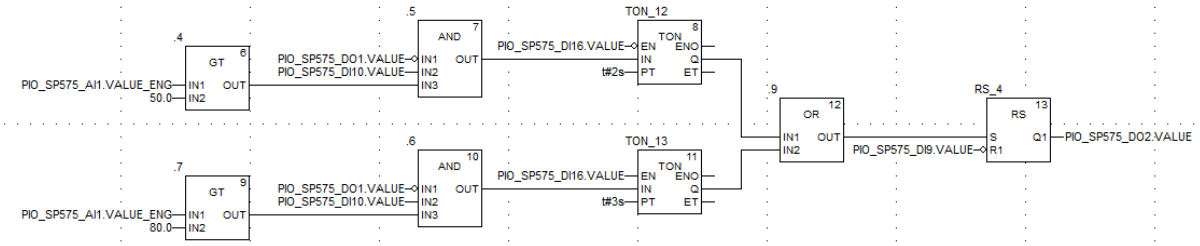
Double-click at the first TON block, the Function Block properties window pops up. Look at the right side for the checkbox “Show EN/ENO”, check this box and close the window. Do the same for the other TON block. This checkbox makes visible the input enable (EN) and enable output (ENO) pins. These pins have the function to activated or deactivate the function. Therefore, associate the first and second TON EN pin with PIO_SP575_DI16.VALUE, but apply the TOGGLE PIN-Negation to the first one. Consequently, when PIO_SP575_DI16.VALUE is FALSE, the first part is executed and DO2 is activated after 2 seconds and if AI1 has reached at least 50%.



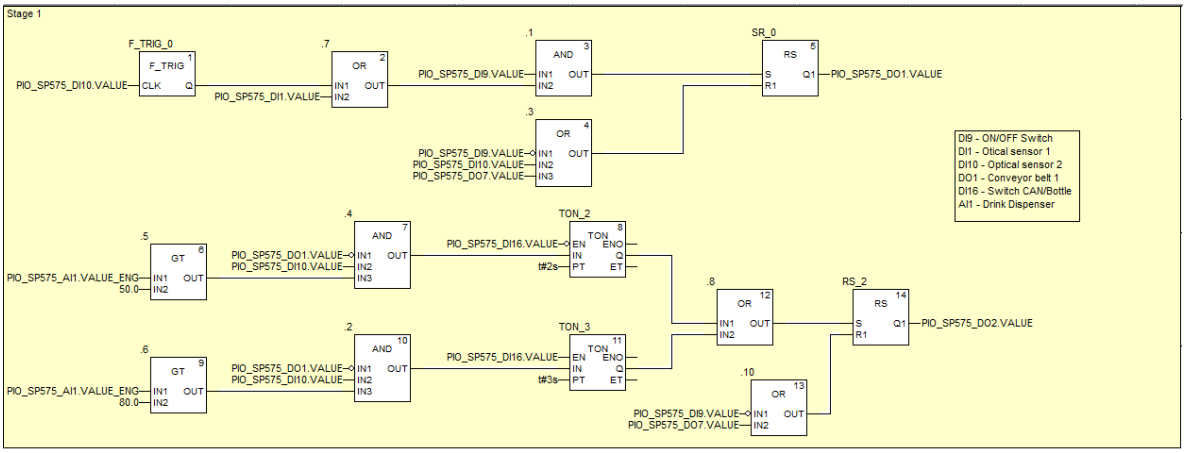
In the code developed in LADDER DIAGRAM (LD), DO2 represents the drink dispenser, however in Function Block Diagram **DO2 represents that the can/bottle were fully filled and the drink had settle, then it ready for the next stage. This output MUST be activated until the object leaves the production line.** From this part forwards, the digital outputs do not represent the conveyor belt motors, rather that the station process has been finished.

Furthermore, TON block deactivates the output when the input becomes FALSE. Because the second optical sensor (DI10) is deactivated after filling process is over, DO2 also deactivates. To prevent DO2 from deactivating a SET/RESET function block is required after the timer function blocks to maintain DO2 state ON. Additionally, FBD does not allow to connect two or more input into an output, thus an OR gate should receive Q pins from both TON function blocks, then connect the OR output pin to the set pin. The reset pin is connected to the negated DI9, because if the power is turned OFF, all processes must stop.

The figure below represents the code developed:



Now, stage one is complete but is missing and comments must be added. Same as LADDER, the comment box is found in the tool bar or using the shortcut key F8. Comment boxes are placed in the background, a good coding would be to extend the comment box to contain a specific area and add a short comment about that code. Comment boxes can be stacked, which makes it easier at writing comments and defining code area, as the example given below.



The following stages describe the scenario and the logic that the code MUST follow. Try to implement your own code before looking for the answer, as it is an exemplification, without any comments or explanations.

Stage Two: Label Process

Consider that the filling process has finished (including the seconds for drink to settle) and the can/bottle starts to move. It takes 1 second for the object (can or bottle) to reach the label station, after the second optical sensor detects that the object has left the drink dispenser station. At the label station, the robotic arm spends 2 seconds to label both objects. When the process is finished, another output (LED) is activated and remains ON, indicating that this station process has finished and the object is now transported to the next station.

Stage Three: Lid Process

To reach the lid station, the object (can or bottle) takes 1 second. At the lid station, the robotic arm spends 3 seconds to place the lid on the CAN and 2 seconds for the bottle. When the process is finished, another output (LED) is activated and remains ON, indicating that this station process has finished and the object is now transported to the next station.

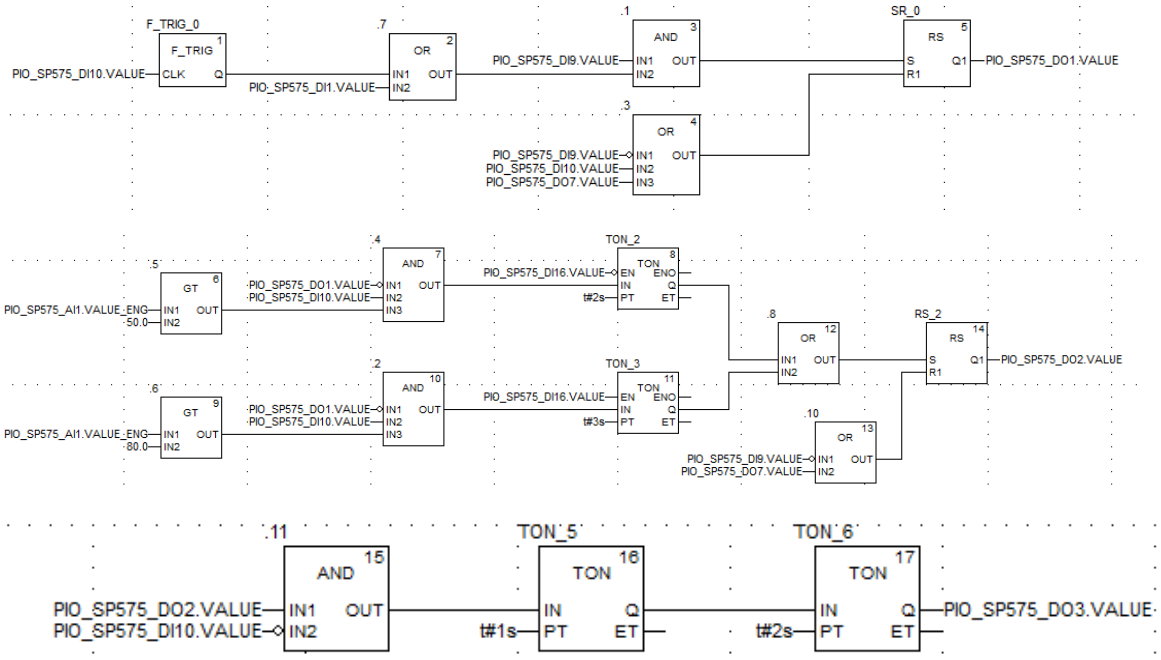
Stage Four: Fault Detection

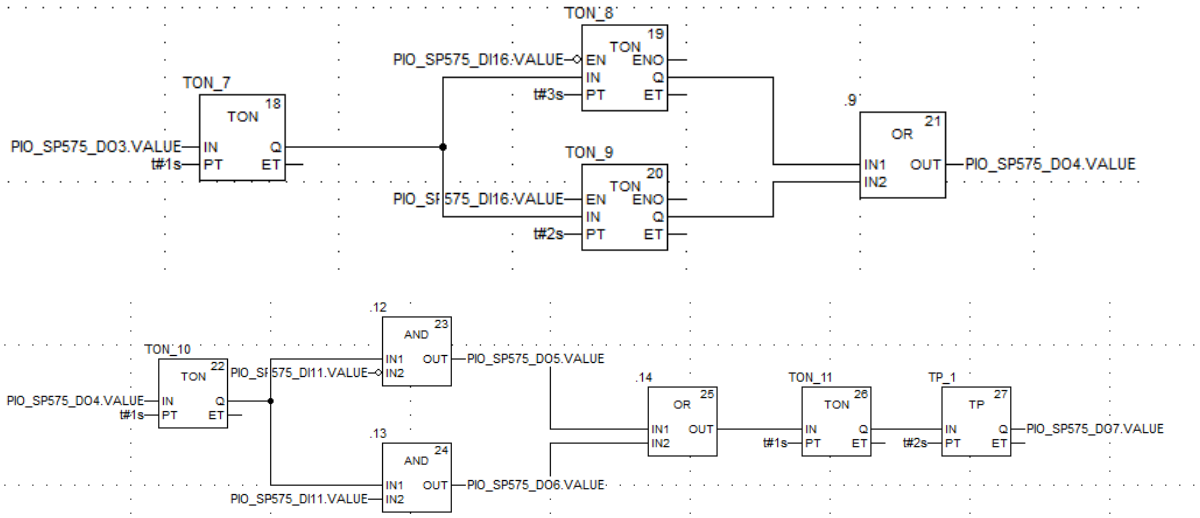
Once more, the object (can or bottle) takes 1 second to reach the vision sensor. The vision sensor analysis the object, if any fault is detected, the object moves to the recycle station, otherwise it moves to the packing area. Two outputs (LEDs) are implemented in this stage, one for each object. When the object has faults, the output (LED) is activated and remain ON.

The object takes another 1 second to reach destination. After this time, a final LED lights up indicating that the production line has complete the whole process. This output (LED) stays active for 2 seconds and goes OFF automatically. During this time, all outputs (LEDs) are automatically reset as well.

Note: the function block TP (timer pulse) generates a pulse signal with the duration of the pre-set timer.

Example code for FBD





Exercises

Go through the code developed and apply some modifications to implement the following exercises:

1. How should the third and fourth optical sensor be inserted into the code? Write it down how would you describe the scenario with those sensors.
2. Go to the Function Block Input Assistant, search and add the function block MOVE. Connect the Analog Input 1 (AI1) at the input pin and the Analog Output 1 (AO1) to the output pin. Note: make the necessary modifications in Remote Connect to enable this connection.
3. Remove the GT (greater than) function block from your code. Add another input pin to the AND gate (IN4), associate this pin with PIO_SP575_AI1.H4_STATE (bottle) and PIO_SP575_AI1.H3_STATE (can), in the respectively position. What do these variables represent? Where were these variables configurated?
4. In your own words, how would you describe/implement a fully automated process, in which more than one object (can or bottle) can be at the production line?