DSEGenset®

DSEControl®

DSE®

DEEP SEA ELECTRONICS

Advanced PLC Programming Guide for DSE Modules

Document Number: 057-314

Author: John Ruddock

CULTURE OF QUALITY

Deep Sea Electronics Ltd.
Highfield House
Hunmanby
North Yorkshire
YO14 0PH
England

Sales Tel: +44 (0) 1723 890099

E-mail: sales@deepseaelectronics.com
Website: www.deepseaelectronics.com

Advanced PLC Programming Guide for DSE Modules

Amendments Since Last Publication

Amd. No.	Comments
1	First release
2	Added DSEE400, general improvements and added more detail

Typeface: The typeface used in this document is *Arial*. Care must be taken not to mistake the upper-case letter I with the numeral 1. The numeral 1 has a top serif to avoid this confusion.

TABLE OF CONTENTS

SECTION PAGE

1 INTRODUCTION

NOTE: It is the responsibility of the PLC programmer to ensure that the PLC program operates exactly as intended. DSE cannot be held responsible for any issues arising from unintended actions of the PLC program.

This guide covers the configuration of the *Advanced PLC Editor* within DSE modules. Refer to section entitled *DSE Module Compatibility* in section 1.4 of this document for a list of modules that support this feature.

The internal PLC allows the system designer to add functionality to the DSE module where such functions do not already exist. It also allows the designer to take existing functions within the module and tailor them to suit the application requirements.

The main point to remember with the PLC is that the designer is not changing existing functions within the DSE module, rather they are using them in differing ways to help ensure that DSE's high level of protection and safety cannot be bypassed with the PLC. However, great care must still be taken to ensure the PLC program operates as required by the designer.

For example, the DSE8610 MKII module contains synchronising and load sharing functions, with protections provided by the MultiSet Communications (MSC) link. It would be inappropriate to allow designers to bypass these protections. However, to allow customisation, DSE have provided digital input functions to alter the process while maintaining all necessary protections. These input functions are also accessible via PLC and the user can create their own bespoke functions, without compromising the safety of the module.

1.1 CLARIFICATION OF NOTATION

Clarification of notation used within this publication.

NOTE:	**Highlights an essential element of a procedure to ensure correctness.**
CAUTION!	**Indicates a procedure or practice, which, if not strictly observed, could result in damage or destruction of equipment.**
WARNING!	**Indicates a procedure or practice, which could result in injury to personnel or loss of life if not followed correctly.**

1.2 GLOSSARY OF TERMS

Term	Description
Boolean	A form of data with only two possible values "true" and "false"
MSC	Multi-Set Communication
GenComm	DSE map of all Modbus registers
Ladder rung	The PLC program is made up of ladder rungs
Normally closed	The condition is true when the item is in-active
Normally open	The condition is true when the item is active
PLC	Programmable Logic Controller A programmable digital device used to create logic for a specific purpose.
SCADA	Supervisory Control and Data Acquisition A system that operates with coded signals over communication channels to provide control and monitoring of remote equipment

1.3 BIBLIOGRAPHY

This document refers to, and is referred by the following DSE publications which are obtained from the DSE website: www.deepseaelectronics.com or by contacting DSE technical support: support@deepseaelectronics.com.

1.3.1 MANUALS

Product manuals are obtained from the DSE website: www.deepseaelectronics.com or by contacting DSE technical support: support@deepseaelectronics.com.

DSE Part	Description
057-238	DSE8610 MKII Configuration Suite PC Software Manual
057-254	DSE8610 MKII Operators Manual
057-257	DSE8660 MKII Configuration Suite PC Software Manual
057-259	DSE8660 MKII Operator Manual
057-251	DSEE400 Configuration Suite PC Software Manual
N/A	DSEGenComm (MODBUS protocol for DSE modules)

1.3.2 THIRD PARTY DOCUMENTS

The following third-party documents are also referred to:

Reference	Description
ISBN-10: 061565438X	A beginner's guide to Programmable Logic Controllers
IEC 61131-3	International standard for programmable logic controllers

1.4 DSE MODULE COMPATIBILITY

NOTE: Editing PLC values from the modules front panel is only possible in stop mode.

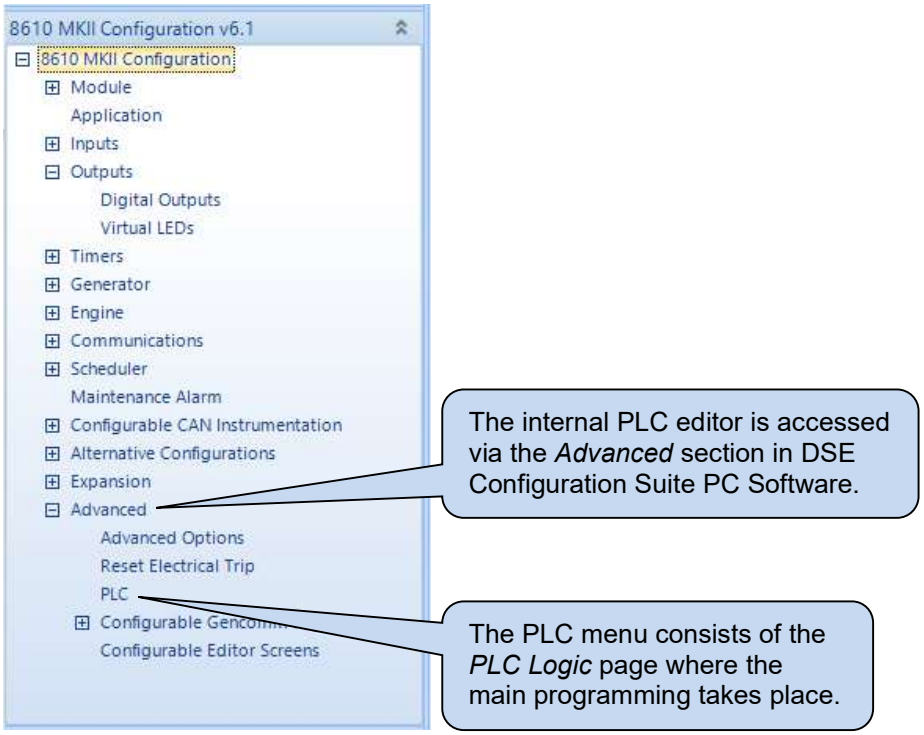
At the time of writing, the following modules include the advanced internal PLC with the following features:

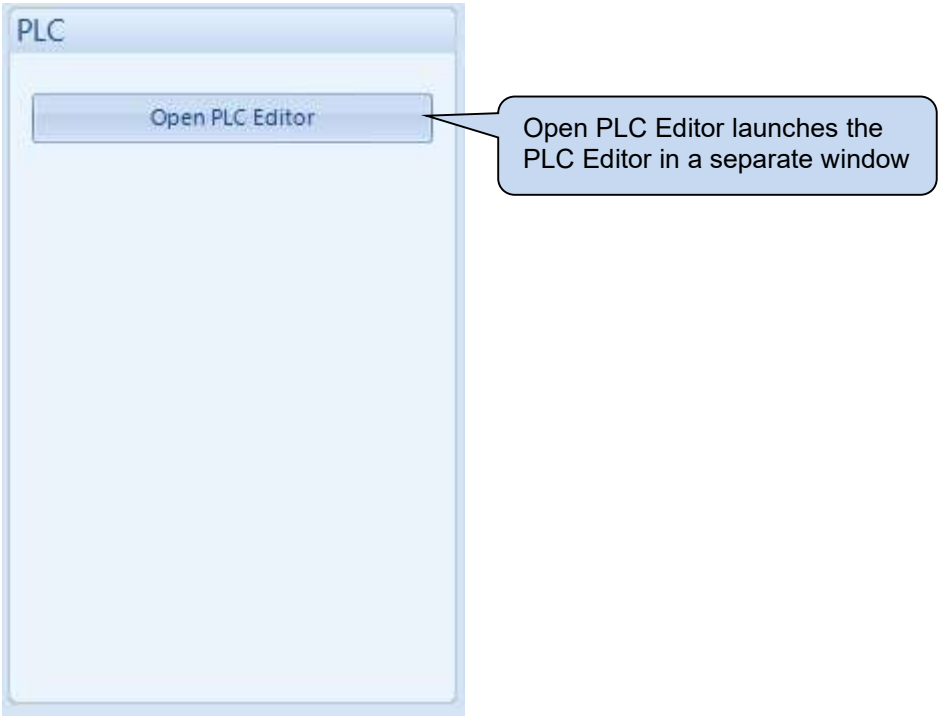
Feature	DSE8610 MKII DSE8660 MKII	DSEE400	DSE7310 MKII DSE7320 MKII	DSE7410 MKII DSE7420 MKII
Number of Nodes	2048	400	2048	2048
Variables	100	100	100	100
Persistent Variables	100	100	100	100
User Plc Coils	100	100	100	100
Timers	50	50	50	50
Counters	50	50	50	50
Virtual Inputs	20	20	20	20
Label Characters	20480	4096	20480	20480
Contact Test	✓	✓	✓	✓
Coil Set	✓	✓	✓	✓
Coil Reset	✓	✓	✓	✓
Coil Drive	✓	✓	✓	✓
Coil Toggle	✓	✓	✓	✓
Calendar Test	✓	✓	✓	✓
Instrumentation Test	✓	✓	✓	✓
Button Press Test	✓	✓	✓	✓
Import PLC	✓	✓	✓	✓
Export PLC	✓	✓	✓	✓
Alarm Reset	✓	✓	✓	✓
Editable Timer and Counter Names	✓	✓	✓	✓
Editable Coil Names	✓	✓	✓	✓
PLC SCADA	✓	✓	✓	✓
Module Display of Counters & Timers	✓	✓	✓	✓
Module Display of Variables & Persistent Variables	✓	✓	✓	✓
Front Panel Editing of Counter and Timer Set Points	✓	⊖	✓	✓
Front Panel Editing of Variables	⊖	⊖	⊖	⊖
Front Panel Variable Format	⊖	✓	⊖	⊖
Front Panel Editing of Persistent Variables	✓	⊖	✓	✓
Maths Functions	✓	✓	✓	✓
Clock Adjust (+/- 1hr)	✓	✓	✓	✓
Logical AND, OR, NOT & Exclusive OR gates	✓	✓	✓	✓
User defined Function Blocks	✓	✓	✓	✓
Rising Edge Trigger	✓	✓	✓	✓
Falling Edge Trigger	✓	✓	✓	✓
Delay On Timer	✓	✓	✓	✓
Delay Off Timer	✓	✓	✓	✓
Built in PID Controller	✓	✓	✓	✓

Feature	DSE8610 MKII DSE8660 MKII	DSEE400	DSE7310 MKII DSE7320 MKII	DSE7410 MKII DSE7420 MKII
Pulse Generator	✓	✓	✓	✓
Logic Selector	✓	✓	✓	✓
Value Selector	✓	✓	✓	✓
String Selector	✓	✓	✓	✓
Sentinel Checker	✓	✓	✓	✓
Bit Shift	✓	✓	✓	✓
User Defined Function Block	✓	✓	✓	✓
Read GenComm Data	✓	⊖	✓	✓
Inter Module Communications RS485	✓	⊖	✓	✓
Inter Module Communications TCP	✓	⊖	⊖	✓
Inter Module Communications MSC	✓	⊖	⊖	⊖
Read MSC Data	✓	⊖	⊖	⊖
Override GenComm	✓	⊖	✓	✓
Bitwise AND	✓	⊖	✓	✓
Bitwise OR	✓	⊖	✓	✓
Rate of Change	⊖	✓	⊖	⊖
Pin protected PLC read	⊖	✓	⊖	⊖

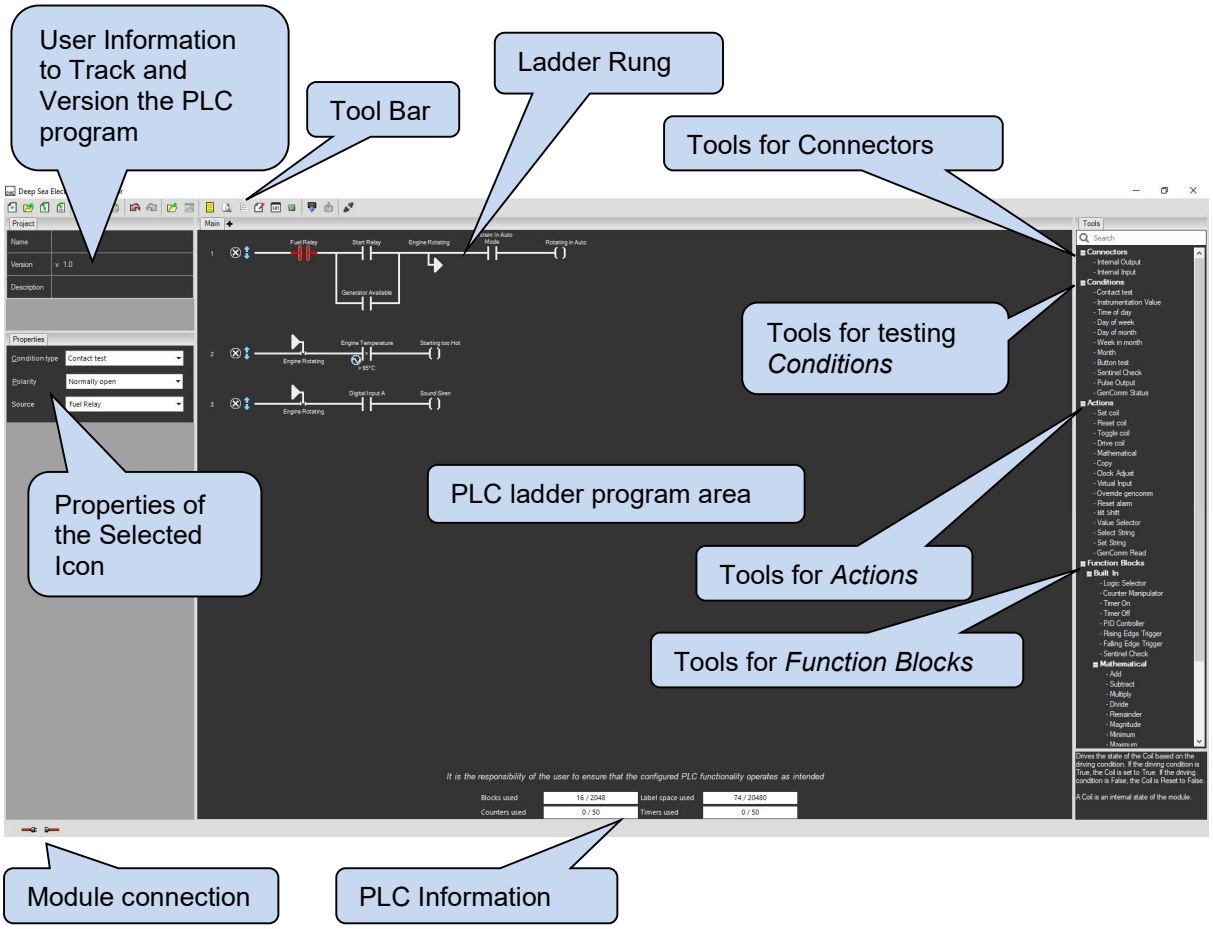
2 ACCESSING THE PLC EDITOR

To access the internal PLC, the DSE Configuration Suite PC Software must be installed and a Configuration from a supported module must be opened. For details of this refer to DSE Publication *057-151 DSE Configuration Suite PC Software Installation & Operation Manual* available from www.deepseaelectronics.com.

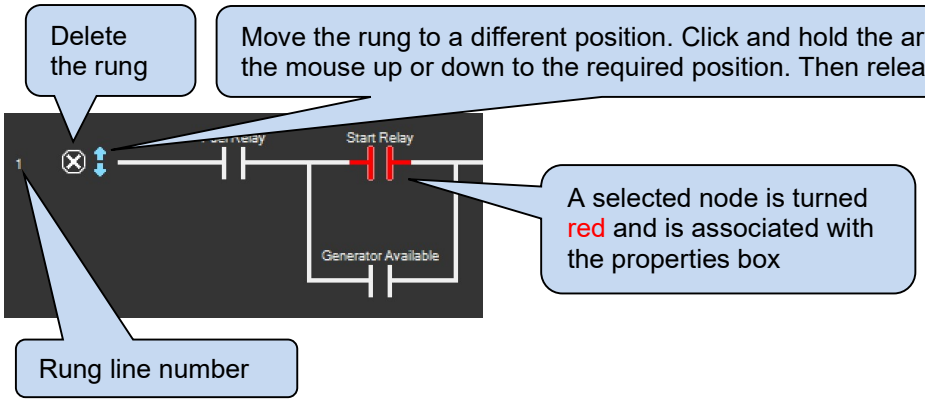




3 PLC EDITOR



To create a program, drag the tools from the *Connector*, *Condition, Actions, Function Blocks* or *Logic* toolbar onto the PLC Ladder Area to build up the required PLC function



The PLC program is evaluated every 100 ms (10 times per second).
The evaluation time of each rung is indeterminate as each rung consists of a variety of functions, each one with a variable execution time.

Actions are 'queued' by the PLC during the evaluation of the rungs, then executed *in order* at the completion of the program before the cycle begins again.
This means that the order of the items in the PLC may change the way the program operates.

3.1 TOOL BAR

The first 9 items on the tool bar follow windows convention and are not explained in detail. The remainder of the items are explained in more detail.

Parameter	Description
	New. Delete all the PLC program from the editor and start a new one.
	Open. Open a previously saved PLC program from disk. This deletes the existing PLC program and replaces it with the opened PLC program. Compatible with new and old PLC programs
	Save. Save the entire PLC program as the existing file name to disk
	Save As. Save the entire PLC program to disk with a new file name.
	Cut. Cut the selected PLC ladder from the editor
	Copy. Copy the selected PLC ladder from the editor
	Paste. Paste the cut or copied selected PLC ladder to the editor
	Undo. Undo the last edit
	Redo. Redo the last undo
	Import PLC. Import a saved to disc PLC program, and adds it to the existing PLC program. Compatible with new and old PLC programs
	Export PLC. Export the highlighted portion of the PLC program to disc
	Add Label. Add a line of text to the PLC editor. This allows description of PLC function.
	Find Item Usage. Shows a list and their position of currently used *Virtual Inputs, Coils, Counters, Timers, Variables, Persistent Variables,* and *strings*. This makes it easier to find where they are within the PLC program
	Show/Hide Line Numbers. Turns on and off PLC line numbers
	Virtual Inputs. View and edit the configuration of virtual inputs
	Variables. View, add and edit coils, counters, persistent variables, timers, strings, and variables.
	System Device Addresses. View, add and edit the system device addresses
	Import PLC from Module. Imports the PLC from the connected module into the editor. This deletes the existing PLC program and replaces it with the imported PLC program
	Set Persistent Variable. View and edit all the persistent variables settings. Only available when the editor is connected to a module
	Clear Pin. Clears the PLC pin if one is set
	Pin. Sets a PIN number that prevents PLC from being displayed and edited.
	Connect Scada. Connects the editor to the connected module and opens SCADA monitoring

3.1.1 IMPORT PLC

Allows the designer to import a PLC program that is saved to disk into the current program. It converts any saved PLC programs in the older format into the new PLC format. This is useful for reusing functions created in other configurations.

Clicking the import PLC icon imports the PLC into the bottom of the existing PLC logic. Dragging the icon into the PLC editor allows the user to place the imported PLC within the existing PLC logic.

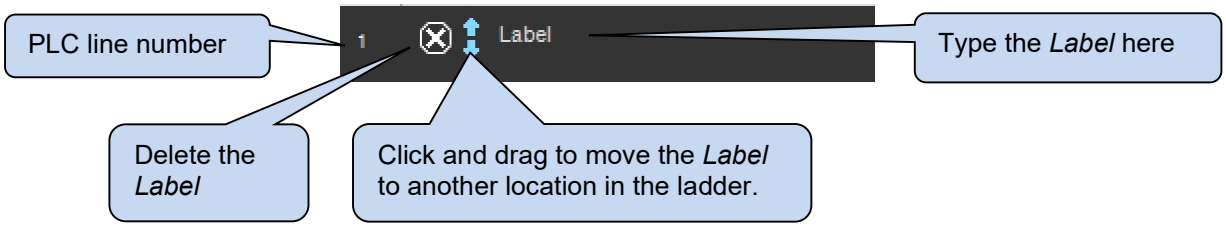
All variables, persistent variables, timers, counters, strings, and coils are also copied into the PLC program.

3.1.2 EXPORT PLC

Allows the designer to export a selected part of the PLC program. Export PLC saves the selected PLC program to disk

3.1.3 ADD LABEL

Drag *Add Label* to the desired position in the PLC editor and a blank label is created. This allows the designer to place notes in the PLC Ladder.

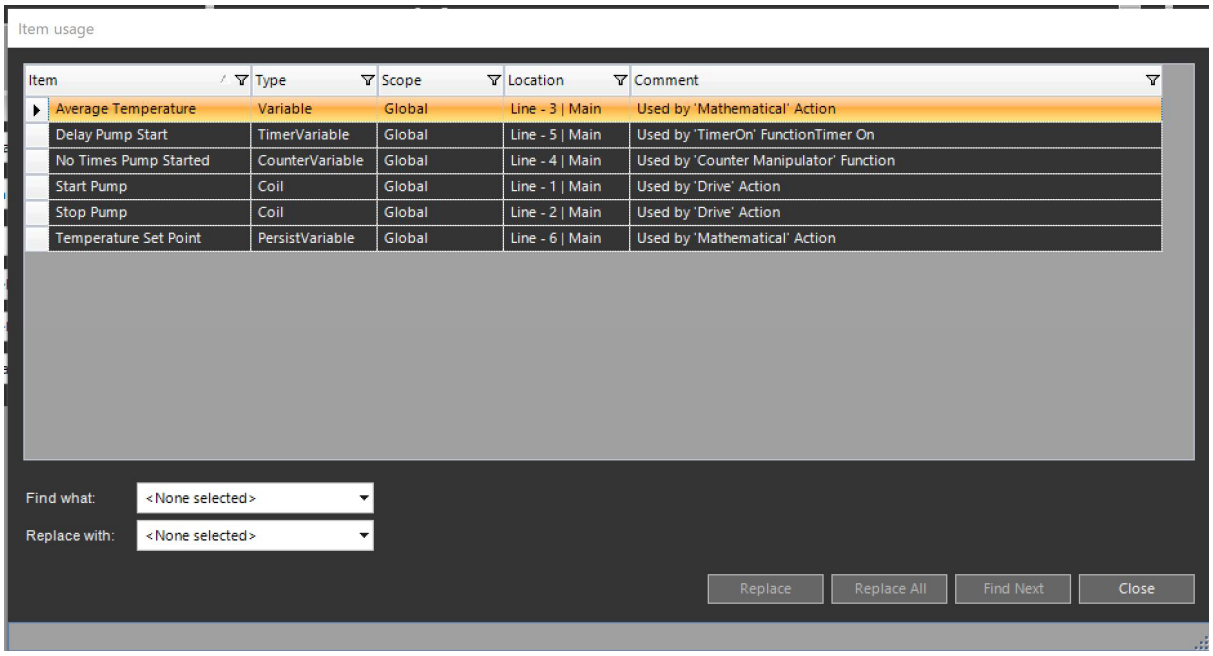


3.1.4 FIND ITEM USAGE

Shows a list of currently used *Coils*, *Virtual Inputs, Counters, Timers, Variables*, *Persistent Variables,* and *strings*

The table shows the variable name, the type of variable, the scope of variable, its location within the PLC and comments.

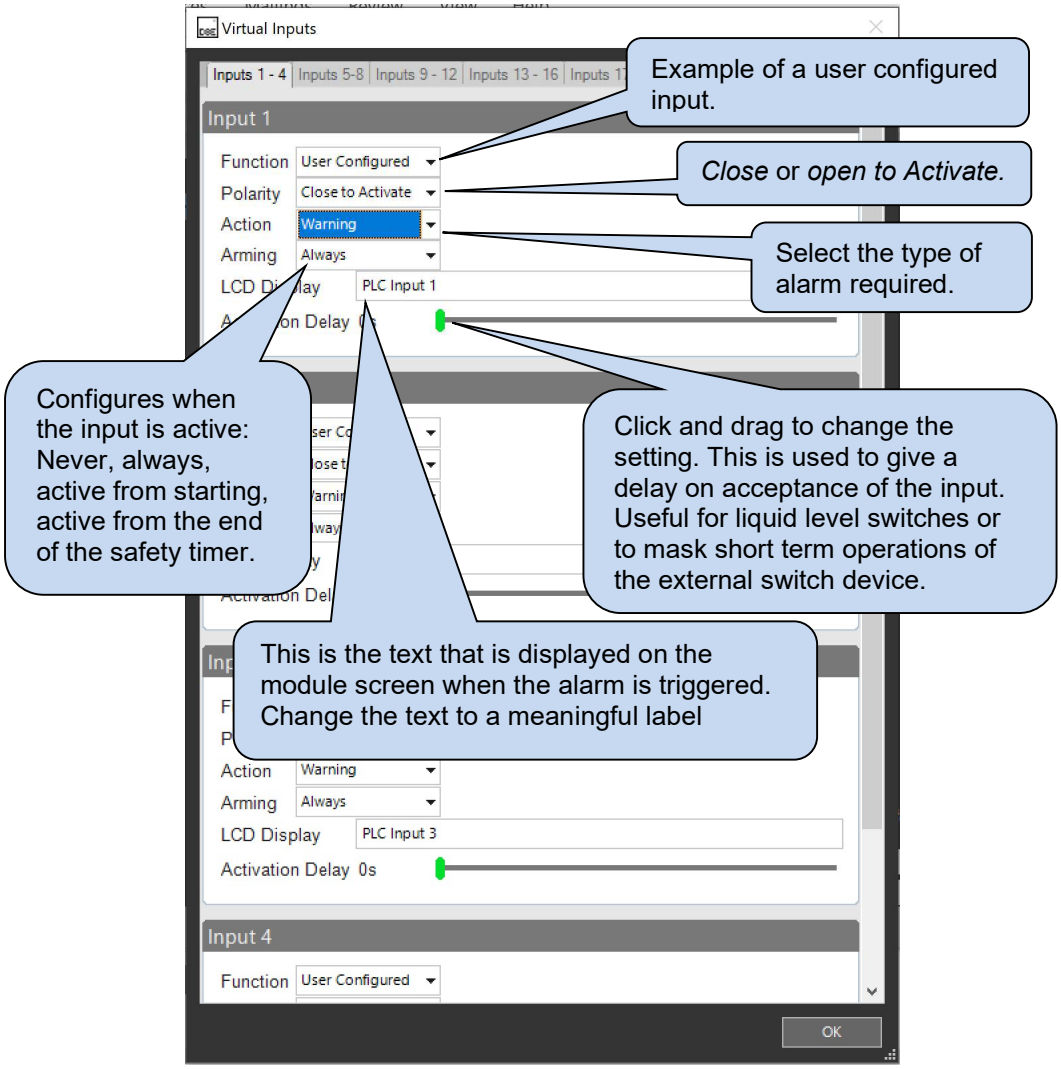
The table also allows the user to change one variable for another using the *Find What* and *Replace with* drop down boxes.



3.1.5 VIRTUAL INPUT

Virtual Inputs are configured in exactly the same way as the modules Digital Inputs. The difference is that the PLC Virtual Input is activated by the PLC and does not require hard wiring. In addition, it does not 'use up' one of the module's hardware inputs.

Select the Virtual Inputs Icon which opens the virtual input settings window. PLC Functions have exactly the same choices as module digital inputs.



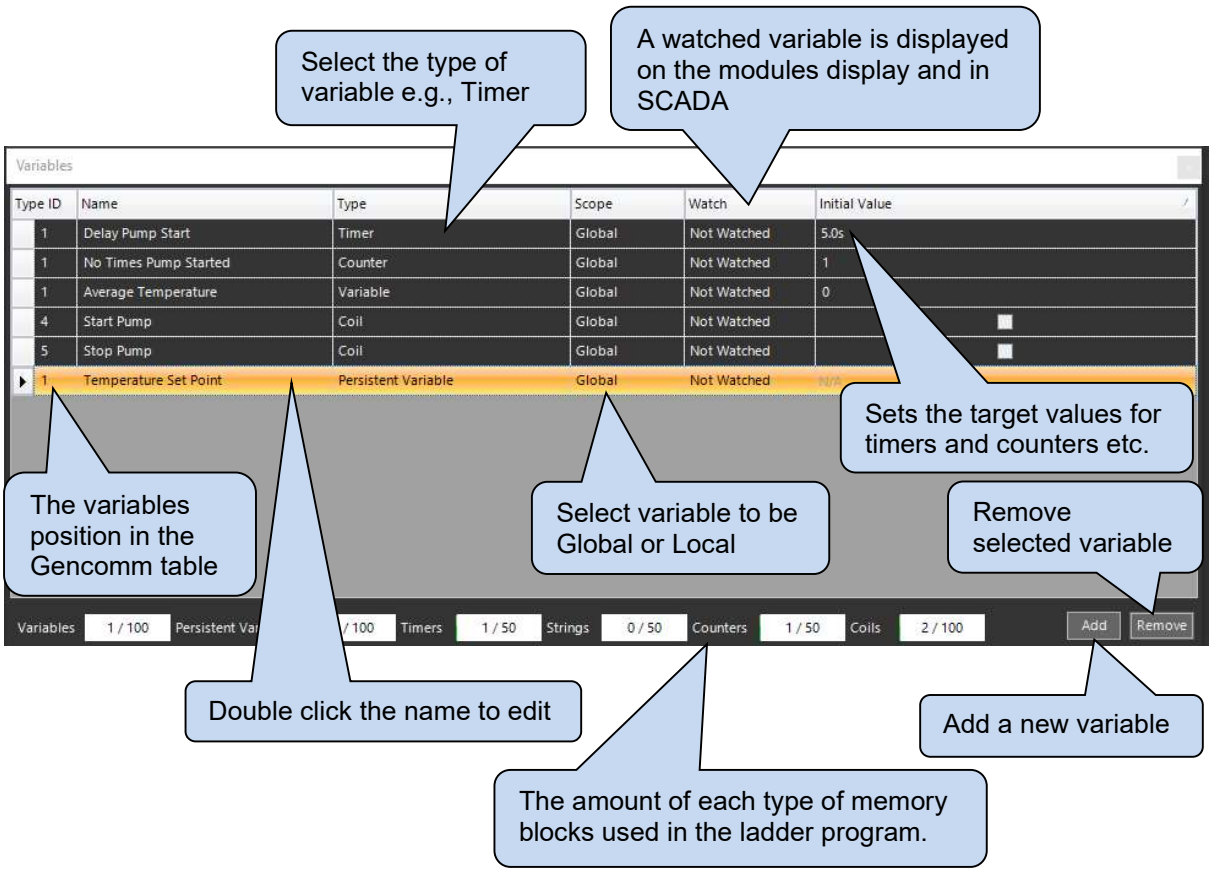
3.1.6 VARIABLES

NOTE: There are two categories for declaring the scope of the variable.

Global: This is a variable that can be used throughout the PLC program.
Local: This is a variable that can be used in a User Defined Function.

See section entitled *User Defined Functions* elsewhere in this document for further information.

Variables is where all the names for *Coils Counters, Timers, Strings, Variables* and *Persistent Variables* are edited and their settings such as timer values adjusted.



3.1.6.1 TYPE

Within the DSE PLC, the user has the option to created *Variables* of different types. These different *Variables* types are listed below .

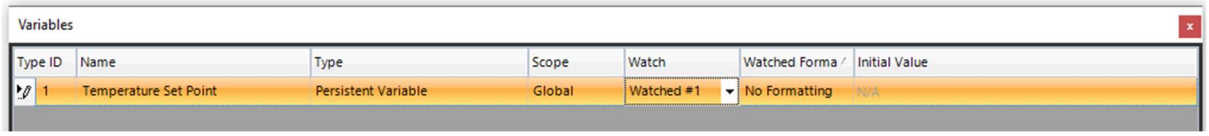
Parameter	Description
Coil	Coils can be considered as *Status Items* within the DSE module. Any operating state or alarm that occurs can be detected by the PLC program. Decisions can then be made as to what action to perform upon particular conditions. It is also possible to create 'user Coils to store the result of a condition or set of conditions. These are known as "Coils" and should be named to suit its function. Module outputs can then be set to operate upon the *Coils* activation. These *Coils* are often called *Output Sources.* Coils should be named to suit their function.
Counter	A Counter is a set value used as a limit. If the limit is reached the output (Q) value switches state.
Persistent Variable	Values placed in Persistent Variables are maintained, even when the module DC power is removed. The values are stored in Non-Volatile (N.V.) memory. To minimise the number of writes to the N.V. memory (extending its life), the values are stored at intervals of one minute since the last write to the N.V. memory and then only if the value has changed. This does not prevent the PLC updating the used value as often as it needs to. Persistent Variables can be viewed and edited in the module instrumentation screens after selection using the DSE Configuration Suite PC Software. For further details, refer to the section entitled *Watched Items* found elsewhere in this document. Persistent Variables should be named to suit their function.
String	A String is a text variable that can display text strings.
Timer	A Timer is a set value used as a limit. If the limit is reached the output (Q) value switches state.
Variable	Values placed in the Variables are lost when the module DC power is removed and after configuration upload from the DSE Configuration Suite PC Software. Variables can be viewed in the module instrumentation screens after selection using the DSE Configuration Suite PC Software. For further details, refer to the section entitled *Watched Items* found elsewhere in this document. Variables should be named to suit their function.

3.1.6.2 WATCHED ITEMS

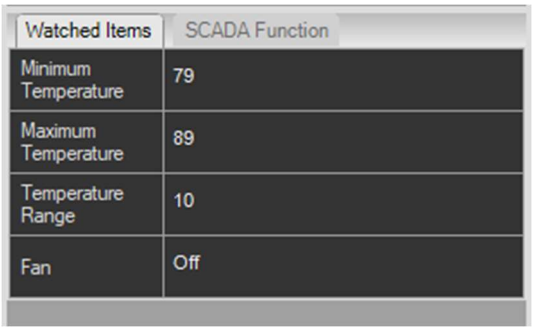
Any items that are watched automatically appear on the modules display. It is also possible to adjust watched *Persistent Variables* from the modules display. This is ideal for user defined set points, e.g. engine temperature, however it is good practice to restrict the range of adjustment.

Restricting the range is easily achieved using the copy function. Refer to section entitled *Copy* else where in this document.

Configure a watched *Persistent Variable and* set it to be watched. Up to sixteen items can be watched.



A table in PLC Scada is also displayed containing all watched items.



The module displays watched items as a range of instruments on the *PLC Instrument* page. Use the left and right arrow keys to scroll to *PLC Instruments*. Each instrument has its own screen, pressing the up and down keys navigates to the different instrument

After selection for display on the host module, *Counters, Timers, Variables, Persistent Variables* and *Strings* are able to be viewed and/or edited as below on the module display.

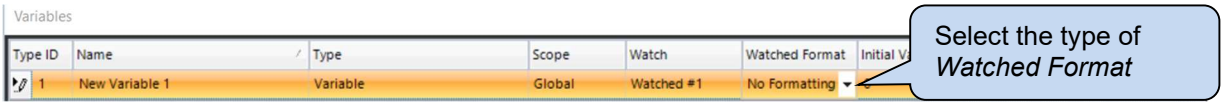
Parameter	View	Edit
Counter Value	✓	⊖
Counter Limit	✓	✓
Timer Value	✓	⊖
Timer Limit	✓	✓
Variable	✓	⊖
Persistent Variable	✓	✓
Strings	✓	⊖

To edit a value, navigate to the correct instrument, press the tick button, then use the up and down arrows to change the value. Press the tick button again to enter the value.

3.1.6.3 WATCHED FORMAT

NOTE: Changing the formatting of the *Variable* or *Persistent Variable* only affects how the value is displayed in SCADA or on the module's display.

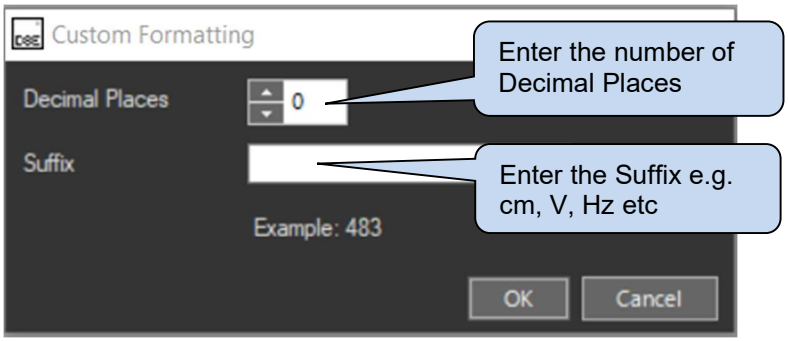
A watched *Variable or Persistent Variable* display format can be changed using the drop-down list in the *Watched Format section.*

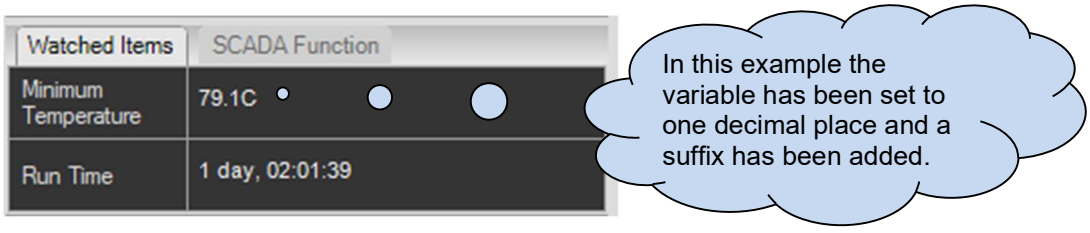


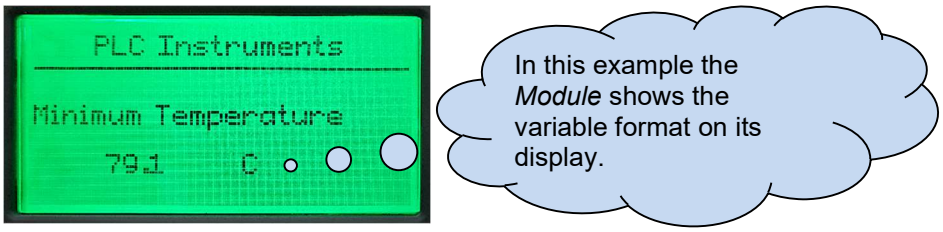
Parameter	Description
Watched Format	Formatting options avaialble are: ***No Formatting:*** No formatting will be applied to the variable ***Custom:*** Adjust the number of decimal places (0-2) and suffix ***Time:*** The time will be displayed in the following format dd,hh:mm:sec with a V*ariable* or P*ersistent* V*ariable* specified in tenths of a second

Custom Formatting

Selecting the *Custom* option displays the following dialog box.

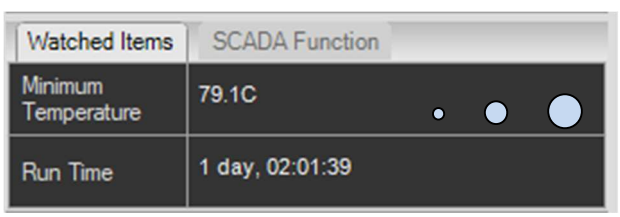






Time

The time is displayed in the following format.

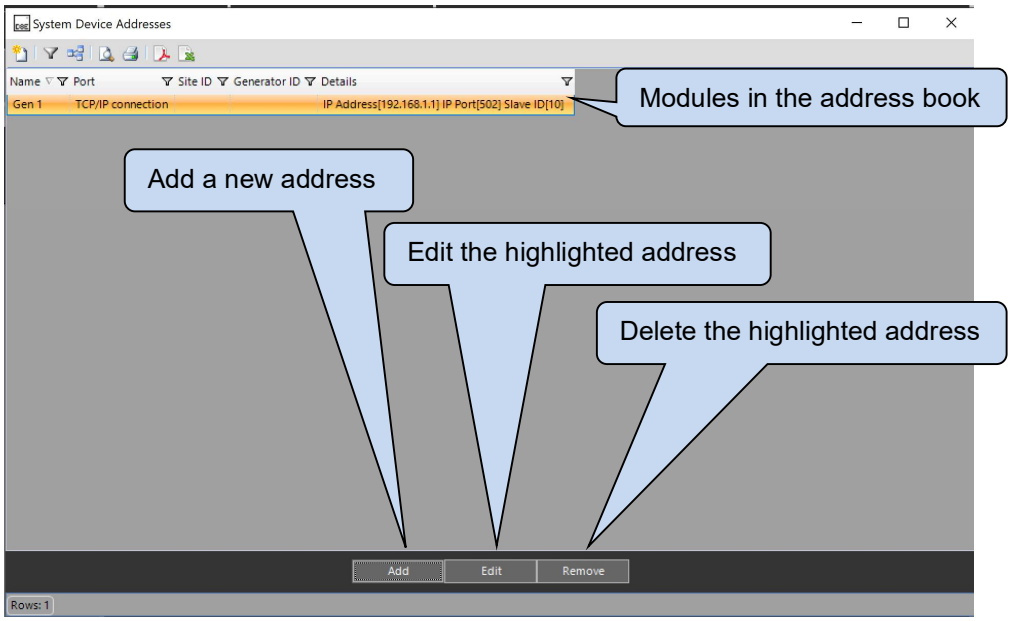


In this example the time is shown in the PLC Scada window.

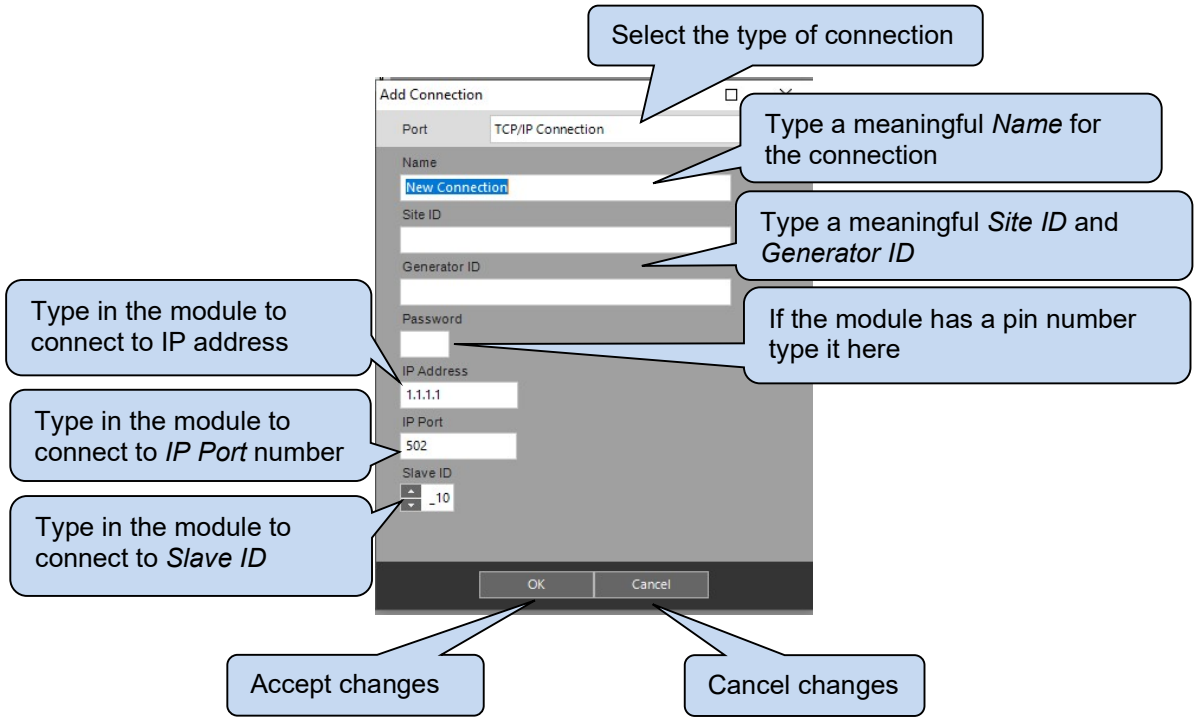
In this example the time is shown on the *Modules* display

3.1.7 SYSTEM DEVICE ADDRESSES

The PLC allows information from other DSE modules connected by ethernet or RS485 to be read into the PLC program. The *System Device Address* book is used to configure and edit all the external modules addresses.



Adding a new address or editing an existing address opens the following dialog box



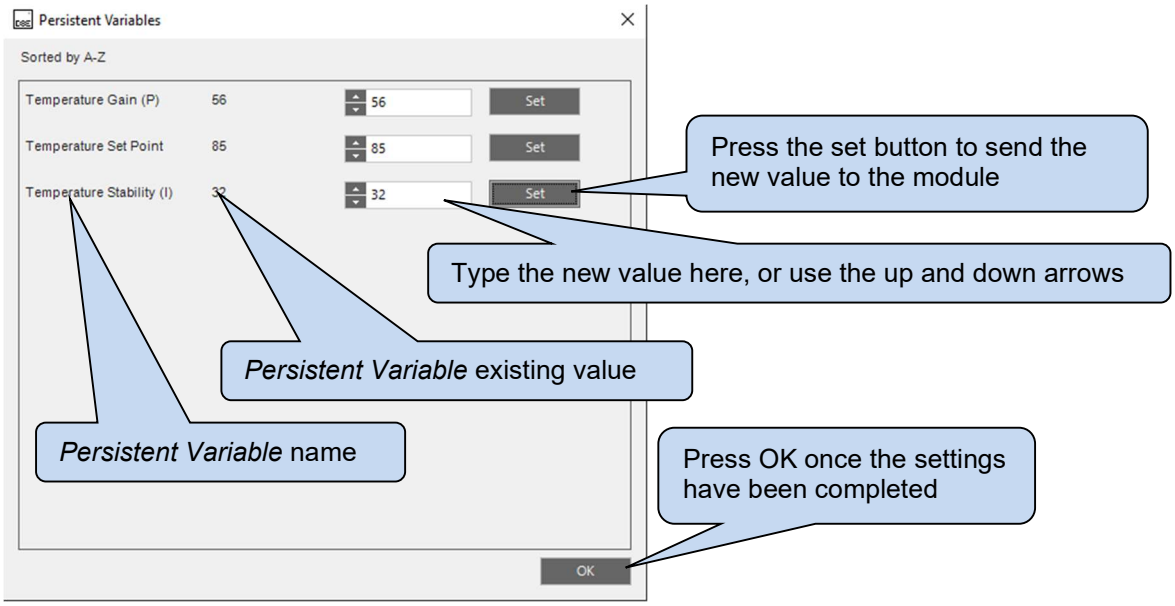
3.1.8 IMPORT PLC FROM MODULE

Imports the PLC program directly from a connected DSE module. This deletes the existing PLC program and replaces it with the opened PLC program.

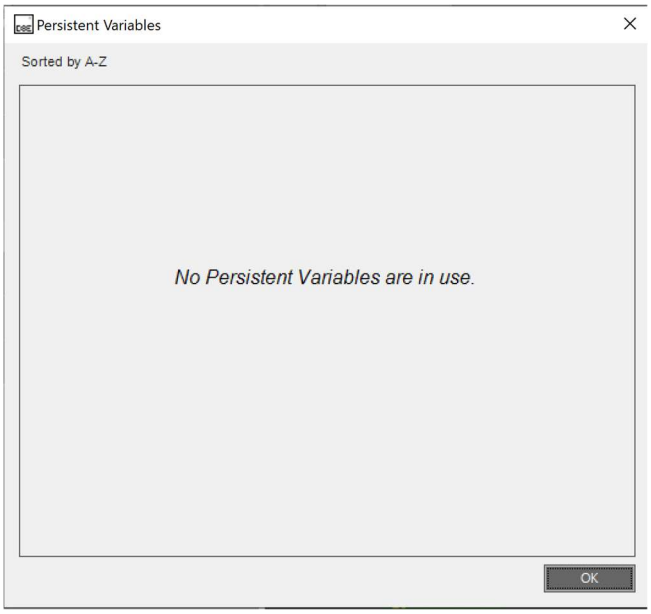
The connected module must be the same type as the PLC program that it is being imported to.

3.1.9 SET PERSISTENT VARIABLES

To enter a value into a PLC *Persistent Variable*, first connect to SCADA then click on the *Set Persistent Variable* icon  . A dialog box appears with a list of all the *Persistent Variable.*



If there are no Persistent Variables the dialogue box displays *No Persistent Variables are in use*



If a Persistent Variable is watched, its values can be adjusted from the modules control buttons. Please see section 3.1.6.2 in this manual, and the modules operator's manual.

3.1.10 CLEAR PIN

Clears the PLC PIN making it possible to view and edit PLC without requiring a pin.

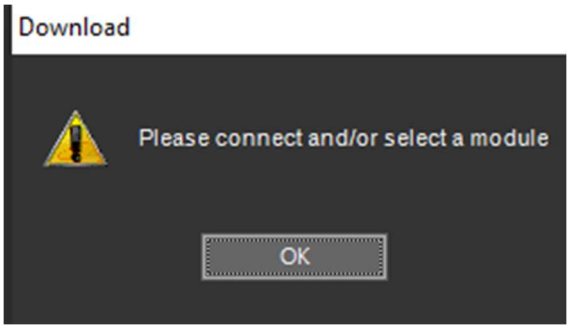
3.1.11 SET PIN

Use this icon to set a PIN so that the PLC cannot be viewed or edited without it. It also blocks viewing PLC SCADA. However, PLC SCADA still displays all watched items, and allow the editing of *Persistent Variables.*

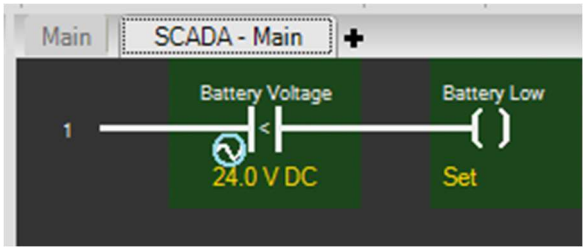
3.1.12 CONNECT SCADA

SCADA stands for Supervisory Control and Data Acquisition and is provided both as a service tool and as a means of monitoring and controlling the PLC program.
As a service tool, the SCADA pages are to check the operation of the PLC as well as checking and adjusting persistent variables.

Click the  icon to open the connection to the module. If no module is connected, the PLC editor warns that there is no module connected



When connection is made a new tab is opened in the PLC editor called SCADA Main and the operation of the PLC program can be viewed. The icon changes to 

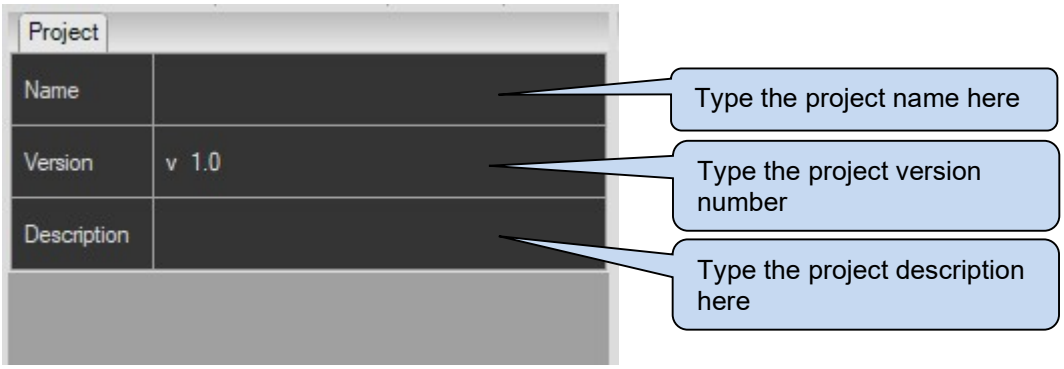


The module connection icon at the bottom right of the PLC changes from  to 

Please see section 5 in this manual.

3.2 PROJECT INFORMATION

The project information is populated with the PLC name, version number and a short description. This is saved with the config file and is uploaded to the module, so that the user can keep track of the PLC project



3.3 TOOLS

3.3.1 CONNECTORS

The connectors toolbar contains the tools that allow the PLC to pass logic from one rung to another. These are described in the following sections.

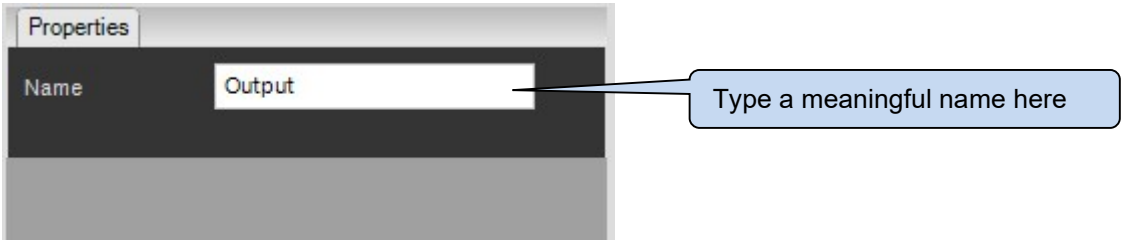
Parameter	Description
Internal Output	Pushes logic out of a rung
Internal Input	Pushes logic into a rung
External Output	Only available within *User Defined Function Blocks*. Pushes Logic out of a *User Defined Function Blocks*
External Input	Only available within *User Defined Function Blocks*. Pushes Logic into a *User Defined Function Blocks*

3.3.1.1 INTERNAL OUTPUT

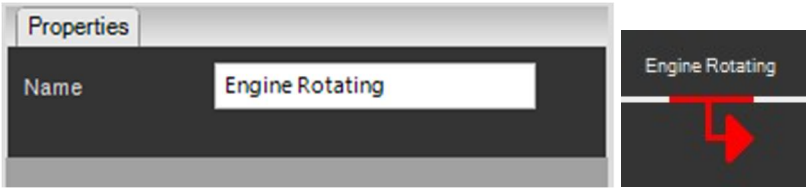
This function works in conjunction with the internal Inputs. It allows a rung state (true or false) to be passed to one or many Internal Inputs of the same name.

Drag the *Internal Output* 

into the relevant place on the ladder rung and configure its properties.

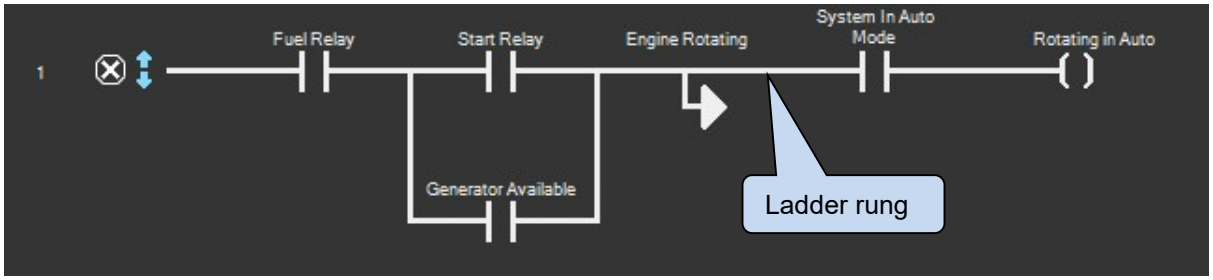


Parameter	Description
Name	The internal output's name. Rename to a meaningful name



Example

In the example below, the *Internal Output* is activated when the preceding logic is true.



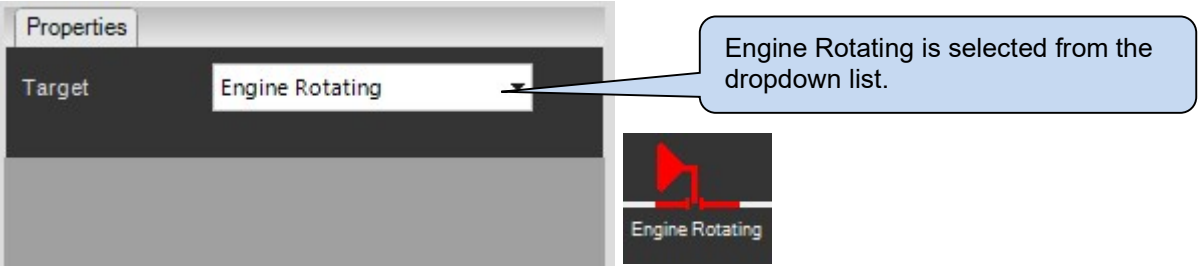
Multiple *Internal Outputs* can be used, but they must have unique names.

3.3.1.2 INTERNAL INPUT

Allows a rung state (true or false) to come from a selected *Internal Output*.

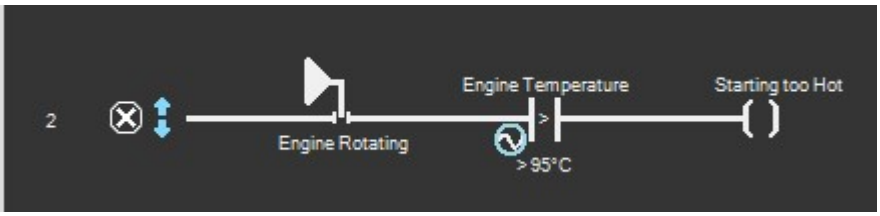
Drag the *Internal Input*  into the relevant place on the ladder rung and configure its properties.

Parameter	Description
Target	Select the *Internal Output* name that links to this *Internal Input*

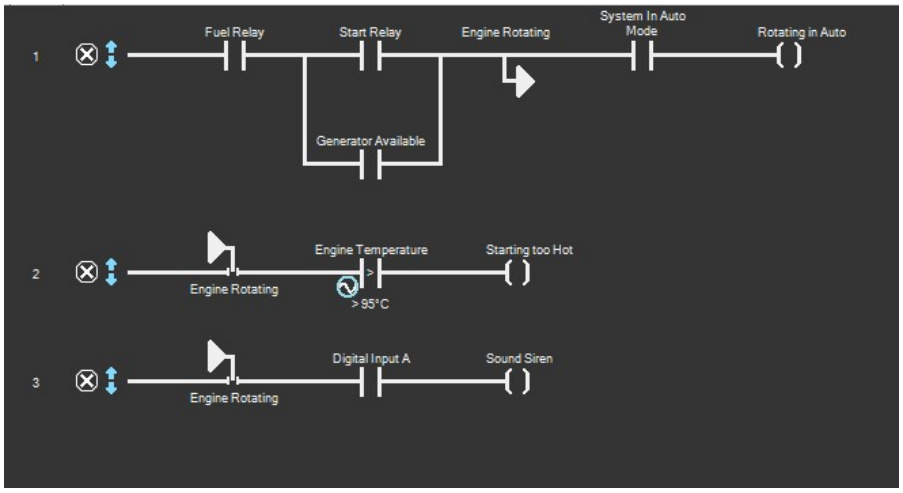


Example

In the below example the *Internal Input* is active when the *Internal Output* of the same name is active.



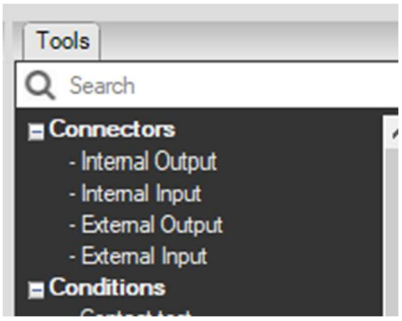
Each *Internal Output* can have multiple *Internal Inputs* associated with it.



3.3.1.3 EXTERNAL OUTPUTS

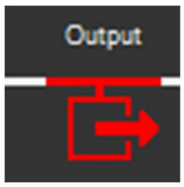
Only available within *User Defined Function Blocks.* Described in section 3.3.4.3 this manual.

When using *User Defined Function Blocks External Outputs* appears under *Connectors* in the *Tools* Menu

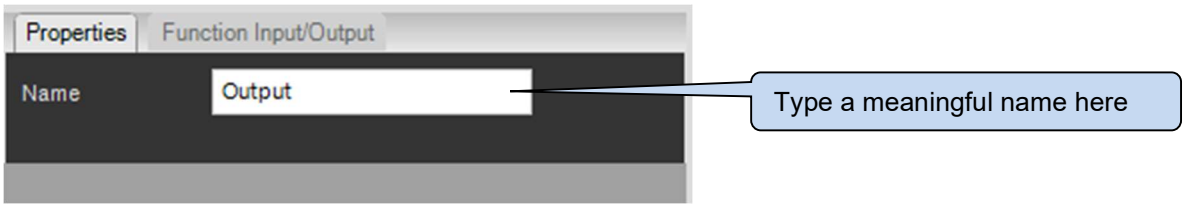


Allows a rung value to be passed out of a *User Defined Function Block*. Adding one of these to the edited function tab, adds an output to the function.

Its symbol is shown below



Drag the *External Output* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
Name	The external output's name. Rename to a meaningful name

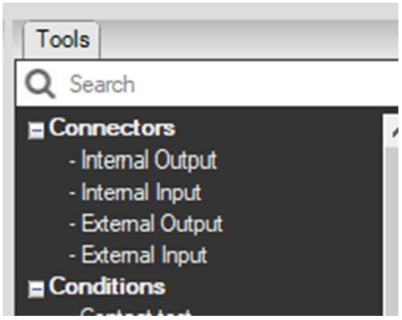
Example

See the section on *[New Function Blocks]* for further explanations

3.3.1.4 EXTERNAL INPUTS

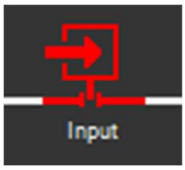
Only available within *User Defined Function Blocks.* Described in section 3.3.4.3 this manual.

When using *User Defined Function Blocks External Input* appear under *Connectors* in the *Tools* Menu

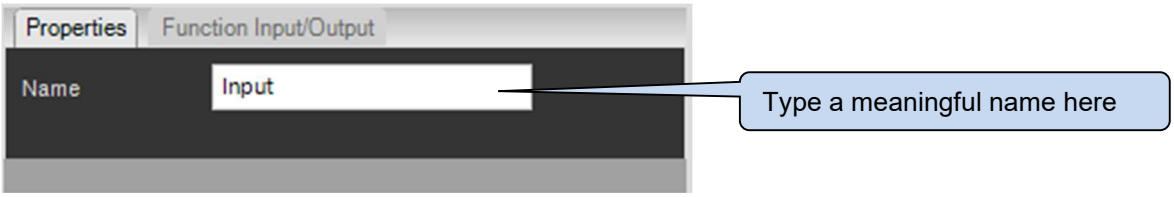


External Input allows a rung value to be passed into a *User Defined Function Block*. Adding one of these to the edited function tab, adds an input to the function.

Its symbol is shown below



Drag the *External Input* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
Name	The external output's name. Rename to a meaningful name

Example

See the section on *[New Function Blocks]* for further explanations

3.3.2 CONDITIONS

Items on the *Conditions* toolbar allow for a variety of conditions to be tested (checked). Not all items are available with all modules.

Parameter	Description
Contact test	Tests the state of a contact to determine if it is True or False.
Instrumentation Value	Tests the module's Instrumentation value against another value in several ways.
Time of Day	Test the module's internal clock for a specific Time of Day e.g. 10:32 am
Day of Week	Test the module's internal clock for a specific Day of Week e.g. Tuesday
Day of Month	Test the module's internal clock for a specific Day of Month e.g. 25th
Week in Month	Test the module's internal clock for a specific Week in Month e.g. Week 3
Month	Test the module's internal clock for a specific Month e.g. March
Button Test	Test the state of the modules buttons to determine if one is pressed
Sentinel Check	Checks the selected instrument to see if its current state is a Sentinel
Pulse Output	Adds a Pulse function to the selected Coil to change the Coil value from True to False at specified intervals.
GenComm Status	Checks that the Device selected from the System Device Addresses is responding to MODBUS queries.

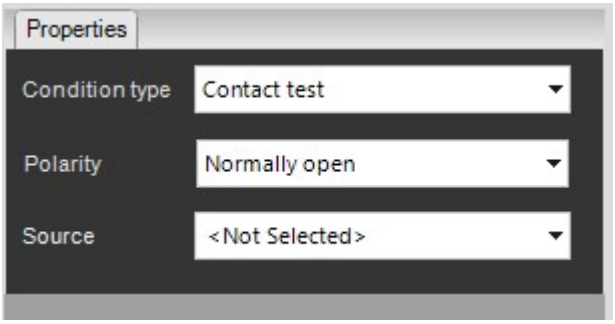
3.3.2.1 CONTACT TEST

A *Contact* is an internal state of the module. Some examples of contacts include operating mode and current alarm conditions. The list of testable flags varies depending upon the module being configured and is the same list for configuring the module's output relays. A full list along with descriptions is contained within the relevant DSE Configuration Suite PC Software Manual.

Complex ladder logic can be built using multiple series and parallel paths.

Drag the Contact test 

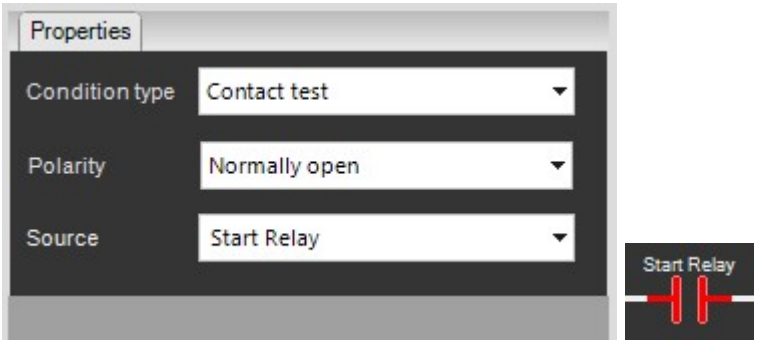
into the relevant place on the ladder rung and configure its properties.



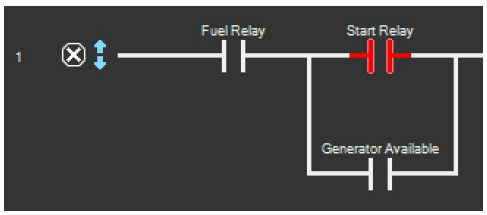
Parameter	Description
Condition Type	Select the *Condition type. Set automatically, but can be changed*
Polarity	Select the *Polarity* to be *Normally open* or *Normally closed*
Source	Select the conditions activation *Source*

Example

The below example shows a contact test configured for Start Relay



The example below shows the *Fuel Relay* and *Start Relay* in series forming an 'AND' function. The *Start Relay* and *Generator Available* are in parallel forming an 'OR' function

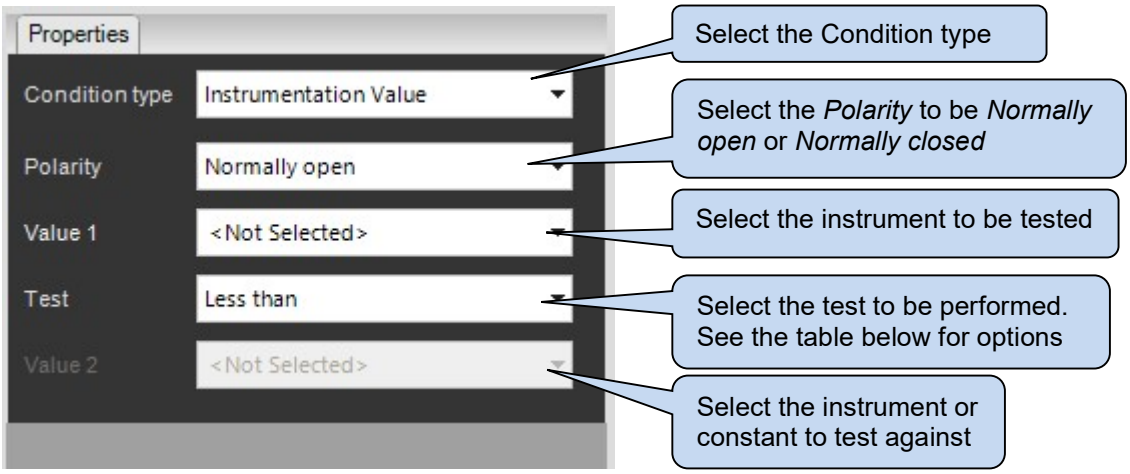


3.3.2.2 INSTRUMENTATION VALUE

Allows the module's instrumentation to be tested against another instrument or constant.

Drag the *Instrumentation Value* 

into the relevant place on the ladder rung and configure its properties.



Test	Mathematical Symbol
Less than	<
Less than or equal to	<=
Equal to	=
Greater than or equal to	>=
Greater than	>
Between	>= AND <=

The list of testable instruments varies depending upon the module. A full list of module instrumentation is contained within the relevant Operator Manual.

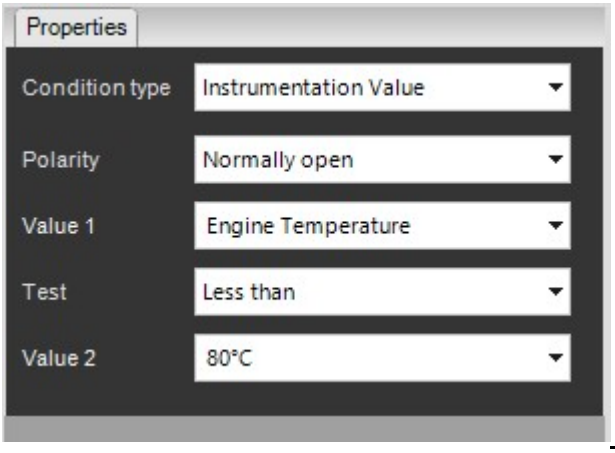
Not all instrumentation can be read from all modules. These include:

- Instruments that are not supported by the module. For example, "Mains Voltage" is only available in modules with Mains Sensing.
- Instruments not configured in the module. For example, the *Fuel Level Input* may be configured to be "*not used*".
- Instruments that are under range or over range. For example, if the *Coolant Temperature* is below the measurable range of the temperature sensor being used.
- Instruments that are in a fault condition. For example, the *Oil Pressure Sensor* may be "open circuit".
- Instruments whose condition cannot be determined. For example, *Power Factor* is not measurable when there is no load applied to the generator.

In these circumstances the module returns a *sentinel* value as listed overleaf. The actual value returned for a given state varies depending upon the size and type of instrument being read.

Example

The below example shows Engine Temperature being tested, for a temperature below 80°C



Sentinel Values For Instrumentation

Size of register	Sentinel Values (Hexadecimal)	Sentinel Values (Decimal)	Description
16 bit unsigned, any scale	0xFFFF	65535	Unimplemented
	0xFFFE	65534	Over measurable range
	0xFFFD	65533	Under measurable range
	0xFFFC	65532	Transducer fault
	0xFFFB	65531	Bad data
	0xFFFA	65530	High digital input
	0xFFF9	65529	Low digital input
	0xFFF8	65528	Reserved
16 bit signed, any scale	0x7FFF	32767	Unimplemented
	0x7FFE	32766	Over measurable range
	0x7FFD	32765	Under measurable range
	0x7FFC	32764	Transducer fault
	0x7FFB	32763	Bad data
	0x7FFA	32762	High digital input
	0x7FF9	327621	Low digital input
	0x7FF8	32760	Reserved
32 bit unsigned, any scale	0xFFFFFFFF	4294967295	Unimplemented
	0xFFFFFFFE	4294967294	Over measurable range
	0xFFFFFFFD	4294967293	Under measurable range
	0xFFFFFFFC	4294967292	Transducer fault
	0xFFFFFFFB	4294967291	Bad data
	0xFFFFFFFA	4294967290	High digital input
	0xFFFFFFF9	4294967289	Low digital input
	0xFFFFFFF8	4294967288	Reserved
32 bit signed, any scale	0x7FFFFFFF	2147483647	Unimplemented
	0x7FFFFFFE	2147483646	Over measurable range
	0x7FFFFFFD	2147483645	Under measurable range
	0x7FFFFFFC	2147483644	Transducer fault
	0x7FFFFFFB	2147483643	Bad data
	0x7FFFFFFA	2147483642	High digital input
	0x7FFFFFF9	2147483641	Low digital input
	0x7FFFFFF8	2147483640	Reserved

Test Calculated Value

NOTE: On some modules, *Calculated Values* are available in the *Test Instrumentation* section.

Calculated values are predefined by DSE and are included to provide additional ways of testing of the module's instrumentation.

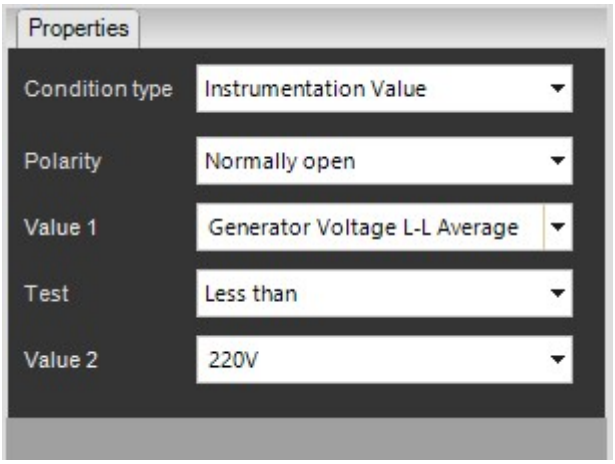
Selection	Description
Average	An average of the instrumentation
Difference	The difference between the maximum and the minimum
Maximum	The highest instrumentation value
Minimum	The lowest instrumentation value
Minimum Index	Indicates the lowest of the three phases (L1=1, L2=2, L3=3)
Maximum Index	Indicates the highest of the three phases (L1=1, L2=2, L3=3)

Example

L1 = 230 V AC
L2 = 233 V AC
L3 = 224 V AC

Results of the *Calculated Values* operators are as follows:

Average	Difference	Maximum	Minimum	Minimum Index	Maximum Index
229 ((230+233+224)/3)	**9** (233-224)	**233** Maximum Value	**224** Minimum Value	**3** L3 is the minimum of the three phases	**2** L2 is the maximum of the three phases

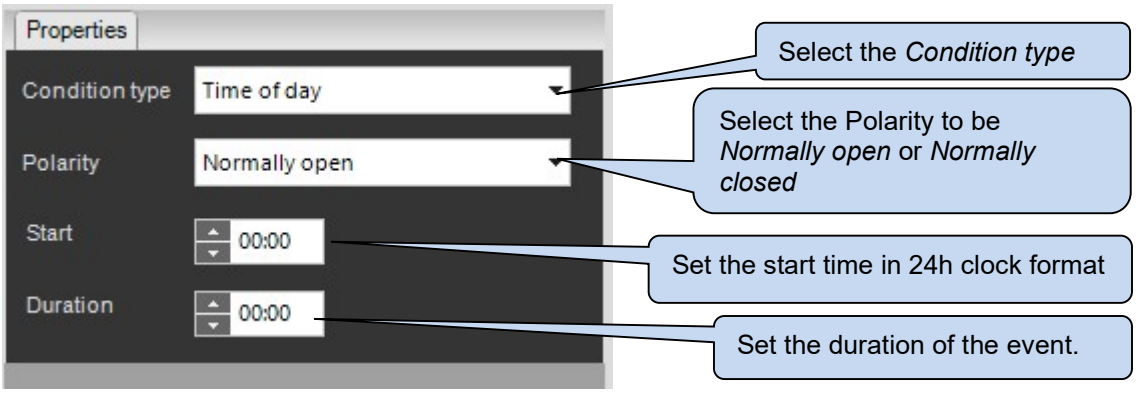


3.3.2.3 TIME OF DAY

Allows the module's time to be tested and allows an action based upon a specific time or time window.

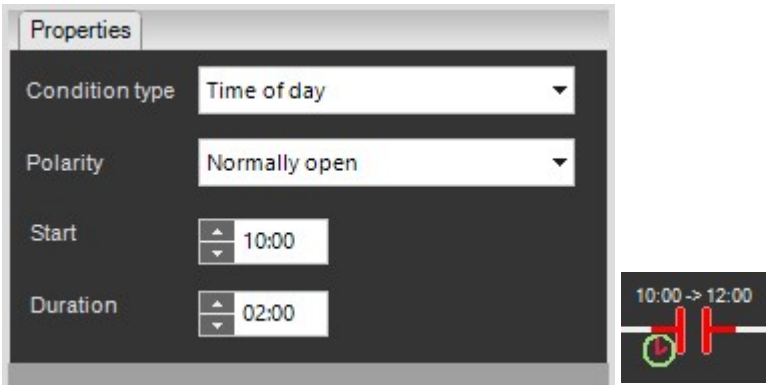
Drag the *Time of Day* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a time window that starts at 10:00 and lasts for 2 hours.

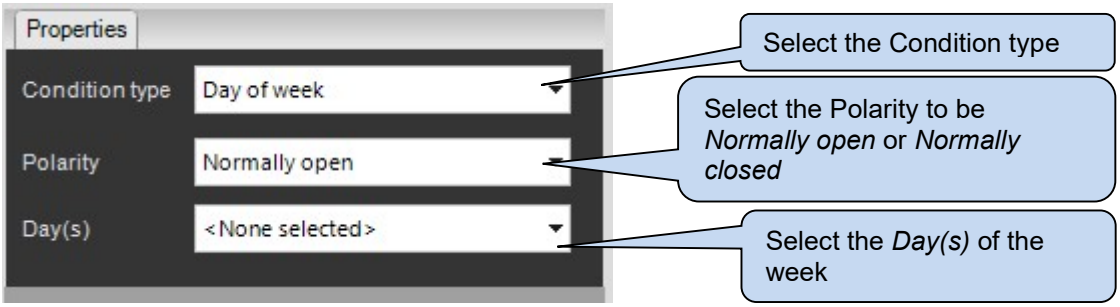


3.3.2.4 DAY OF WEEK

Allows the module's day to be tested and allows an action based upon a specific day or days.

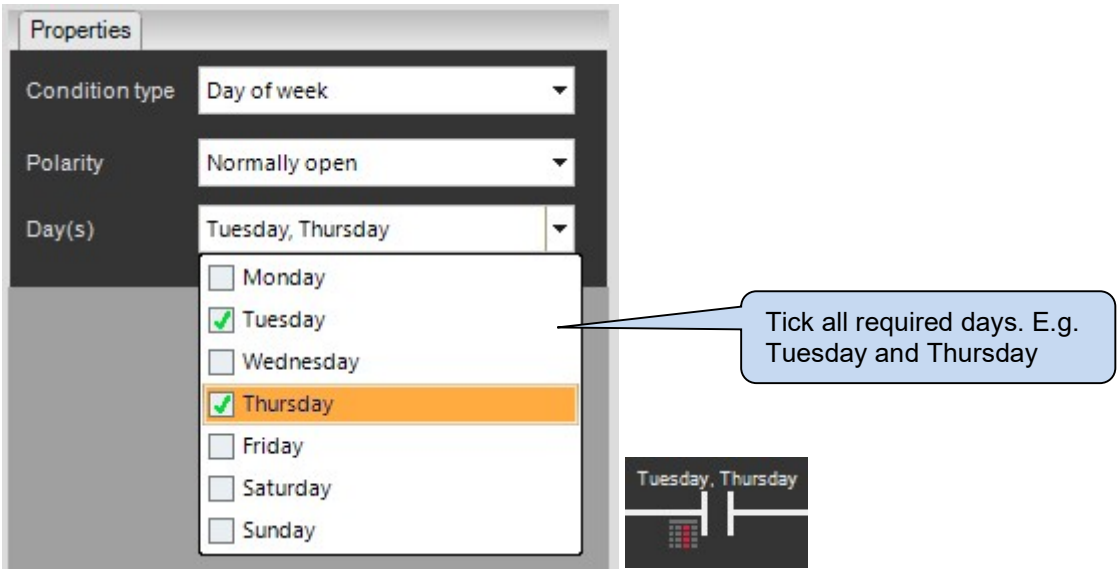
Drag the *Day of week* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows 2 days to be selected, Tuesday and Thursday. This works as an OR gate, the condition becomes true if the day of week is either Tuesday or Thursday.

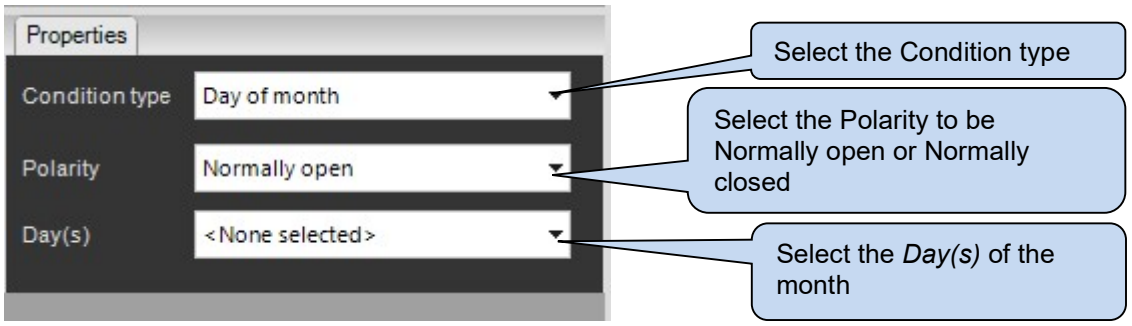


3.3.2.5 DAY OF MONTH

Allows the module's date to be tested and allows an action based upon a specific day or days.

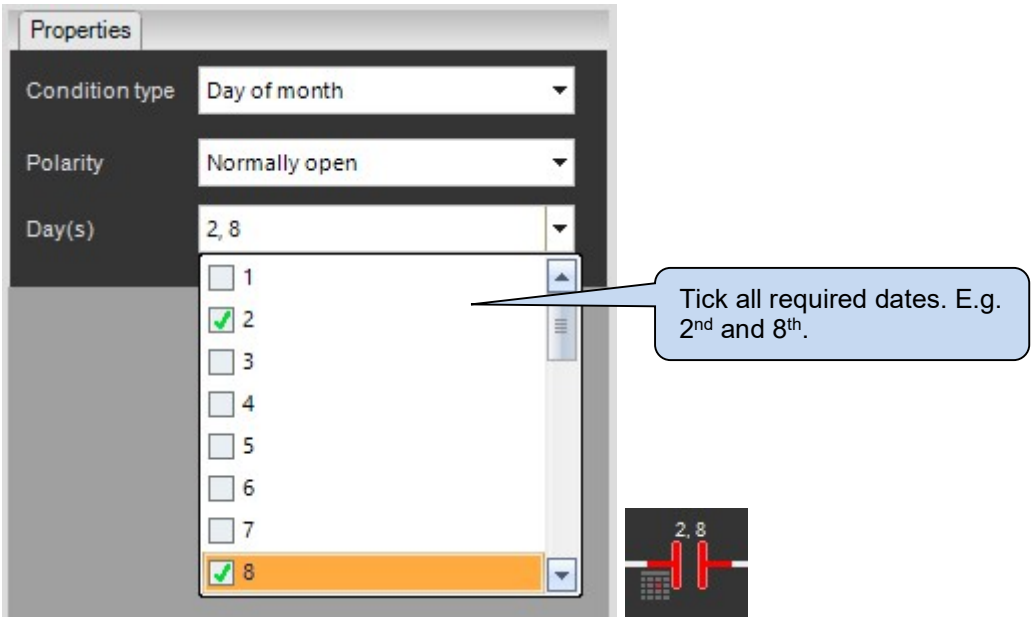
Drag the *Day of month* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows 2 dates to be selected, 2nd and 8th. This works as an OR gate, the condition becomes true if the date is either 2nd or 8th.

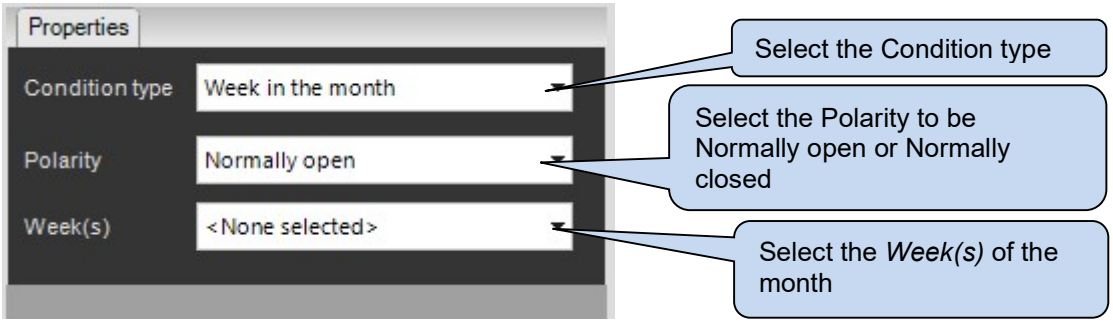


3.3.2.6 WEEK IN MONTH

Allows the module's week to be tested and allows an action based upon a specific week or weeks.

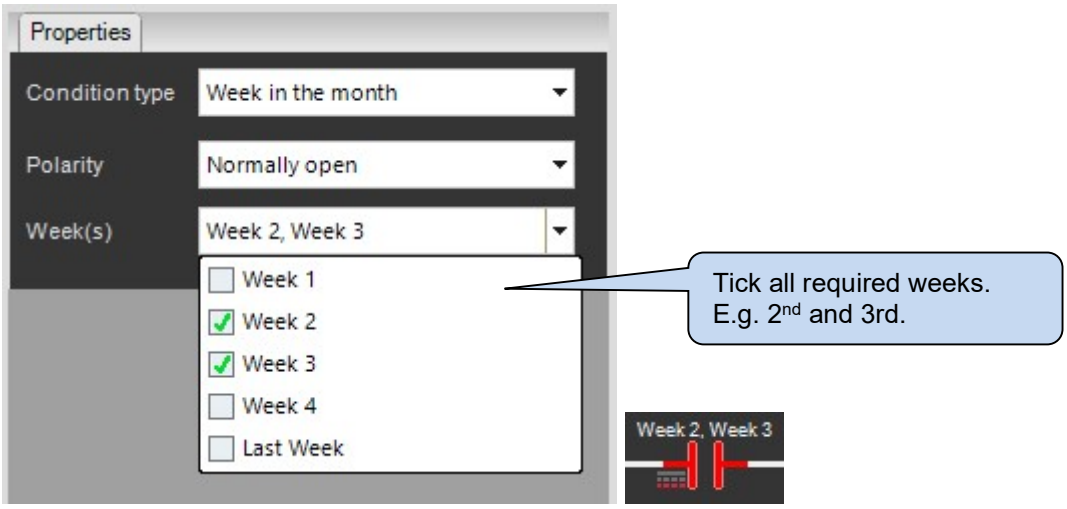
Drag the *Week in month* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows 2 weeks to be selected, 2nd week and 3rd week. This works as an OR gate, the condition becomes true if the week is either the 2nd week or 3rd week in the month.

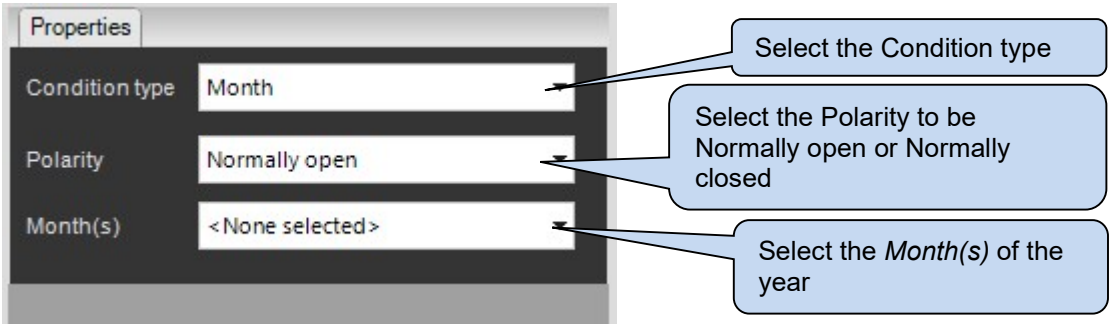


3.3.2.7 MONTH

Allows the module's month to be tested and allows an action based upon a specific month or months.

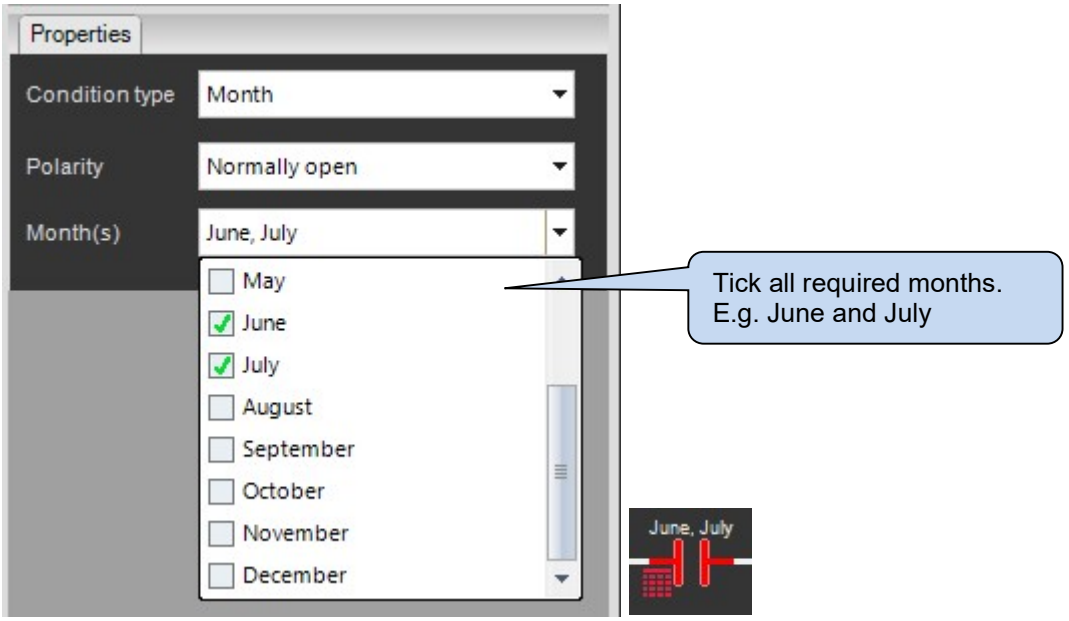
Drag the *Month* 

into the relevant place on the ladder rung and configure its properties.



Example

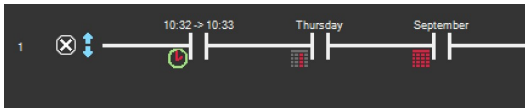
The below example shows 2 months to be selected, June and July. This works as an OR gate, the condition becomes true if the month is either the June or July.



It is also possible to combine two or more of these tests to make a more specific test.

For example:

+ + allows a test for a specific time, day, and month, for instance 10:32am on any Thursday in September.

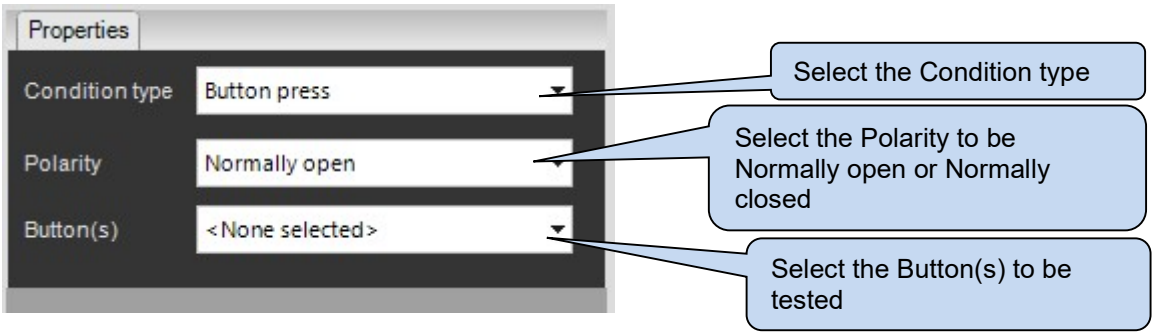


3.3.2.8 BUTTON TEST

Allows the PLC program to check if any of the control buttons are being pressed on the module.

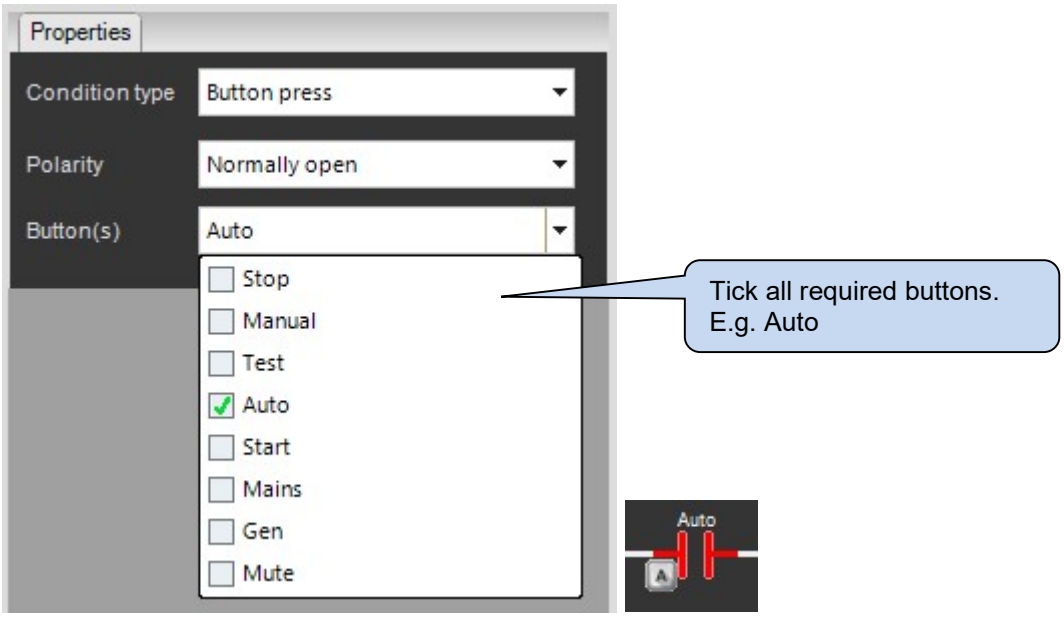
Drag the Button test 

into the relevant place on the ladder rung and configure its properties.



Example 1

The below example shows a Button test configured for Auto. The condition becomes true whilst the Auto button is pressed



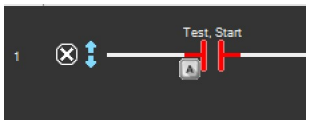
Depending upon module type this allows testing for the following button presses:

DSEGenset Range	DSEE400
Stop/Reset	Auto
Manual	Start
Auto	Stop
Test	Speed Lower
Start	Speed Raise
Mains	
Gen	
Mute	

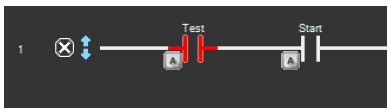
Example 2

Testing for multiple button presses.

This example is true when the *Test* button OR *Start* button is pressed.



This example is true when the *Test* button AND *Start* button are pressed together.

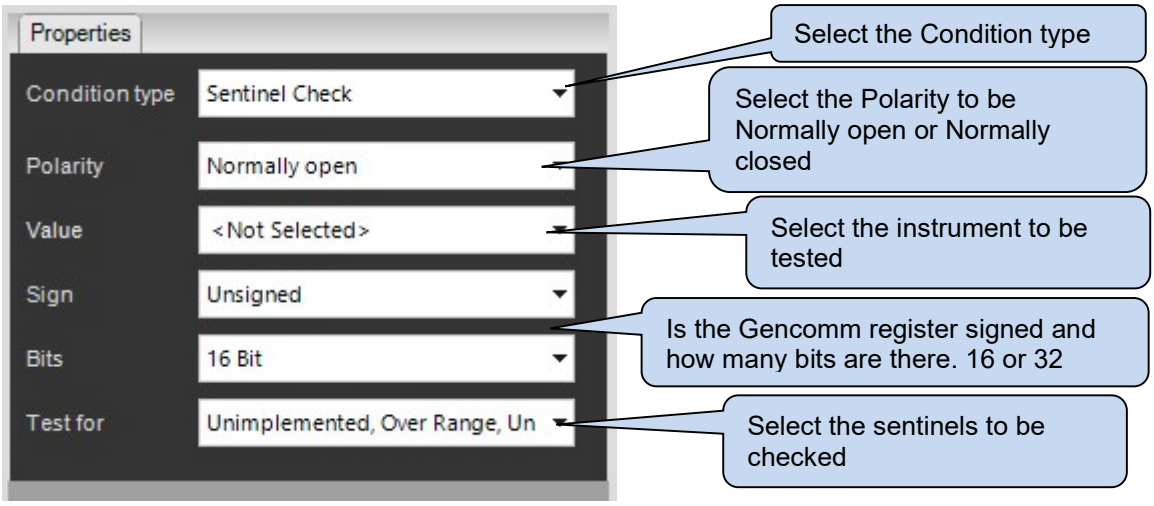


3.3.2.9 SENTINEL CHECK

Checks the selected instrument to see if its current state is a Sentinel Value. Sentinels are to show that the instrument is not available, or out of range.

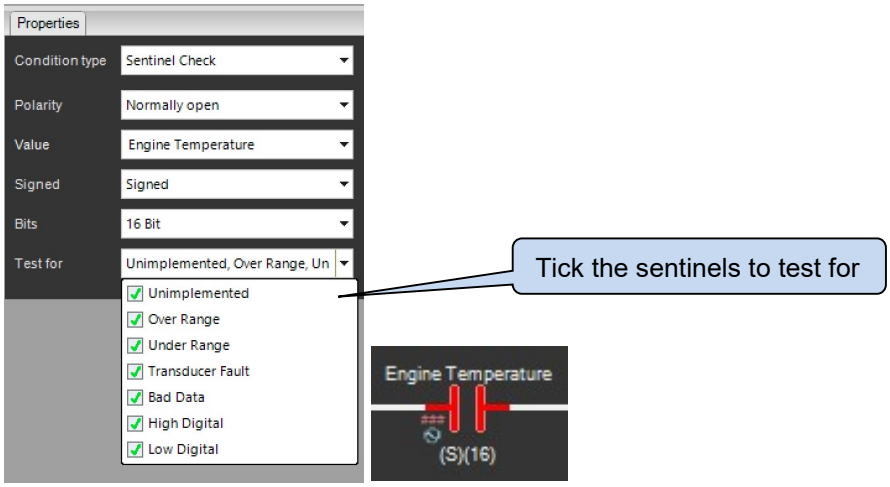
Drag the *Sentinel Check* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a Sentinel Check configured for engine temperature. The condition becomes true if the instrument returns a selected sentinel

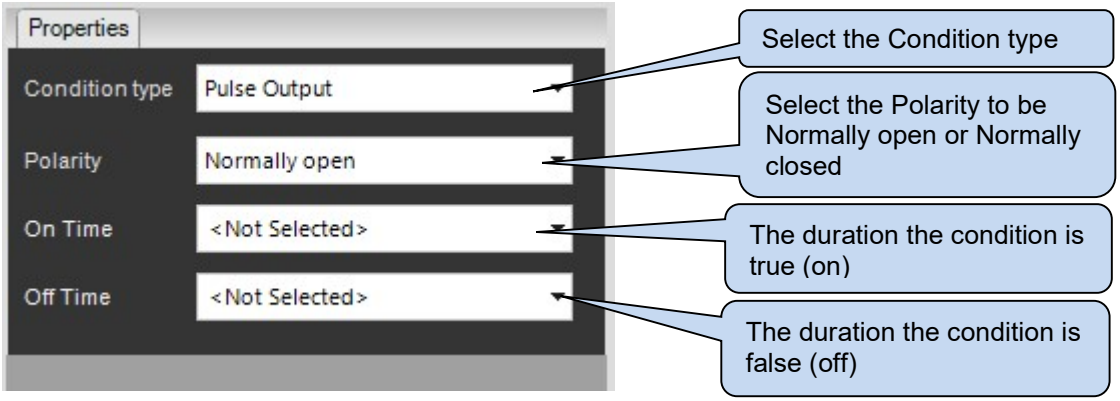


3.3.2.10 PULSE OUTPUT

Adds a Pulse function to the selected logic to flash between True and False at specified intervals.

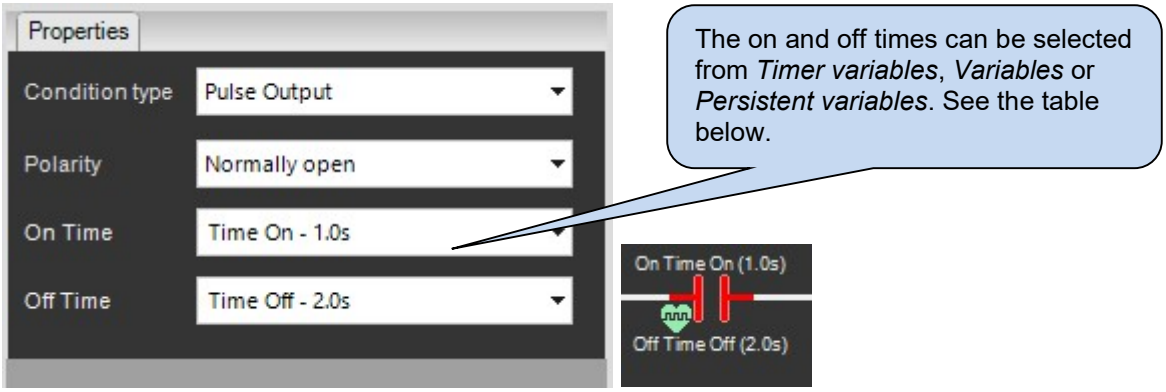
Drag the *Pulse Output* 

into the relevant place on the ladder rung and configure its properties.



Example

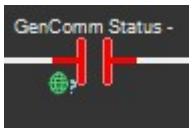
The below example shows a *Pulse Output* configured for an on time of 1 second and an off time of 2 seconds.



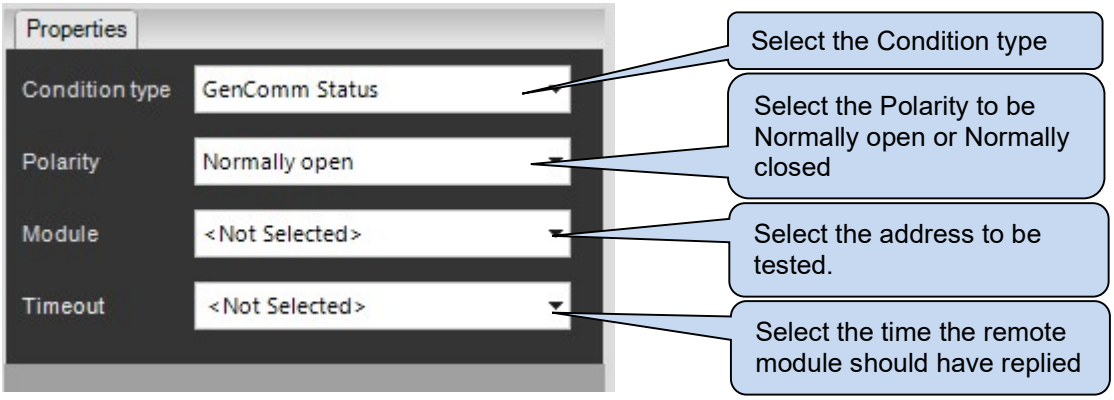
Selection	Description
Persistent Variables	This allows the pulse on and off settings to be edited from the modules display by setting them to watched.
Timer Variables	This is a fixed timer value, set in the variables window. To adjust the timer setting the PLC editor must be opened, select the variables icon, and adjust the timer. The config needs to be uploaded again.
Variables	This allows PLC programming to adjust the pulse on and off settings. An example of this would be to flash an LED at dual speeds, i.e. fast or slow depending on various conditions. The timer is in 0.1 seconds so a variable set for 10 would equal 1 second.

3.3.2.11 GENCOMM STATUS

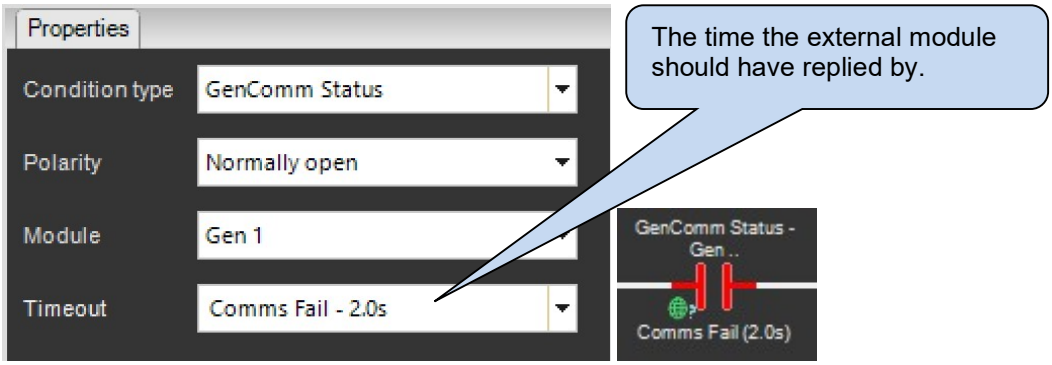
Checks that the Device selected from the System Device Address Book is responding to MODBUS queries.

Drag the *GenComm Status* 

into the relevant place on the ladder rung and configure its properties.



GenComm status properties box.



3.3.3 ACTIONS

The action toolbar contains the tools that allow the PLC to perform certain actions. These are described in the following sections.

Action	Description
Set coil	Sets the state of a Coil to True. The Coil remains True when the driving condition is removed, until a Reset or Toggle action changes it to False. This is useful for 'remembering' that a condition has occurred. A Coil is an internal state of the module.
Reset coil	Resets the state of a Coil to False. The Coil remains False when the driving condition is removed, until a Set or Toggle action changes it to True. This is useful for 'forgetting' that a condition has occurred.
Toggle coil	Toggles the state of a Coil upon the driving condition becoming True. If the Coil is currently True, it Resets it to False. If the Coil is currently False, it Sets it to True.
Drive coil	Drives the state of the Coil based on the driving condition. If the driving condition is True, the Coil is set to True. If the driving condition is False, the Coil is Reset to False.
Mathematical	PLC mathematics allows the user to manipulate instrumentation values with mathematical functions, placing values and results into the module's *Variables* or *Persistent Variables* for access later either by the PLC itself or via the module's display.
Copy	Allows a value to be copied into a Variable or Persistent Variable.
Clock adjust	Adjusts the internal clock by adding or subtracting one hour.
Virtual input	Drives the Virtual Input based on the driving condition. If the driving condition is True, the Virtual Input is activated. If the driving condition is False, the Virtual Input is deactivated. A Virtual Input provides the same functionality as Module Digital Input but is activated by the PLC program and does not require hard wiring.
Override GenComm	Allows the PLC program to write to the module's various MODBUS registers to change their value e.g. Power Output Requirement.
Reset alarm	This action allows individual alarms to be Reset. An alarm can only be reset if the condition that generated the alarm is no longer present.
Bit shift	Logically Shifts the specified value Left or Right by the specified number of Bits.
Value selector	Selects between one of two values based upon the driving condition and copies the value to the specified Variable or Persistent Variable.
Select string	Shows the selected Message on the module's display.
Set string	Shows the set Message on the module's display.
GenComm read	Reads the value of the specified GenComm register and copies the value to the specified Variable or Persistent Variable.
Rate of Change	If number of samples is greater than 1, that number of samples are taken of the input parameter and averaged. The rate of change of this average is then recorded in the output variable

3.3.3.1 TRIGGER TYPE

Some *Actions* are ‘edge triggered’. This means the action takes place when the preceding C*ondition(s)* change. If a *Condition* remains unchanged, the *Action* is not repeated.

Some *Actions* are ‘level triggered’. This means the action takes place if the C*ondition(s)* are true and continues to be actioned until the *Condition(s)* become false.

Action Type	Trigger Type
Set Coil	Edge
Drive Coil	Edge
Reset Coil	Edge
Toggle Coil	Edge
Increment Counter	Edge
Decrement Counter	Edge
Zero Counter	Edge
Mathematical	Level
Copy	Level
Clock Adjust	Edge
Timer	Level
Virtual Inputs	Level
GenComm Override	Level
Alarm Reset	Level

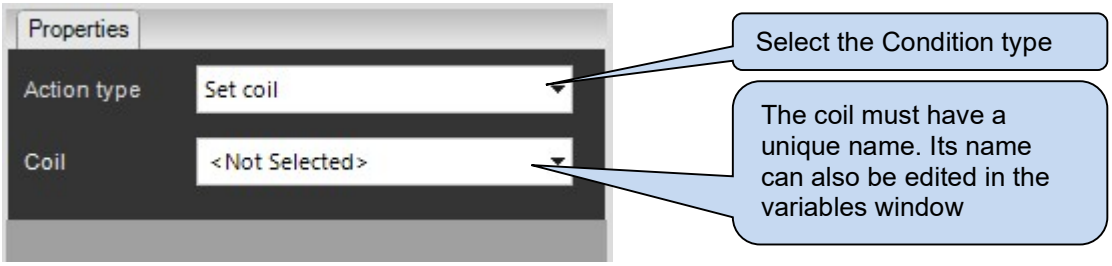
3.3.3.2 SET COIL

A Coil is an internal state of the module.

Set coil sets the state of a Coil to True. The Coil remains True when the driving condition is removed, until another Reset or Toggle action changes it to False. This is useful for 'remembering' that a condition has occurred.

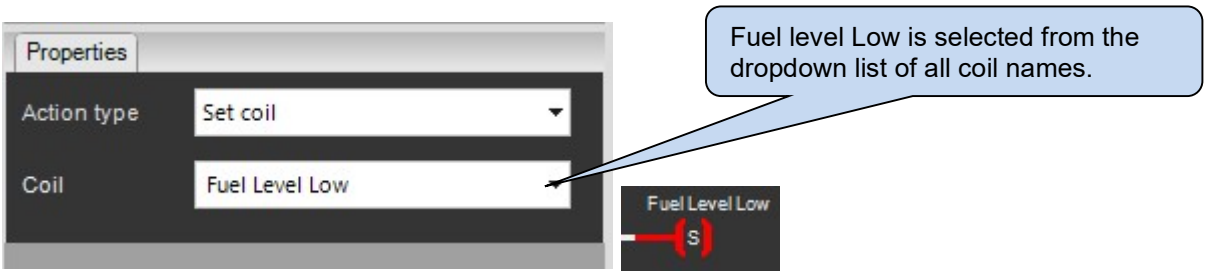
Drag the *Set Coil* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *Coil* configured for Fuel Level Low.



Example Set And Reset Coil Function

The *Fuel Level Low* coil is set when the fuel level falls below 10%, even if it is only for a fraction of a second.

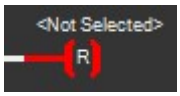
The *Fuel Level Low* coil is reset when the fuel level rises above 80%, even if it is for only a fraction of a second.

A digital output can then be configured for Fuel Level Low coil. This output is then follow the Fuel Level Low coil.

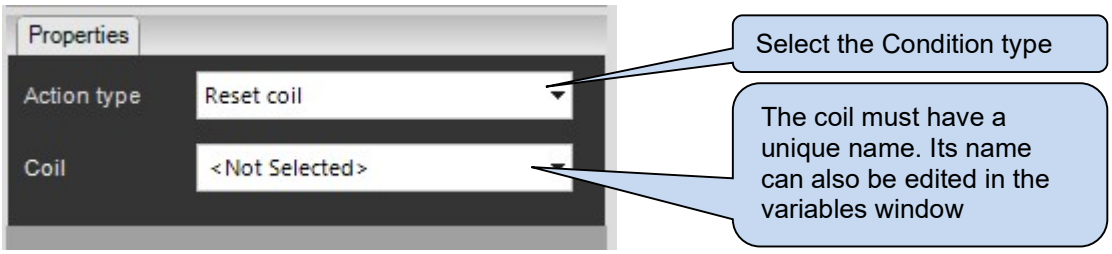
3.3.3.3 RESET COIL

A Coil is an internal state of the module.

Resets the state of a Coil to False. The Coil remains False when the driving condition is removed, until another Set or Toggle action changes it to True. This is useful for 'forgetting' that a condition has occurred.

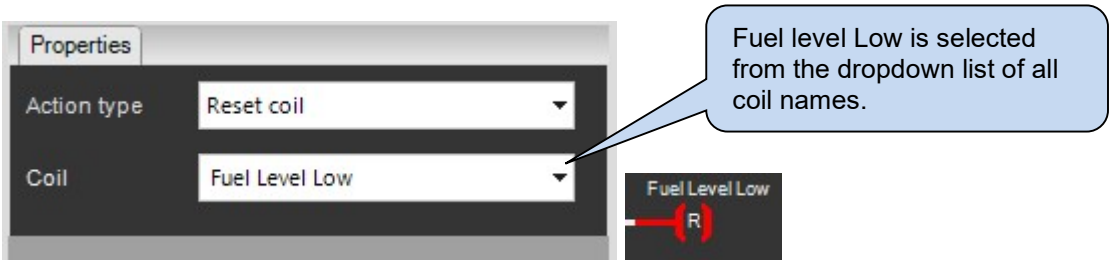
Drag the *Reset Coil* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *Coil* configured for Fuel Level Low.



Example Set And Reset Coil Function

The *Fuel Level Low* coil is set when the fuel level falls below 10%, even if it is only for a fraction of a second.

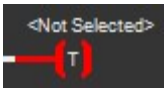
The *Fuel Level Low* coil is reset when the fuel level rises above 80%, even if it is for only a fraction of a second.

A digital output can then be configured for Fuel Level Low coil. This output follows the Fuel Level Low coil.

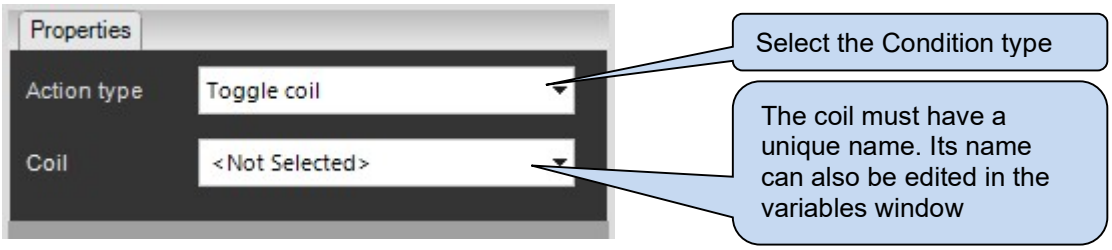
3.3.3.4 TOGGLE COIL

A Coil is an internal state of the module.

Toggles the state of a Coil upon the driving condition becoming True. If the Coil is currently True, it Resets it to False. If the Coil is currently False, it Sets it to True.

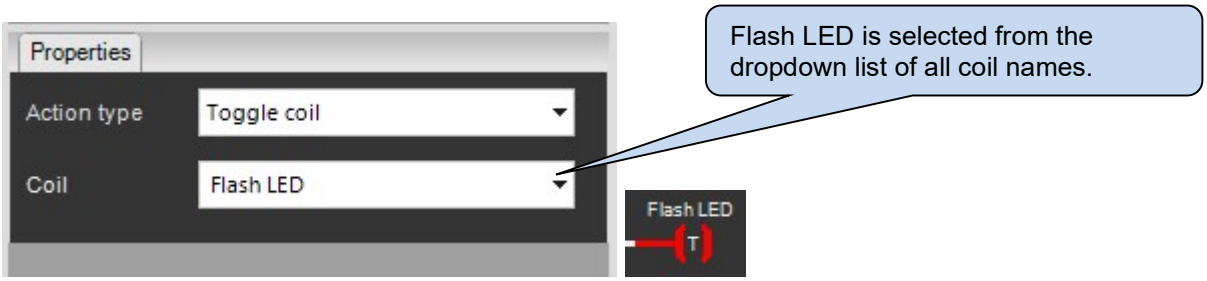
Drag the *Toggle Coil* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *Coil* configured for Flash LED.

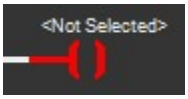


Digital input A could be a push button and each press of the button would switch the Toggle PB on and off.

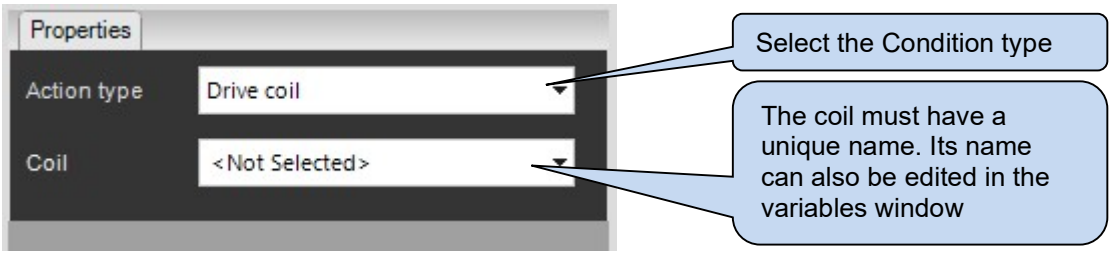
3.3.3.5 DRIVE COIL

A Coil is an internal state of the module.

Drives the state of the Coil based on the driving condition. If the driving condition is True, the Coil is set to True. If the driving condition is False, the Coil is Reset to False.

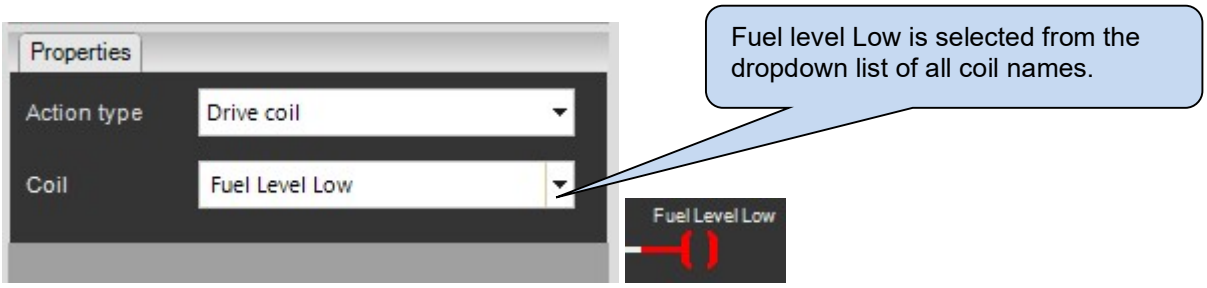
Drag the *Drive Coil* 

into the relevant place on the ladder rung and configure its properties.

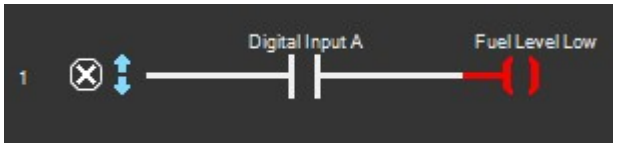


Example

The below example shows a *Coil* configured for Fuel Level Low.



Digital input A could be a fuel level low float switch. When the digital input A actives the Fuel Level Low Drive Coil is activated. When the digital input A deactivates the Fuel Level Low Drive Coil is deactivated.



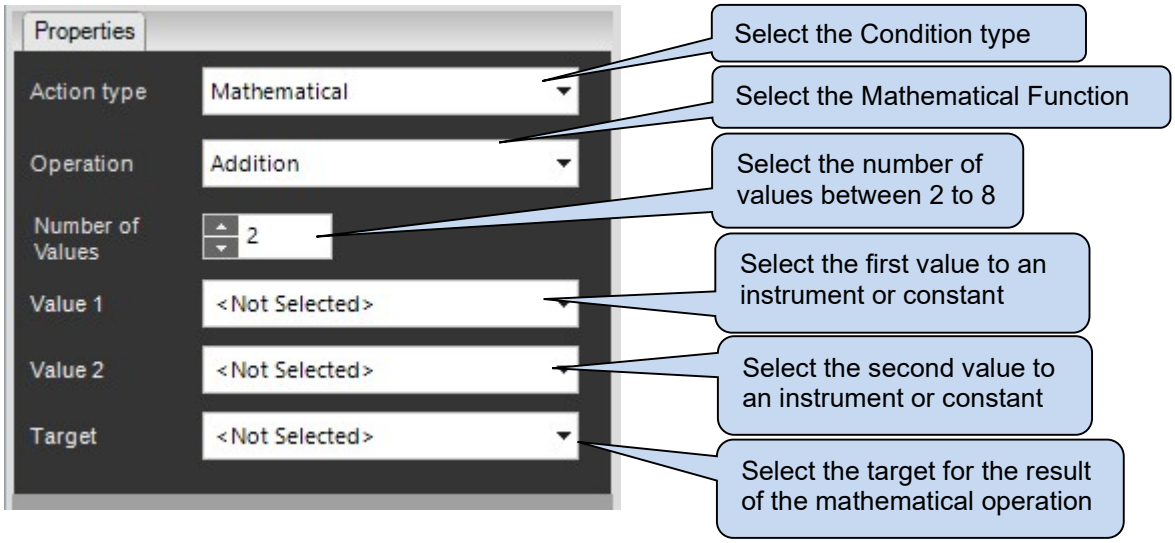
3.3.3.6 MATHEMATICAL

Actions to allow mathematical functions to be performed and the results placed in a *Variable* or *Persistent Variable*.

Item	Description
Operation Value 1 Value 2 Up to Value 8	***Addition***: *Value1 + Value 2 up to Value 8* ***Subtraction***: *Value 1 – Value 2 up to Value 8* ***Multiply***: *Value 1 * Value 2 up to Value 8* ***Divide***: *Value 1* / *Value 2* up to Value 8(remainder is lost) ***Remainder***: The remainder of *Value 1* / *Value 2 up to Value 8* ***Magnitude***: The value with its 'sign' removed. For example, both 6 and -6 give a value of 6. ***Minimum***: The lowest of *Value 1* and *Value 2 up to Value 8*. For example, *Value 1* = 5, *Value 2* = 7. The *Minimum* is 5. ***Maximum***: The highest of *Value 1* and *Value 2 up to Value 8*. For example, *Value 1*=5, *Value 2* = 7. The *Maximum* is 7. ***Average***: The average of all the values entered, For example, Value 1 = 6, Value 2 = 4. The *average* is 5 (remainder is lost) ***Range***: The difference between the maximum and minimum of the entered values. For example, *Value 1* = 5, *Value 2* = 7. The *Range* is 2 **Bitwise AND:** Described separately **Bitwise OR:** Described separately
Target	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

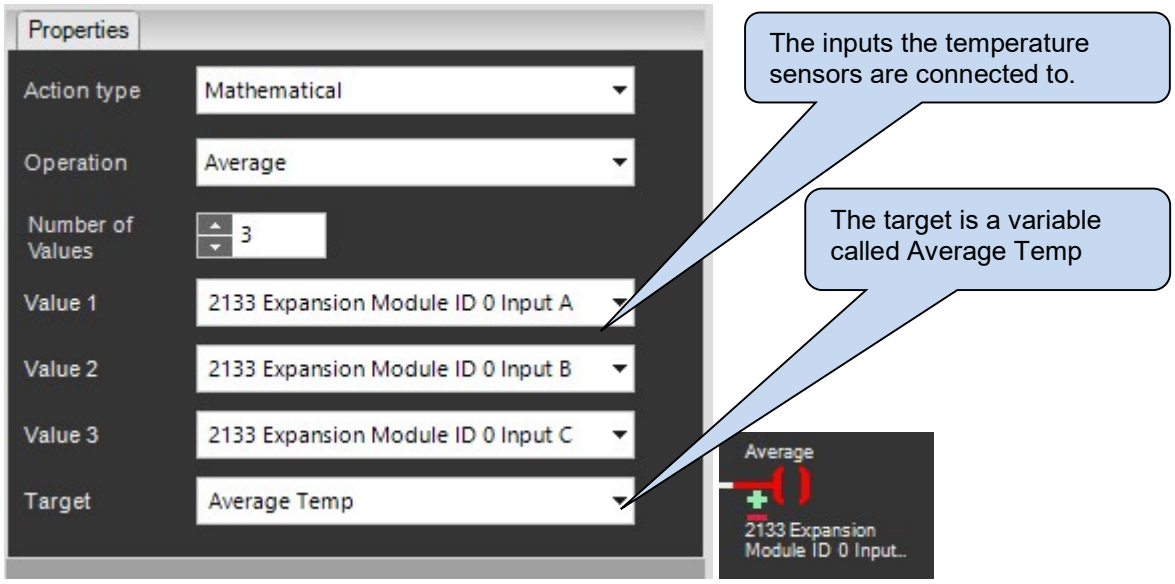
Drag the *Mathematical* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *mathematical* operation to find the Average of 3 alternator winding temperatures that have been connected to a 2133 expansion module.

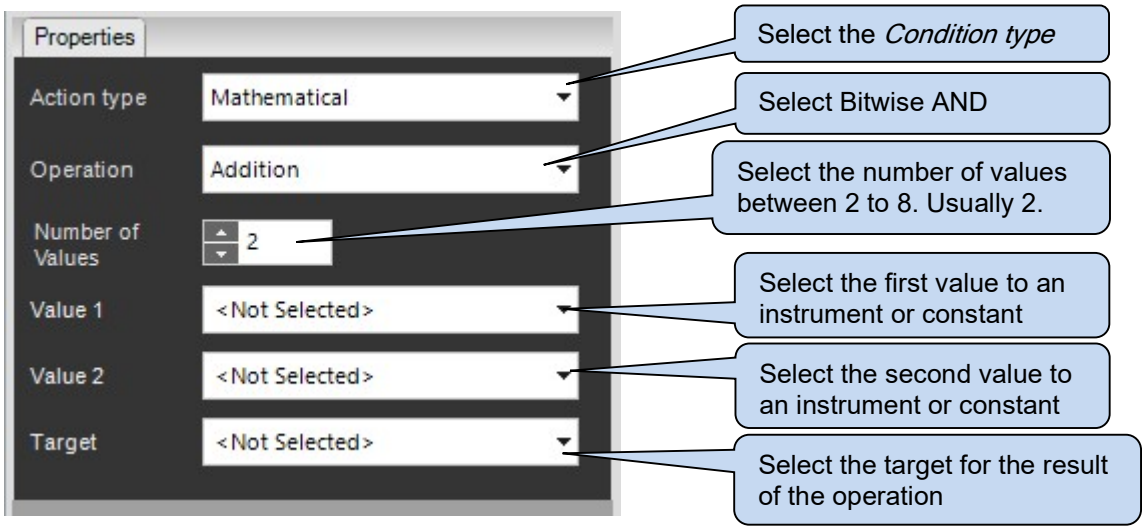


3.3.3.6.1 BITWISE AND

Bitwise AND, working in conjunction with *Bit Shift* is used to extract a certain number of bits from a 32-bit register. This is ideal for decoding a custom message that is transmitted on the MSC link.

Drag the *Mathematical* 

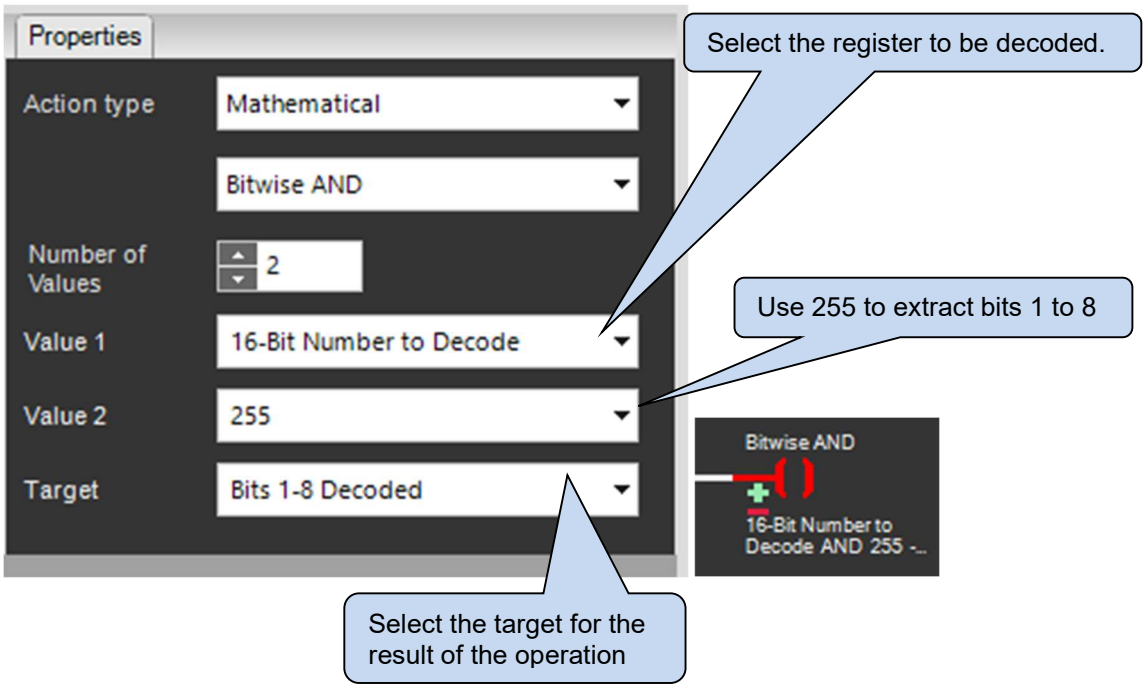
into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *Bitwise AND* operation to extract the first 8-bits of a 16-bit number.

> **NOTE: To extract the second 8-bits, *Value 2* must be 65280. Once the second 8-bits have been extracted, *Bit Shift* must be used to shift the bits 8 places to the right to attain a correct decimal representation.**

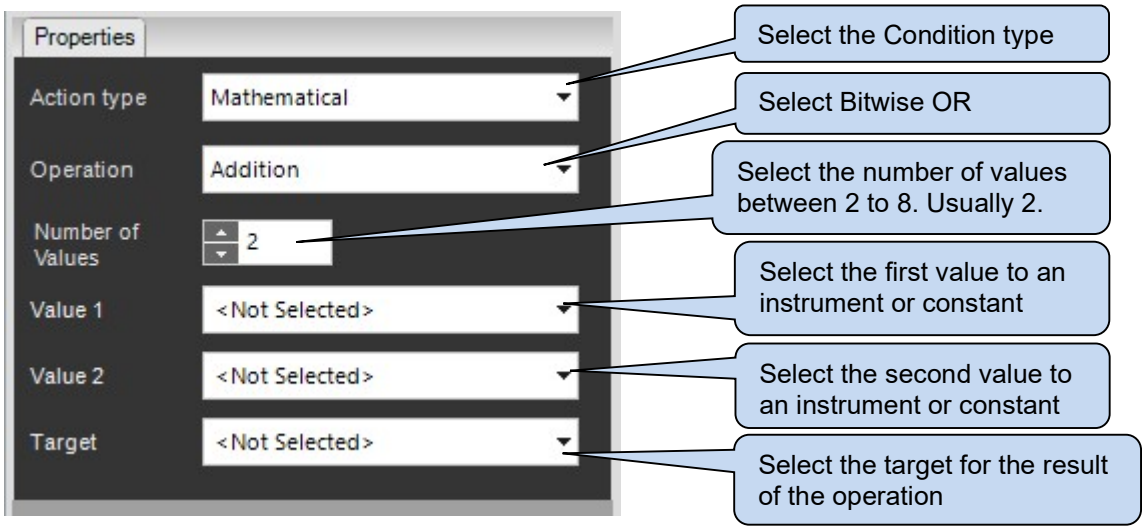


3.3.3.6.2 BITWISE OR

Bitwise OR, working in conjunction with *Bit Shift* is used to combine multiple bits into one 32-bit message. This is ideal for coding a custom message that is transmitted on the MSC link.

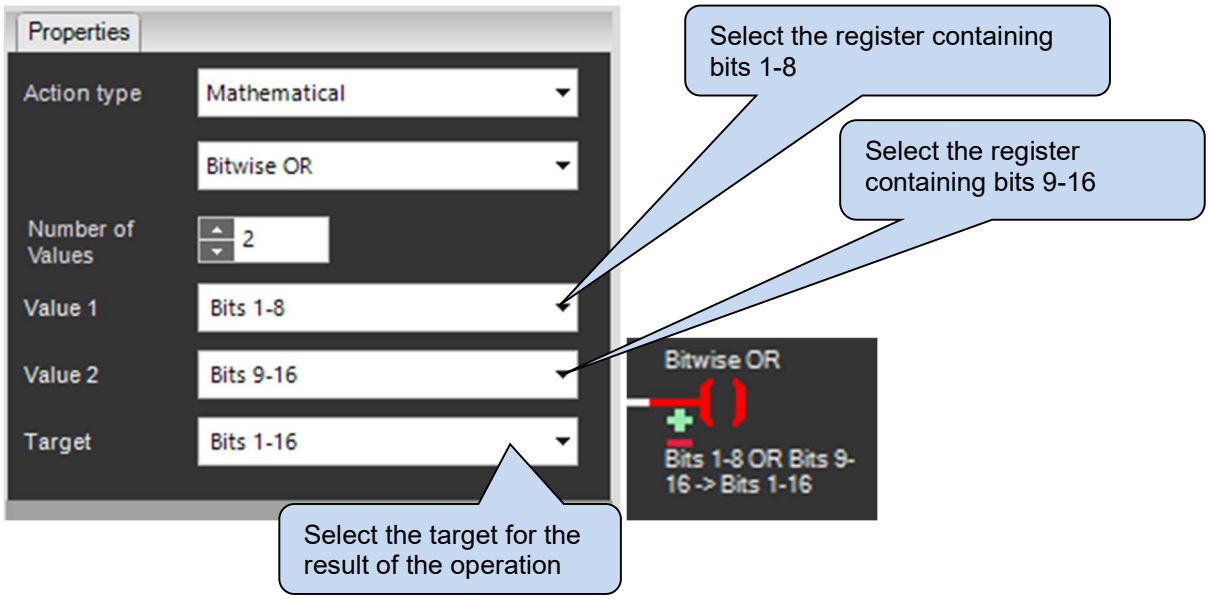
Drag the *Mathematical* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows a *Bitwise OR* operation to combine 2, 8-bit values into a single 16-bit number.

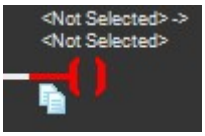


Example Combining Two 8-Bit Numbers Into One 16-Bit Message

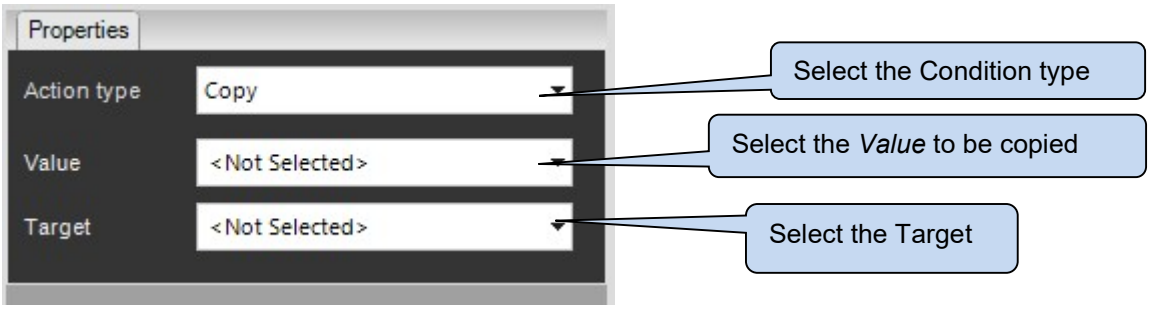
Engine Temperature 81°C	Binary 0101 0001 0000 0000
Fuel Level 26 %	Binary 0000 0000 0001 1010
Combined	Binary 0101 0001 0001 1010

3.3.3.7 COPY

Allows a value to be copied or placed into a *Variable* or *Persistent Variable*.

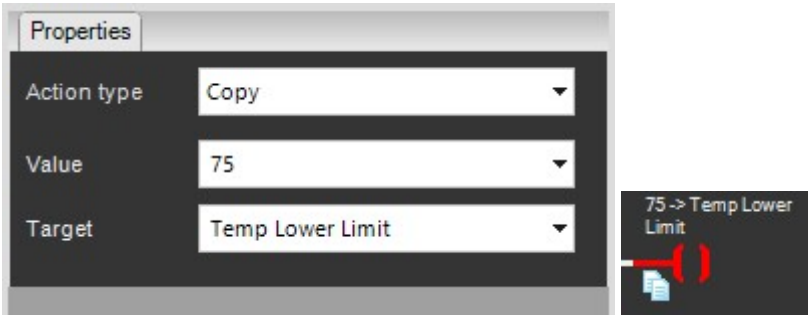
Drag the *Copy* 

into the relevant place on the ladder rung and configure its properties.



Example 1

The below example shows a *copy* operation that copies 75 into the target Temp Lower Limit.



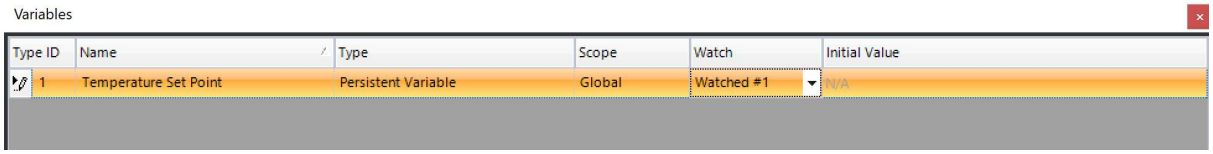
Item	Description
Value	***Fixed Value***: Enter the value manually. ***Calculated Value***: Select the calculated instrumentation value from a list of pre-defined options. ***Instrumentation Value***: Select one of the modules instrumentation items. ***Variable***: Select one of the *Variables*. ***Persistent Variables***: Select one of the *Persistent Variables*.
Target	The location where the Value is to be placed. ***Variable***: Values placed in the *Variables* are lost when the module DC power is removed. ***Persistent Variables****:* Values placed in the *Persistent Variables* are maintained, even when the module DC power is removed. For further details of *Variables* and *Persistent Variables*, see section 3.1.6.1 in this document.

Example 2

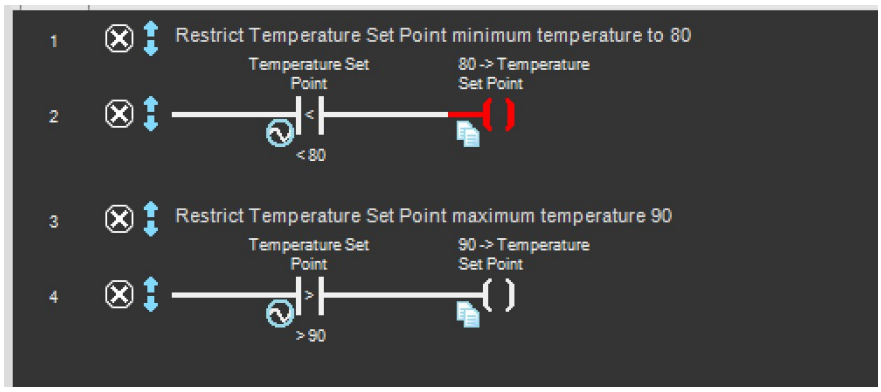
Any items that are watched automatically appear on the modules display. It is also possible to adjust watched *Persistent Variables* from the modules display. This is ideal for user defined set points, e.g. engine temperature, however it is good practice to restrict the range of adjustment.

Restricting the range is easily achieved using the copy function

Configure a watched *Persistent Variable*



Use PLC logic to restrict its range e.g. around 85°C

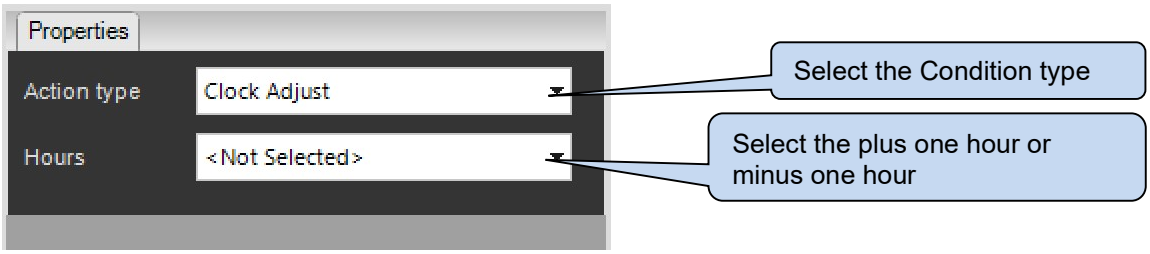


3.3.3.8 CLOCK ADJUST

This allows one hour to be added or subtracted from the module's internal clock. This enables a user defined function to be created that adjust the internal clock for Daylight Saving adjustments

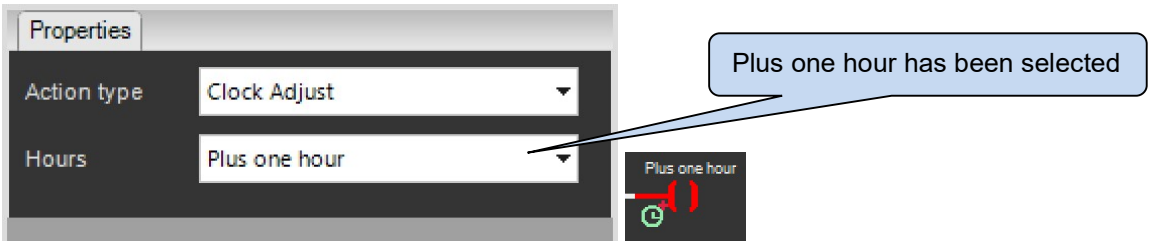
Drag the *Clock Adjust* 

into the relevant place on the ladder rung and configure its properties.

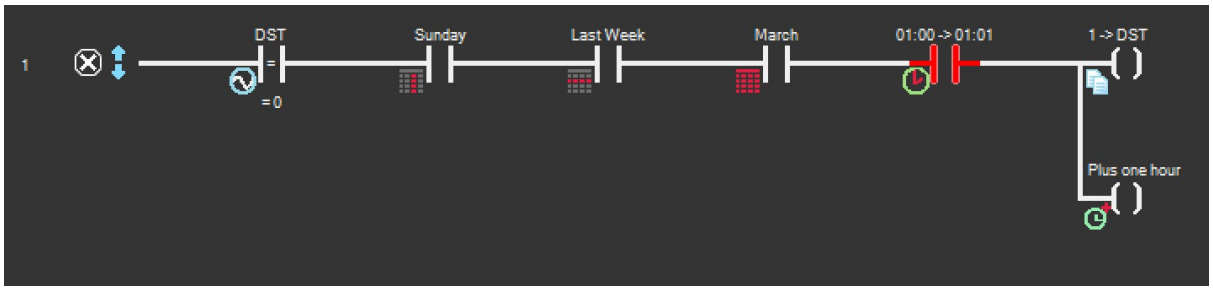


Example

The below example shows a *clock adjustment of plus one hour*.



Increasing the modules time one hour on the last Sunday in March at 1 am. Additionally this sets a user Persistent Variable (DST) to the value of 1. This allows us to check in the PLC SCADA or in the module instrumentation if DST is active:

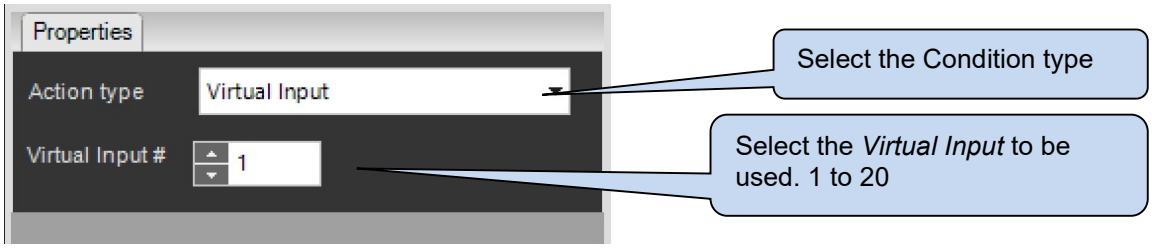


Decreasing the modules time one hour on the last Sunday in October at 2 am. We additionally check the status of DST (*Persistent Variable*) to ensure the clock is not continually adjusted one hour later!

3.3.3.9 VIRTUAL INPUT

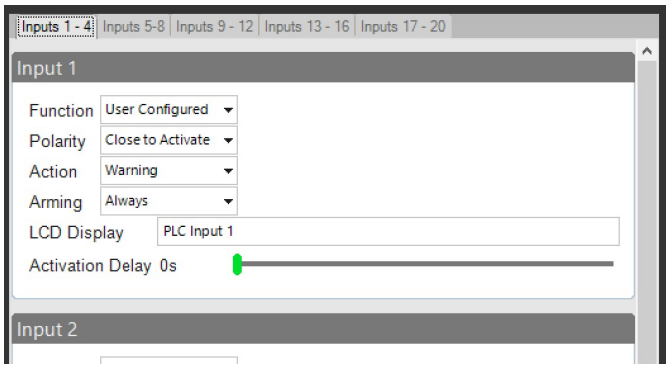
Drag the *Virtual Input* 

into the relevant place on the ladder rung and configure its properties.



Configure The Input

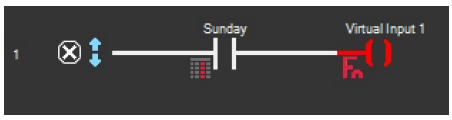
Virtual Inputs are configured in exactly the same way and has the same selections as a modules Digital Inputs. The difference is that the PLC Function is activated by the PLC and does not require hard wiring.

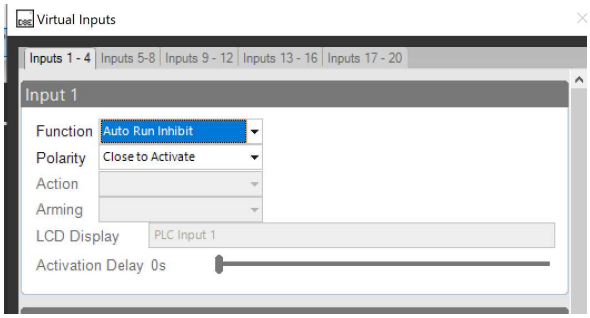


Item	Description
Function	***User Configured***: Allows the user to configure the *Function* to perform an alarm or status indication. ***Digital Input Selection List***: Allows the user to select from a predefined selection list. Refer to the DSE Configuration Suite PC Software Manual for the host module in use for a full description of possible selections.
Polarity	***Close to Activate***: The Function is 'normally inactive' and must be driven in the PLC to activate it. ***Open to Activate***: The Function is 'normally active' and must be driven in the PLC to de-activate it.
Action (Only applicable when function is set to "*User Configured*")	***Electrical Trip***: When activated, an electrical trip alarm is generated, the load switch is opened (if closed) and the generator placed into the cooling run before stopping. **Indication**: No alarm condition is generated, and the set continues to run. This is often used to create status indications or be monitored by the user's PLC logic. ***Warning***: When activated, a warning alarm is generated but the set remains running. ***Shutdown***: When activated, a shutdown alarm is generated, the load switch (if closed) is immediately opened, and the set is immediately stopped.
Arming	Selects when the alarm can be activated. Active from mains parallel Always From safety on From starting Never
LCD Display	The text that is displayed on the modules LCD when the input activates and generates an alarm
Activation Delay	This is used to give a delay on acceptance of the virtual input.

Example

Using a function to prevent running the generator upon a mains failure during the whole of Sunday. This uses Virtual Input 1, configured to "Auto Run Inhibit".



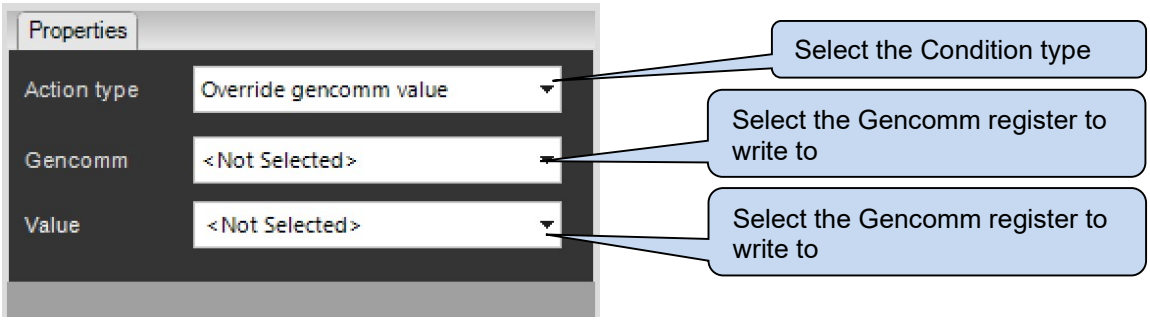


3.3.3.10 OVERRIDE GENCOMM

Allows the PLC program to write to the module's various MODBUS registers to change their value e.g. Power Output Requirement.

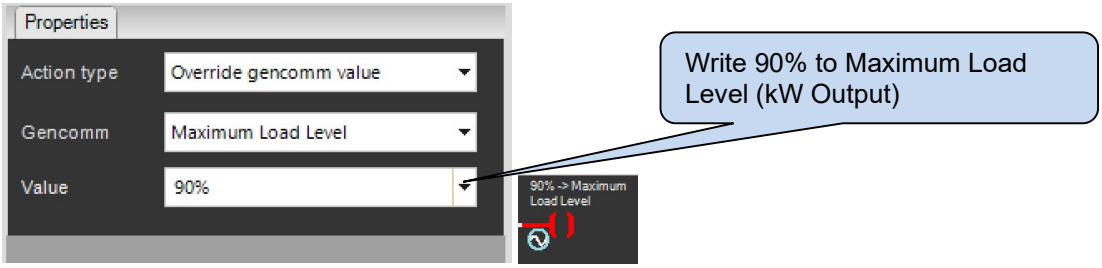
Drag the *Override GenComm* 

into the relevant place on the ladder rung and configure its properties.

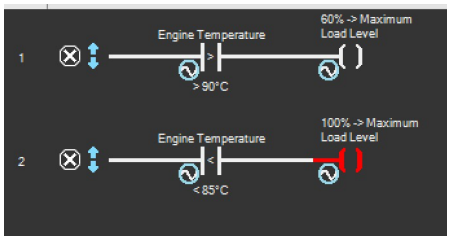


Example 1

The below example shows a GenComm register, Maximum Load Level (kW Output when running in parallel with the mains (Grid/Utility), set *for 90%*

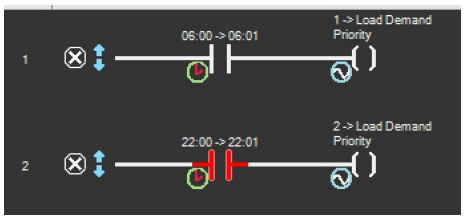


Using Override GenComm, the amount of power the generator produces when in parallel with the mains (Grid/Utility) can be changed.



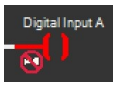
Example 2

Using Override GenComm, the *Run Priority* of the set can be changed.

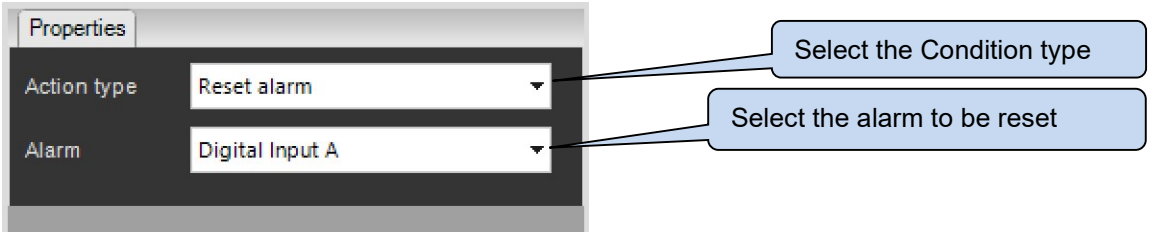


3.3.3.11 RESET ALARM

This action allows individual alarms to be reset. An alarm can only be reset if the condition that generated the alarm is no longer present.

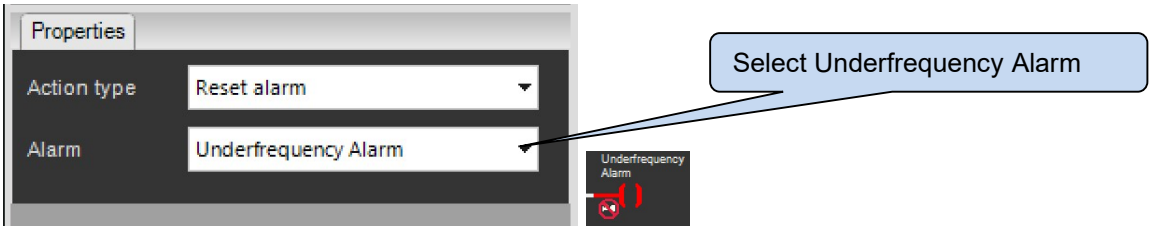
Drag the *Reset Alarm* 

into the relevant place on the ladder rung and configure its properties.



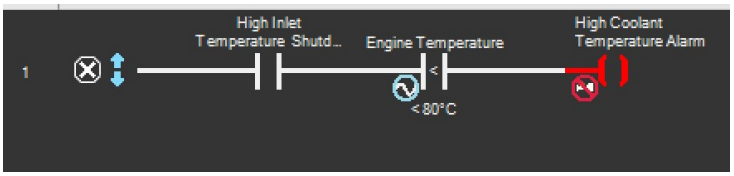
Example 1

The below example shows the resetting of under frequency alarm



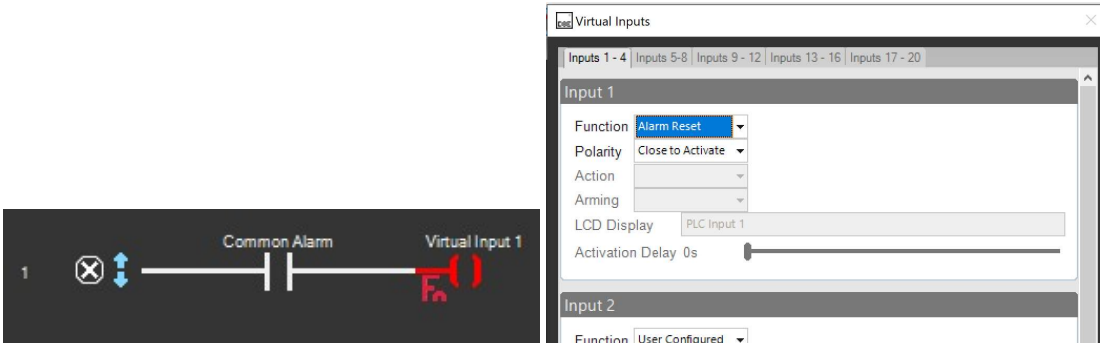
Example 2

Resetting an over temperature alarm once the engine has cooled down.



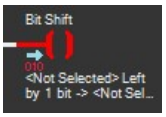
Resetting All Alarms

To perform an action that resets ALL alarms, it is more appropriate to drive a PLC Function that has been configured to *Alarm Reset*.

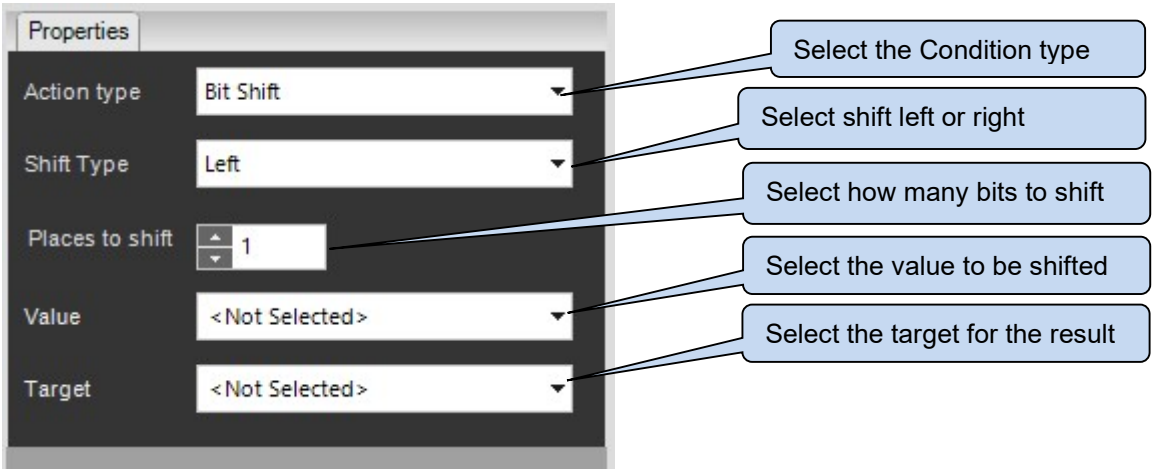


3.3.3.12 BIT SHIFT

Logically Shifts the specified value Left or Right by the specified number of Bits. The result is a 32bit number.

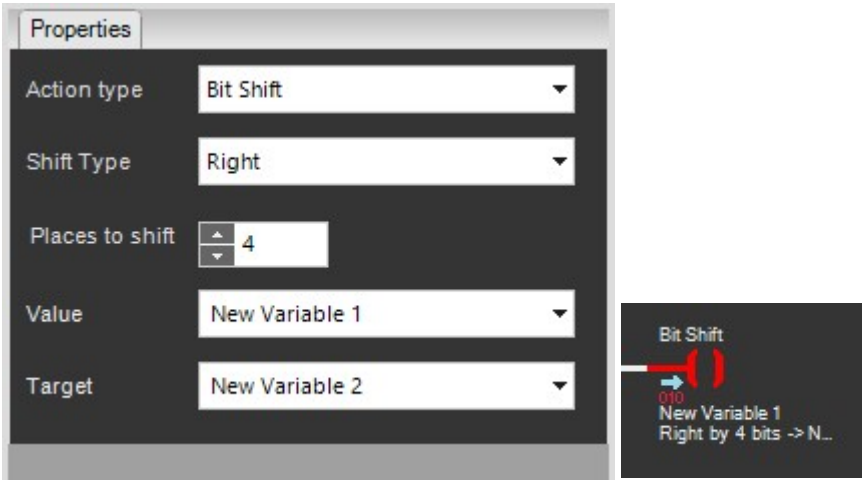
Drag the *Bit Shift* 

into the relevant place on the ladder rung and configure its properties.



Example

The below example shows the bit shift of one variable into another.



Assume that user defined New Variable 1 contains the value 2,921, which converts into binary number as 0000 1011 0110 1001.

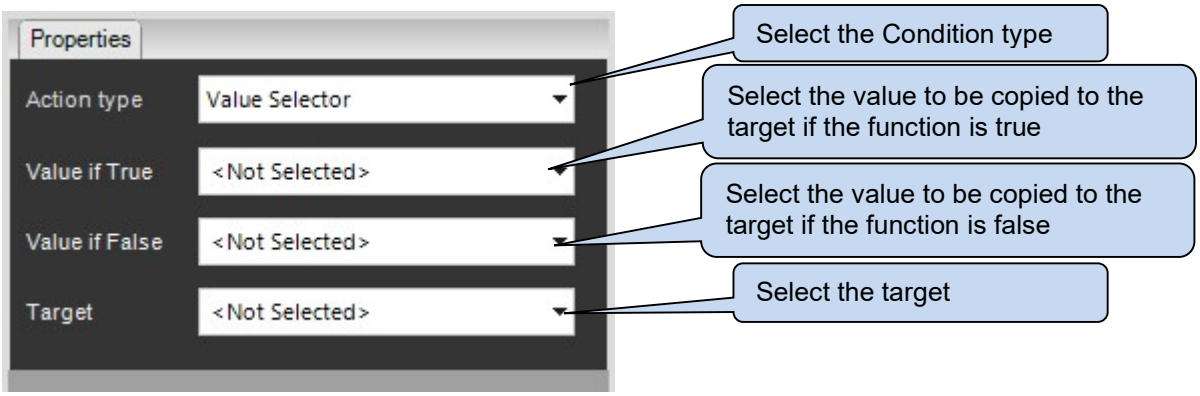
The *Bit Shift* (in the example above) would shift the binary number 4 bits to the right and converts it to a 32-bit number (if it is a 16-bit number). So, New Variable 2 would contain
0000 0000 0000 0000 0000 0000 1011 0110

3.3.3.13 VALUE SELECTOR

Selects between one of two values based upon the driving condition and copies the value to the specified Variable or Persistent Variable.

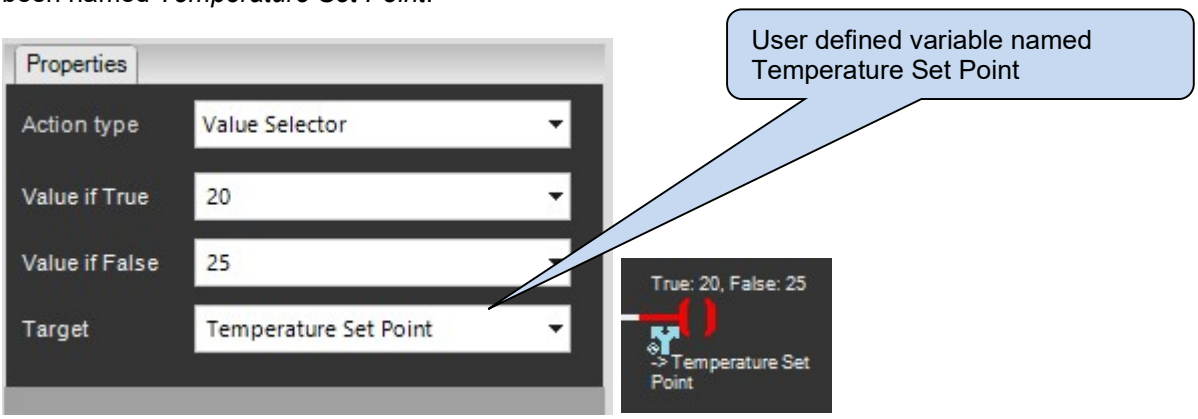
Drag the *Value Selector* 

into the relevant place on the ladder rung and configure its properties.

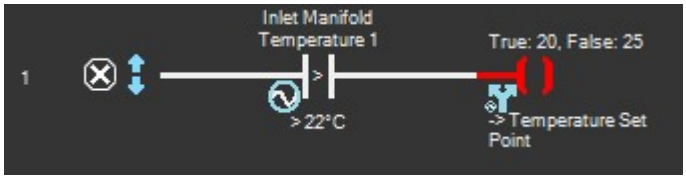


Example

The below example shows a *Value Selector* that copies a value into a user defined variable that has been named *Temperature Set Point*.



In this example a fixed value of 20 is copied into the user variable *Temperature Set Point* if the *Inlet Manifold Temperature 1* is greater than 22°C. If the *Inlet Manifold Temperature 1 is below 22°C, then 25 is* copied into the user variable *Temperature Set Point*

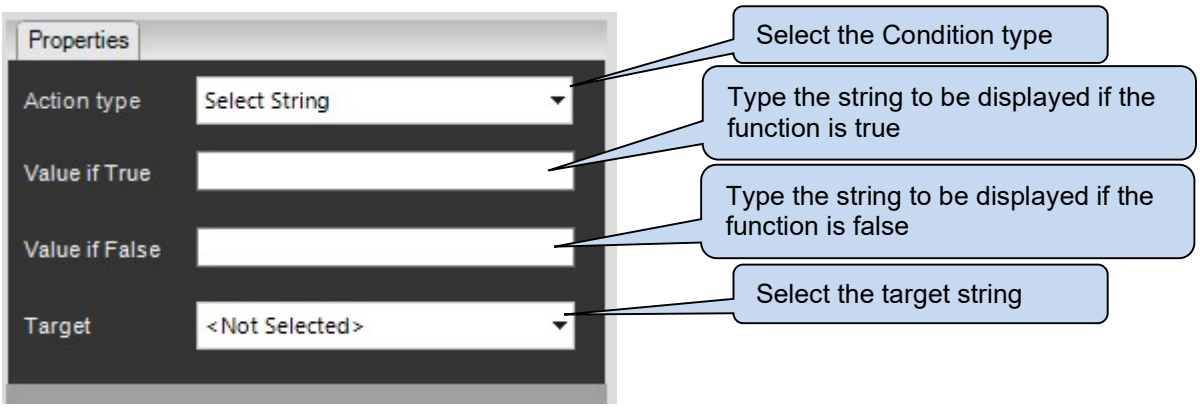


3.3.3.14 SELECT STRING

Shows the selected Message on the module's display.

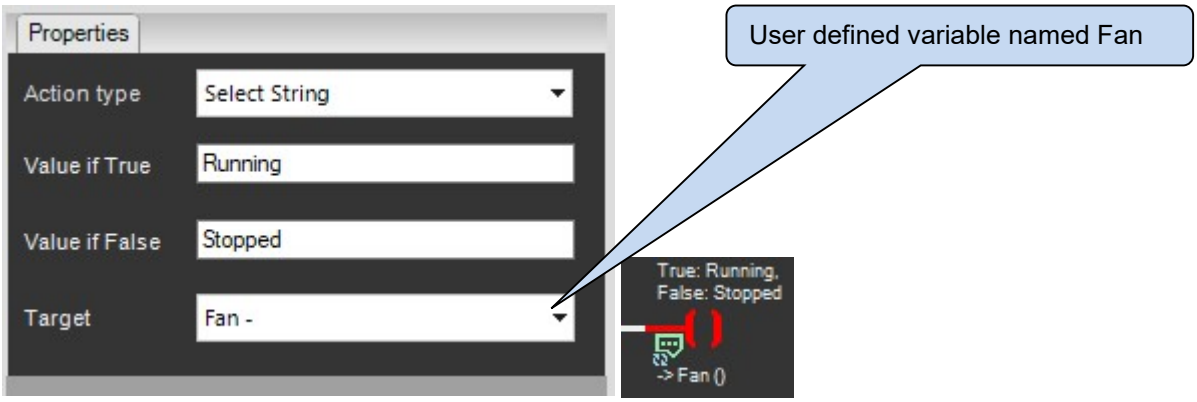
Drag the *Select String* 

into the relevant place on the ladder rung and configure its properties.

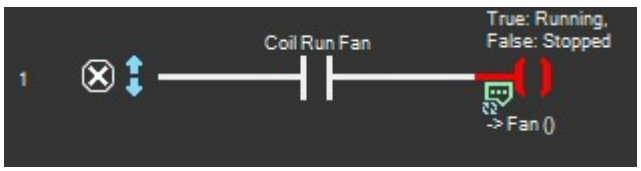


Example

The below example shows a *Select String* that copies a string into a user defined variable that has been named *Fan*.



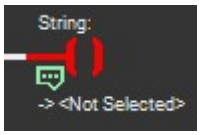
In this example the string Running is copied into the user variable *Fan* if the *Coil Run Fan* is True. Stopped is copied into the user variable *Fan* if the *Coil Run Fan* is False



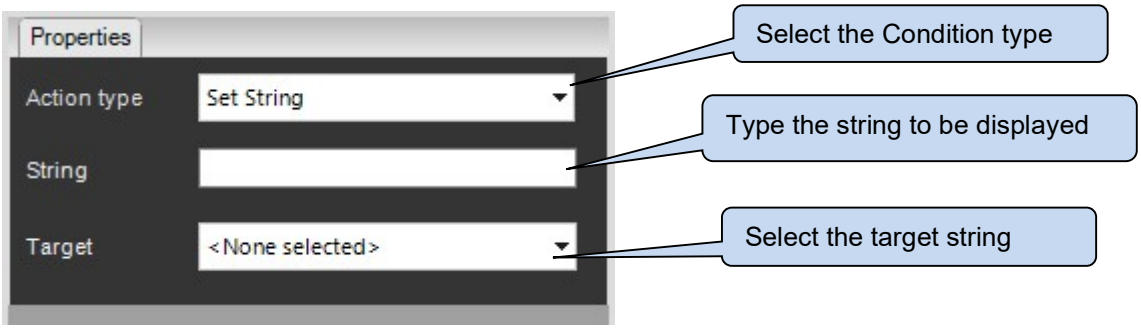
If the user variable *Fan* is watched then a display page automatically appears on the modules display. The instrument is called Fan and shows either Stopped or Running.

3.3.3.15 SET STRING

Shows the set Message on the module's display.

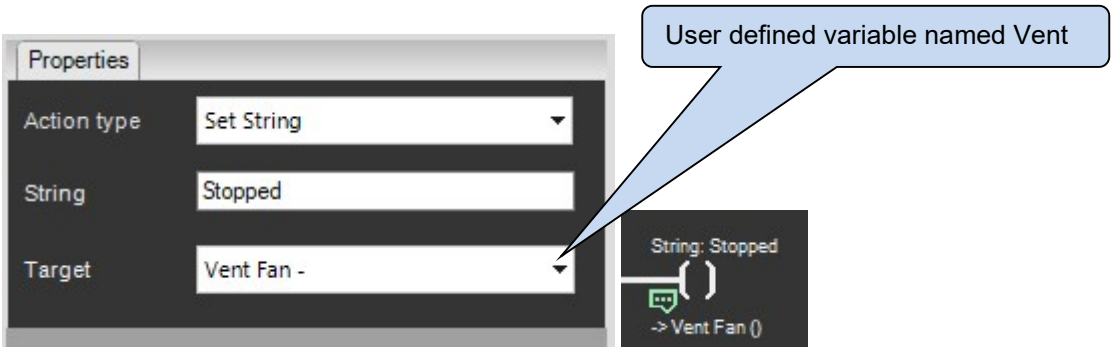
Drag the *Set String* 

into the relevant place on the ladder rung and configure its properties.



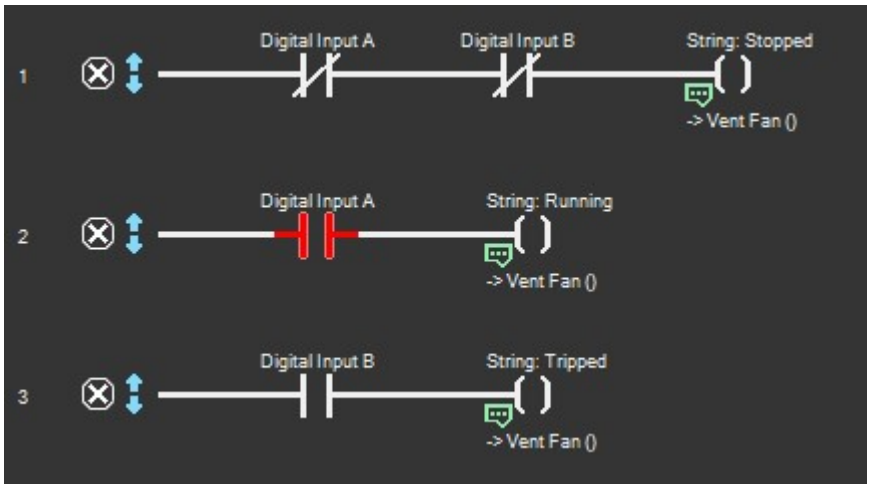
Example

The below example shows a *Set String* that copies the string *Stopped* into a user defined variable that has been named *Vent Fan*.



Set string can be used to create a state machine

In this example the string *Vent Fan* is switched between messages *Stopped*, *Running* and *Tripped* depending on the state of digital input A and digital input B



If the user variable *Vent Fan* is watched then a display page automatically appears on the modules display. The instrument is called *Vent Fan* and shows either Stopped, Running or Tripped.

If input A and input B are activated at the same time, the string *Vent Fan* displays Tripped, simply because it is the last one to be activated.

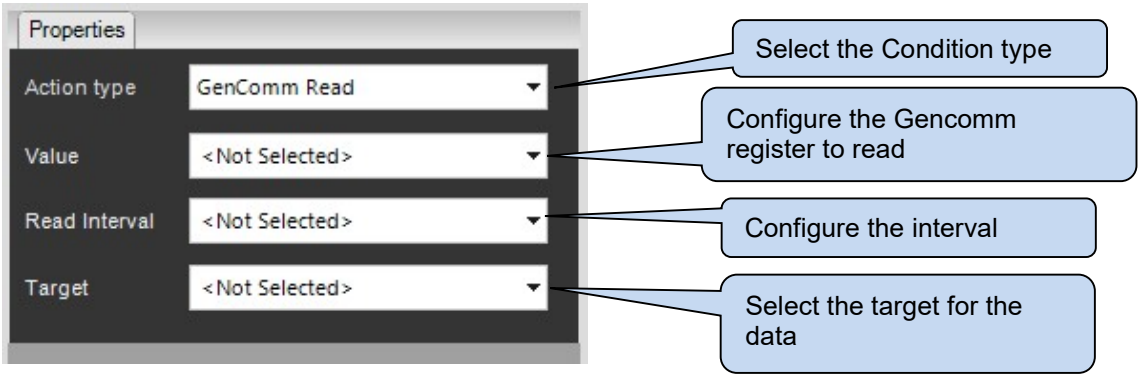
If no condition is true, (which is very unlikely in the above example) the string *Vent Fan* remembers its last requested display.

3.3.3.16 GENCOMM READ

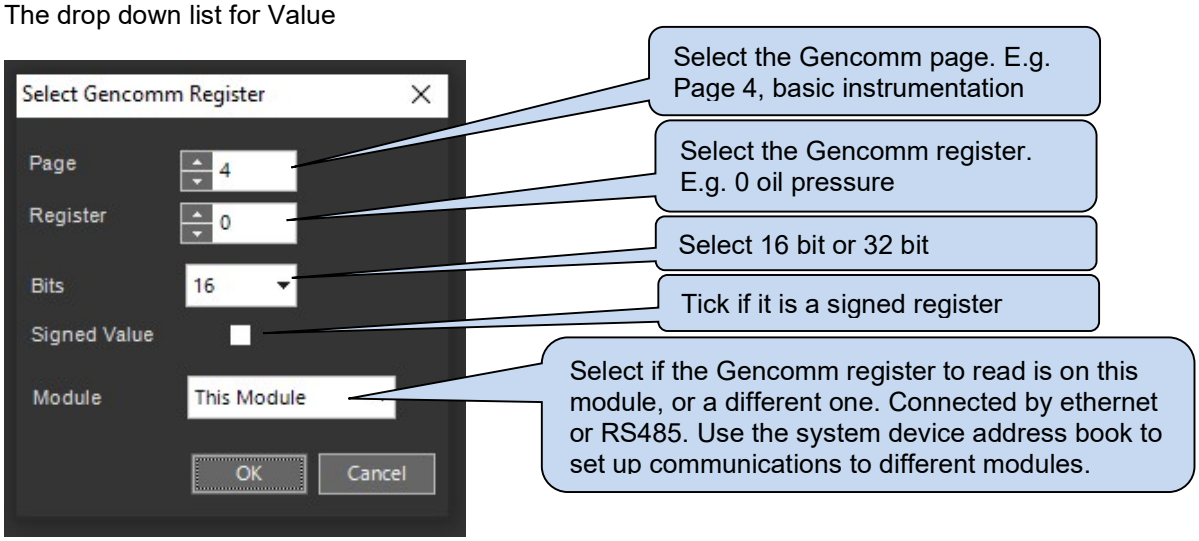
Reads the value of the specified GenComm register (Single / Cyclic) and copies the value to the specified Variable or Persistent Variable.

Drag the *GenComm Read* 

into the relevant place on the ladder rung and configure its properties.

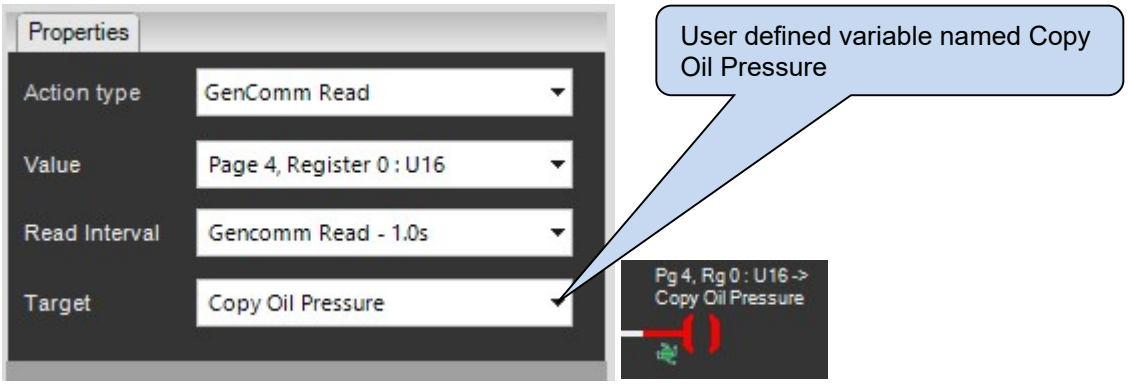


The drop down list for Value



Example

The below example shows a *GenComm Read* that copies page 4 register 0 unsigned 16-bit register (which is oil pressure) to user defined variable that has been named *Copy Oil Pressure*.



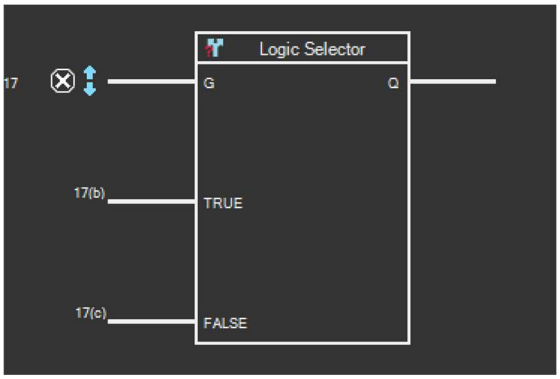
3.3.4 FUNCTION BLOCKS

3.3.4.1 BUILT IN

3.3.4.1.1 LOGIC SELECTOR

Selects between one of two Coil States based upon the driving condition to Drive the Output. If the driving condition is True, the Output is Driven by the state of the TRUE Input. If the driving condition is False, the Output is Driven by the state of the FALSE Input.

Its symbol is shown below

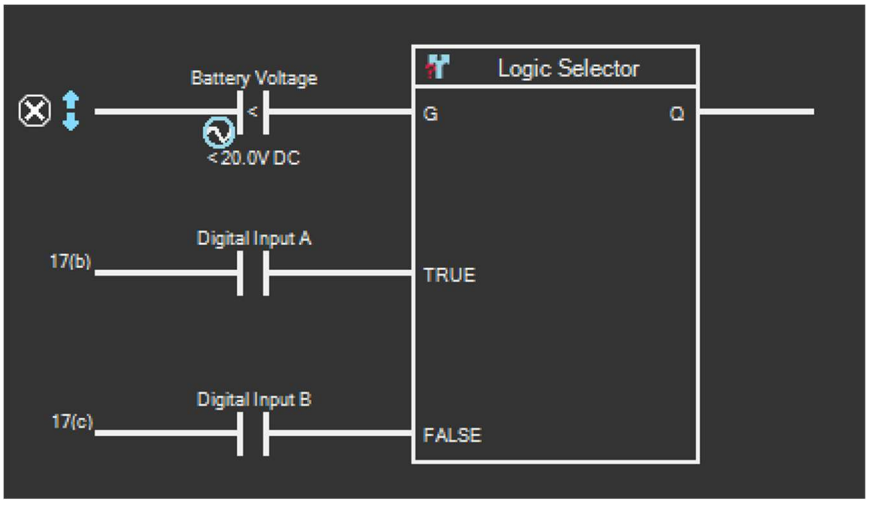


Drag the *Logic Selector* tool into the relevant place on the ladder rung. There are no properties for the logic selector.

Example

In the example below output Q follows *Digital Input A* if the battery voltage is below 20.0V.

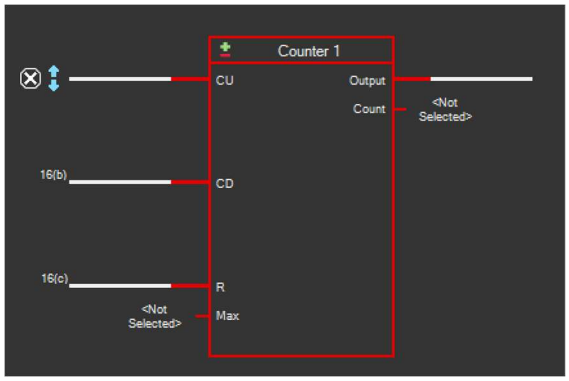
Output Q follows *Digital Input B* if the battery voltage is greater than 20.0V.



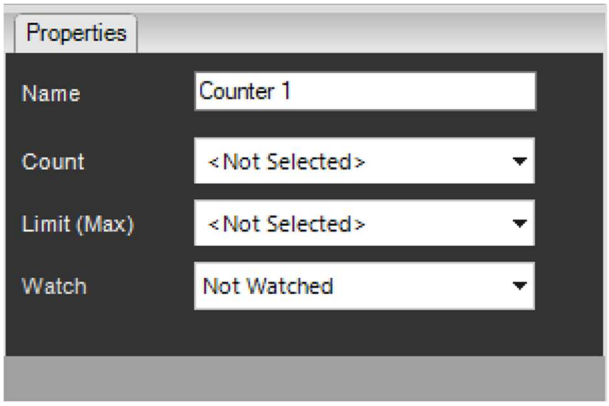
3.3.4.1.2 COUNTER MANIPULATOR

Allows the management of a *Counter* using a single function block.
Once the Counter Value has reached the *Counter Limit*, the Output is Driven to be True. The *Counter Limit* is set by a *Counter Variable, Variable or Persistent Variable*.

Its symbol is shown below



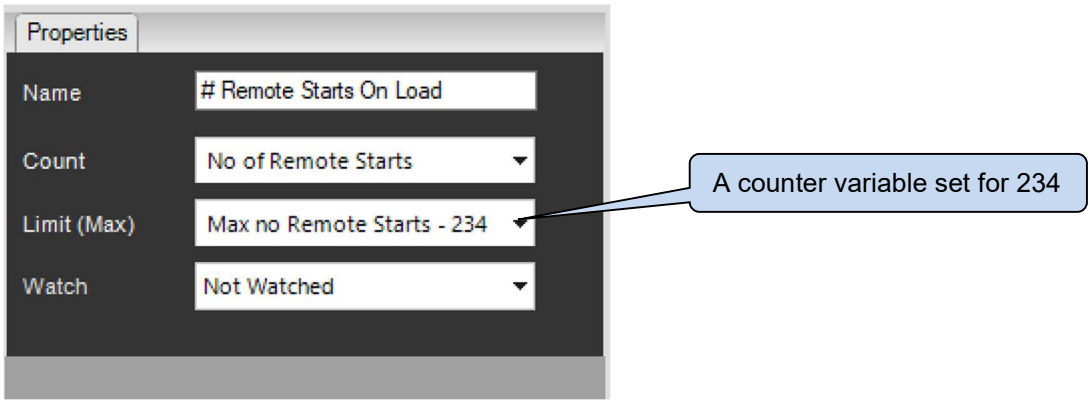
Drag the *Counter Manipulator* tool into the relevant place on the ladder rung and configure its properties.

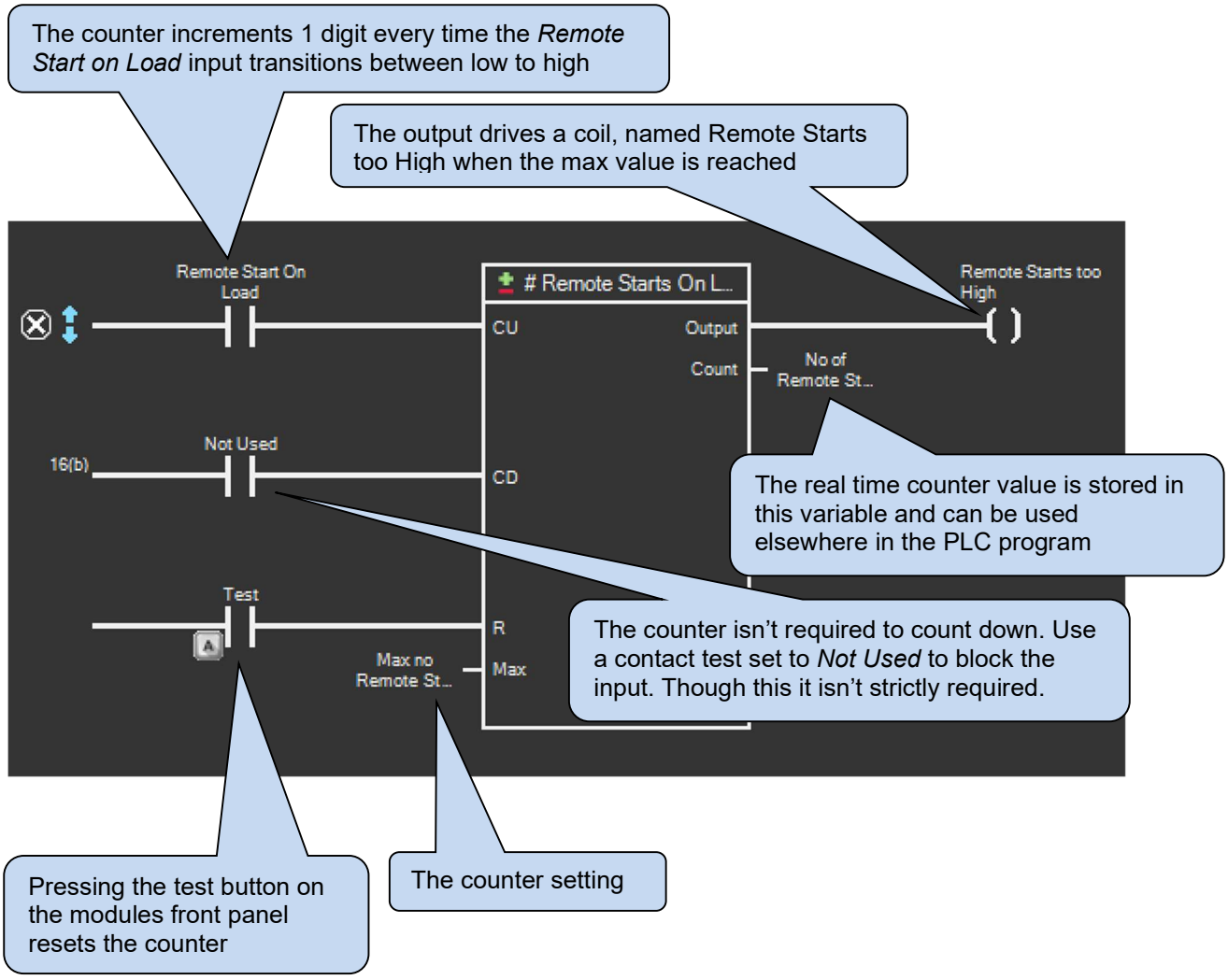


Parameter	Description
Name	The counters name. Rename the counter to a meaningful name
Count	This parameter is optional and should be left as *Not Selected* if it is not required **Not Selected:** The realtime count is not stored and cannot be used elsewhere **Persistent Variable:** The realtime count is stored in none volatile memeory and can be used else where in the PLC program **Variable:** The realtime count is stored in volotile memeory and can be used else where in the PLC program
Limit (Max)	Select the limit of the counter. The counters output becomes true when the counter reaches this value **Not Selected:** The is the default setting, but it must be configured to one of the below. **Counter Limit Variable:** The counter limit is stored in a dedicated counter variable and can only be adjusted from the *Variables* icon in the Tool Bar **Persistent Variable:** The counter limit is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name **Variable:** The counter limit is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Watch	Select if the counter is to be watched. Refer to section entitled *Watched Items* else where in this document.

Example

The below example shows a *Counter,* re-named as # Remote Starts on Load, it's output is high when the count reaches the Limit (Max).

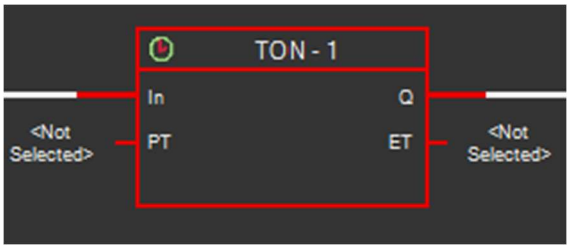




3.3.4.1.3 TIMER ON

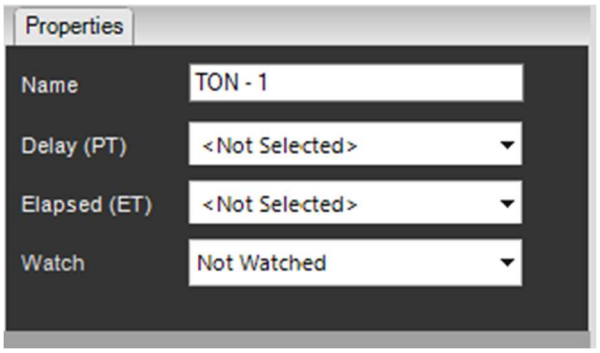
Timer On is also known as a TON and its symbol is shown below. A Timer On (TON) delays a condition becoming true.

Its symbol is shown below



If the driving condition to *In (Input)* is True, the *ET (Elapsed Time)* begins. Once the *ET (Elapsed Time)* has reached the *PT (Delay)* value, *Q (Output)* is driven to be True. If the driving condition to the *In (Input)* is False at any time, the timer *ET (Elapsed Time)* resets and *Q (Output)* is driven to be False.

Drag the *Timer On* tool into the relevant place on the ladder rung and configure its properties.



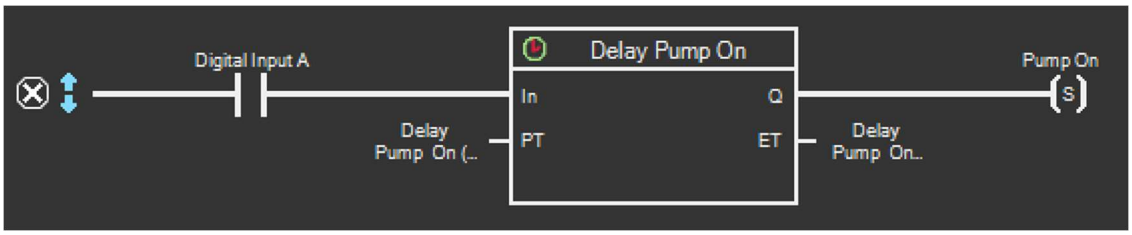
Parameter	Description
Name	The timers name. Rename the timer to a meaningful name
Delay (PT)	Select the limit of the timer. The timers output becomes true when the timer reaches this value **Not Selected:** The is the default setting, but it must be configured to one of the below. **Persistent Variable:** The timer limit is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name **Timer Variable:** The timer limit is stored in a dedicated timer variable and can only be adjusted from the *Variables* icon in the Tool Bar **Variable:** The timer limit is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Elapsed (ET)	This parameter is optional and should be left as *Not Selected* if it is not required **Not Selected:** The elapsed time is not stored and cannot be used elsewhere **Persistent Variable:** The elapsed time is stored in none volatile memeory and can be used else where in the PLC program **Variable:** The elapsed time is stored in volotile memeory and can be used else where in the PLC program
Watch	Select if the timer is to be watched. Refer to section entitled *Watched Items* else where in this document.

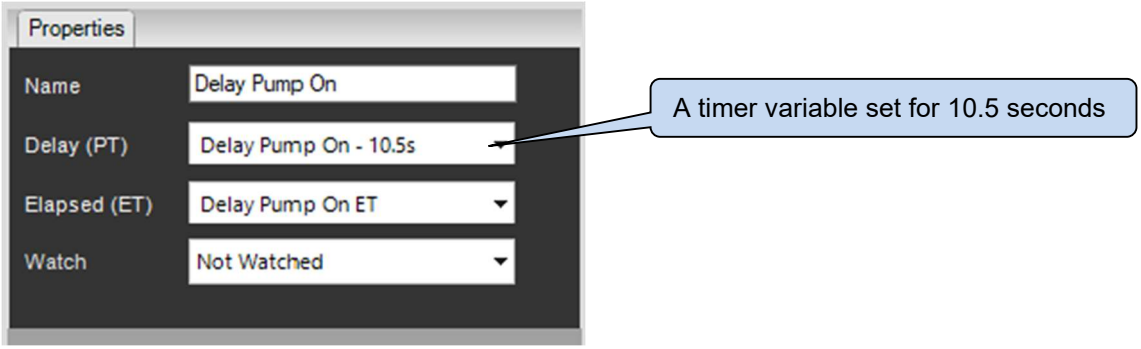
Example

The below example shows a *Timer On,* re-named *as* Delay Pump On.

If *Digital Input A* is active, the *ET (Elapsed Time)* increases. 10.5 seconds after *Digital Input A* first activates, the *ET (Elapsed Time)* matches the *PT (Delay),* and *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named Pump On is *Set.*

If *Digital Input A* is not active, the *ET (Elapsed Time)* resets to zero and *Q (Output)* is driven to False. Even though *Q (Output)* is driven to False, the *Coil* named Pump On may still be *Set* as there is no PLC to *Reset* it.

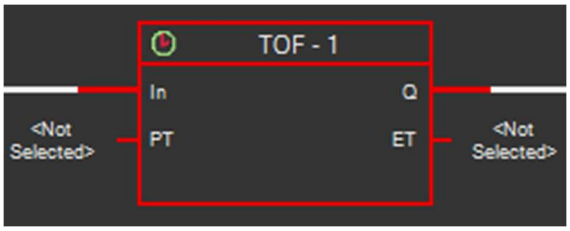




3.3.4.1.4 TIMER OFF

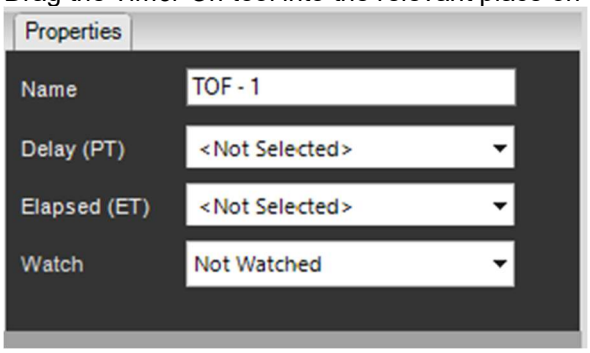
Timer Off is also known as a TOF and its symbol is shown below. A *Timer Off (TOF)* delays a condition becoming False.

Its symbol is shown below



If the driving condition to *In (Input)* is True, *Q (Output)* is immediately driven to be True. If the driving condition to the *In (Input)* then becomes False, the *ET (Elapsed Time)* begins. Once the *ET (Elapsed Time)* has reached the *PT (Delay)* value, *Q (Output)* is driven to be False. If the driving condition to the *In (Input)* is True at any time, the timer *ET (Elapsed Time)* resets and *Q (Output)* is driven to be True.

Drag the *Timer On* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
Name	The timers name. Rename the timer to a meaningful name
Delay (PT)	Select the limit of the timer. The timers output becomes true when the timer reaches this value **Not Selected:** The is the default setting, but it must be configured to one of the below. **Persistent Variable:** The timer limit is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name **Timer Variable:** The timer limit is stored in a dedicated timer variable and can only be adjusted from the *Variables* icon in the Tool Bar **Variable:** The timer limit is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Elapsed (ET)	This parameter is optional and should be left as *Not Selected* if it is not required **Not Selected:** The elapsed time is not stored and cannot be used elsewhere **Persistent Variable:** The elapsed time is stored in none volatile memeory and can be used else where in the PLC program **Variable:** The elapsed time is stored in volotile memeory and can be used else where in the PLC program
Watch	Select if the timer is to be watched. Refer to section entitled *Watched Items* else where in this document.

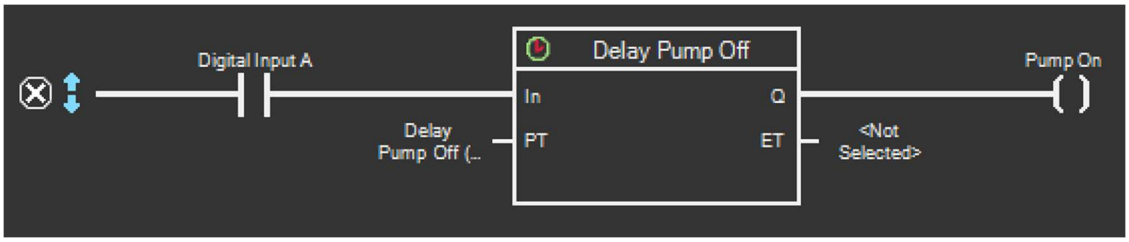
Example

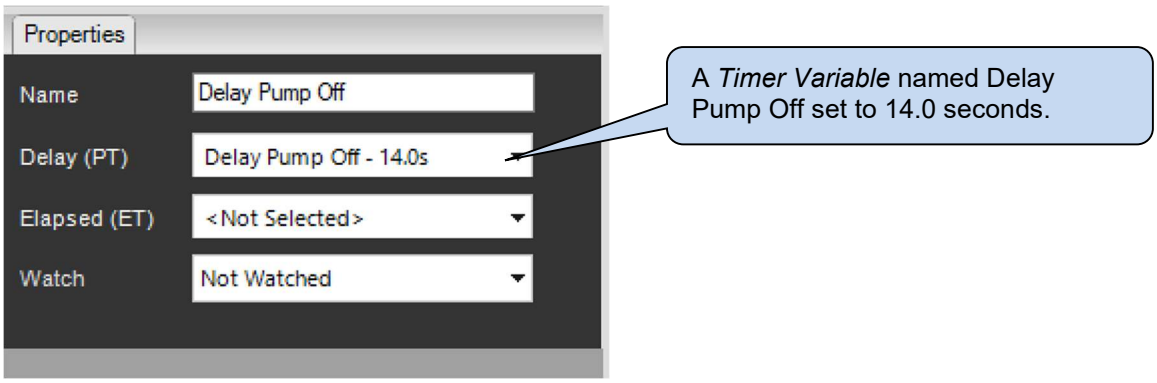
The below example shows a *Timer Off,* re-named *as* Delay Pump Off.

If *Digital Input A* is active, *Q (Output)* is immediately driven to be True. As *Q (Output)* is driven to True, the *Coil* named Pump On is *also driven to be True*

If *Digital Input A* is deactivated, the *ET (Elapsed Time)* increases. 14.0 seconds after *Digital Input A* first deactivated, the *ET (Elapsed Time)* matches the *PT (Delay),* and *Q (Output)* is driven to False. As *Q (Output)* is driven to False, the *Coil* named Pump On is *also driven to False.*

If *Digital Input A* is active, the *ET (Elapsed Time)* resets to zero and *Q (Output)* is driven to True.

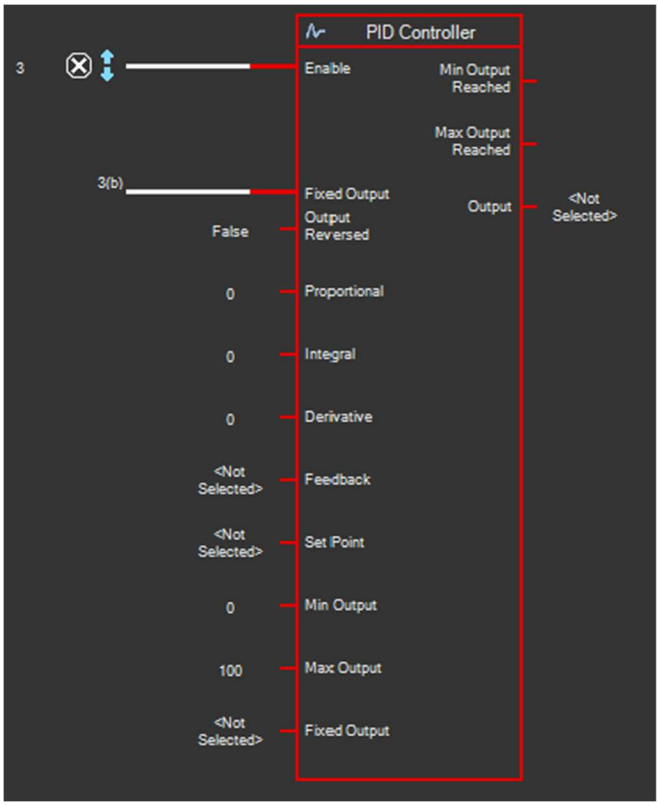




3.3.4.1.5 PID CONTROLLER

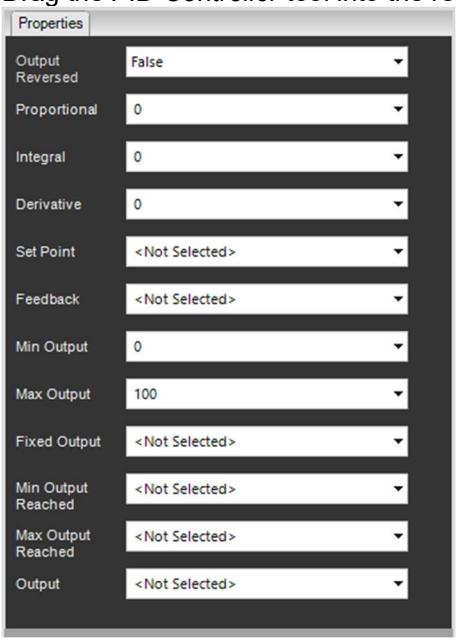
Allows closed loop control of the Output to maintain the Feedback at the Set Point. The Output adjustment is based on the Proportional (P), Integral (I) and Derivative (D) settings of the controller

Its symbol is shown below



The *PID controller* adjusts the output so that the feedback is equal to the set point. Proportional, Integral and Derivative control speed of response and overshoot.

Drag the *PID Controller* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
Output Reversed	When true the controller works to bring the feedback down to the set point e.g. Cooling fan. When false the controller works to bring the feedback up to the set point e.g. Heater.
Proportional	Select the proportional of the controller. **Fixed Value:** Enter the value manually **Persistent Variable:** The value is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name. This option makes tuning the PID easy when the generator is running. **Variable:** The value is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Integral	Select the integral of the controller. **Fixed Value:** Enter the value manually **Persistent Variable:** The value is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name. This option makes tuning the PID easy when the generator is running. **Variable:** The value is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Derivative	Select the derivative of the controller. **Fixed Value:** Enter the value manually **Persistent Variable:** The value is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name. This option makes tuning the PID easy when the generator is running. **Variable:** The value is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Set Point	The target the controller is working to **Fixed Value:** Enter the value manually **Persistent Variable:** The value is stored in none volatile memeory and can be set else where in the PLC program or the modules front panel if watch is selected. The *persistant Variable* should be given a meaningful name. This option makes tuning the PID easy when the generator is running. **Variable:** The value is stored in volatile memeory and can be set else where in the PLC program. The *Variable* should be given a meaningful name
Feedback	Select the instrument to be monitored and controlled

Parameter	Description
Minimum Output	The minimum output the internal algorithm drives to. This can be any number between -2147483648 and 2147483647 **Fixed Value:** Enter the value manually. **Persistent Variable:** The value is stored in none volatile memeory. The *persistant Variable* should be given a meaningful name. **Variable:** The value is stored in volatile memeory. The *Variable* should be given a meaningful name.
Maximum Output	The maximum output the internal algorithm drives to. This can be any number between -2147483648 and 2147483647
Fixed Output	This input is used to drive the internal algorithm to a fixed output, e.g. 50%
Min Output Reached	Boolian output that is driven true when the internal algorithm has reached its minimum point. Select the name of the *coil* the output drives.
Max Output Reached	Boolian output that is driven true when the internal algoritham has reached its maximum point. Select the name of the *coil* the output drives.
Output	The output of the algoritham is used to control third party equipment such as the speed of a fan. Select the variable to be used. This variable should be mapped to a DSE2152 analogue output. **Persistent Variable:** The value is stored in none volatile memeory. The *persistant Variable* should be given a meaningful name. **Variable:** The value is stored in volatile memeory. The *Variable* should be given a meaningful name

Example

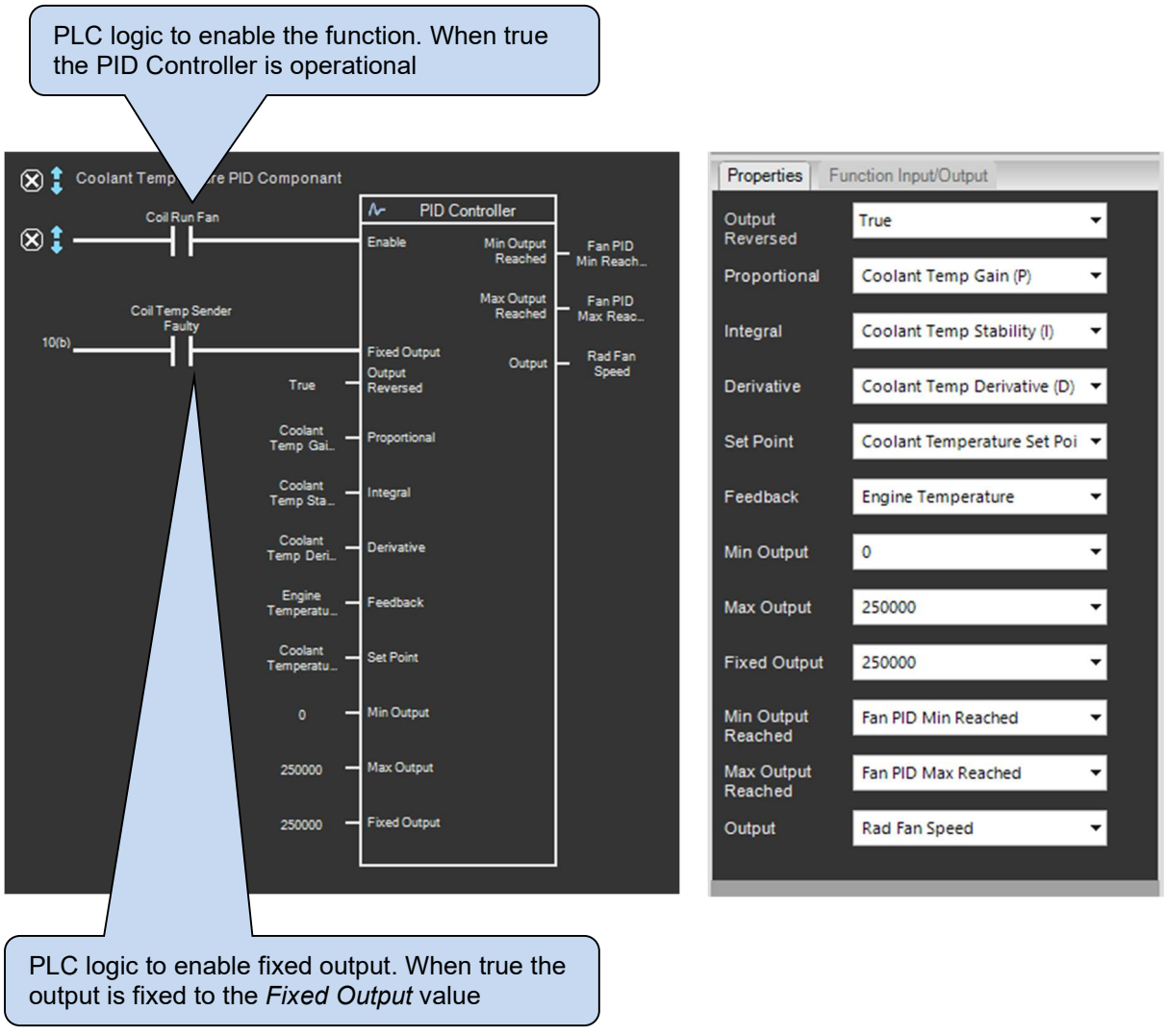
The below example shows a *PID Controller,* configured to control the speed of a fan to keep the engine at a set temperature.

The *PID Controller* is turned on when the logic gate *Coil Run Fan* is true. When the *PID Controller* is not operational its *Output* is 0.

The values for Proportional, Integral, Derivative, and set points are all *Persistent Variables* which allows them to be adjusted whilst the engine is running, tuning the *PID controller.*

If *the logic gate* Coil Temp Sender Faulty becomes true, the PID Controller output is switched to 250,000 which in this case is maximum output.

The *PID Controller* adjusts the Output between 0 and 250,000, which in turn controls the speed of the fan via a DSE2152 analogue output.



3.3.4.1.6 RISING EDGE TRIGGER

Looks for the Transition of *In (Input)* from False to True. *Q (Output)* is then true for 100ms before returning to False. *Q (Output)* remains False until *In (Input)* returns to False and subsequently again transitions from False to True.

Its symbol is shown below

Drag the *Rising Edge* tool into the relevant place on the ladder rung. It does not have any properties.

Example

The below example shows a *Rising Edge Trigger*

When digital input K is turned on the *Rising Edge Trigger* allows one pulse through to the mathematical function. The mathematical function then adds 1 to the variable called Count On. Therefore, the variable Count On increments by 1 every time the digital input K is turned on.

3.3.4.1.7 FALLING EDGE TRIGGER

Looks for the Transition of *In (Input)* from True to False. *Q (Output)* is then true for 100ms before returning to False. *Q (Output)* remains False until *In (Input)* returns to True and subsequently again transitions from True to False.

Its symbol is shown below

Drag the *Falling Edge* tool into the relevant place on the ladder rung. It does not have any properties.

Example

The below example shows a *Falling Edge Trigger*

When digital input K is turned off the *Falling Edge Trigger* allows one pulse through to the mathematical function. The mathematical function then adds 1 to the variable called Count Off. Therefore, the variable Count Off increments by 1 every time the digital input K is turned off.

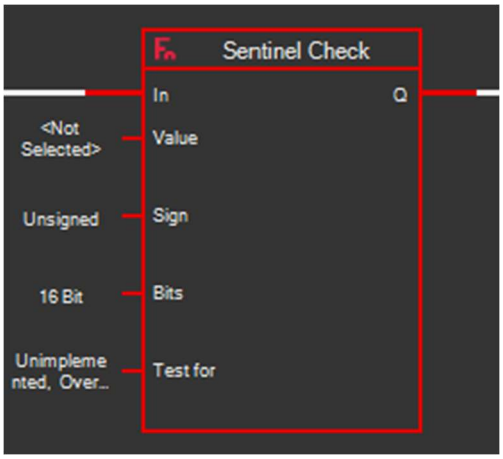
3.3.4.1.8 SENTINEL CHECK

This has the identical function of the *Sentinel Check* found in section 3.3.2 of the document under Tools, Conditions. Except the user interface is different.

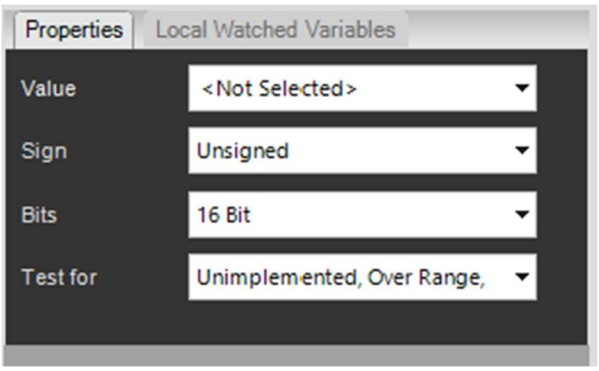
Checks the selected instrument to see if its current state is a Sentinel Value. Sentinels are to show that the instrument is not available, or out of range.

Q (Output) is True when *In (Input)* is True.

Its symbol is shown below



Drag the *Sentinel Check* tool into the relevant place on the ladder rung and configure its properties.

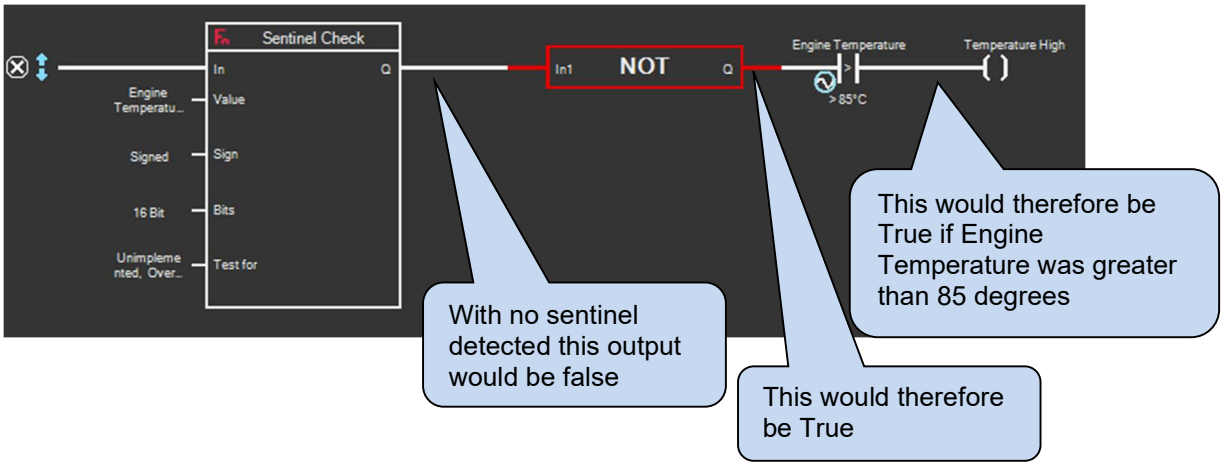


Parameter	Description
Value	The instrument to be checked for a sentinel. E.g. *Engine Temperature*
Sign	The sign of the *GenComm* register. Either *Unsigned* or *Signed*
Bits	The lenge of the *GenComm* register. Either 16 bits or 32 bits
Test For	Select the sentinels to check for. **Unimplemented:** The instrument is not configured **Over Range:** The instrument is reading values above its range **Under Range:** The instrument is reading values below its range **Tranducer Fault:** The instrument has no reading because of a faulty transducer or wiring. **Bad Data:** The instrument is not available because the ECU is powered down. **High Digital:** The instrument is configured as a digital switch and is true **Low Digital:** The instrument is configured as a digital switch and is false

Example

The below example shows a Sentinel Check configured for engine temperature. The condition is true if the instrument returns a selected sentinel

The *Sentinel Check Output Q* is True if Engine Temperature instrument is a Sentinel.
If the *Sentinel Check Output Q* is False, the *NOT Gate Output Q* is True and if the Engine Temperature is greater than 85 the coil Temperature High would be True.



3.3.4.2 MATHEMATICAL

These mathematical functions are very similar to the mathematical functions under the Action list, but they can only be configured for 2 values. They perform in the same way but the graphical representation in the PLC editor is very different.

3.3.4.2.1 ADD

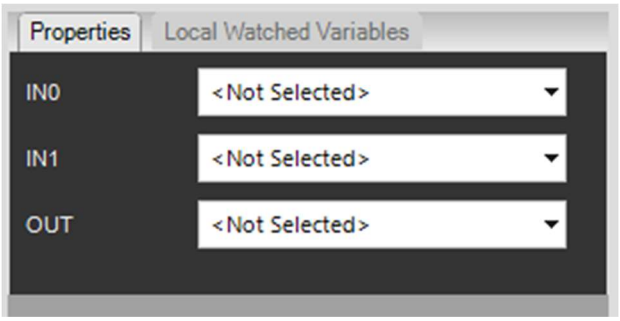
Allows a mathematical add function to be performed and the result stored in a *Variable* or *Persistent Variable*.

The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below

Drag the *Add* tool into the relevant place on the ladder rung and configure its properties.

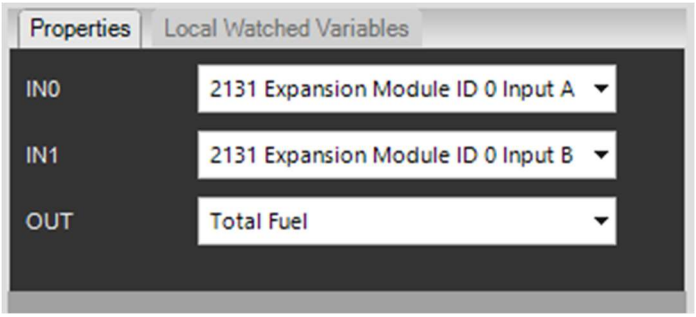


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is IN0 + IN1

The below example shows an Add function configured to add 2131 expansion module ID 0 input A to 2131 expansion module ID 0 input B and place the result in a *variable* called Total Fuel



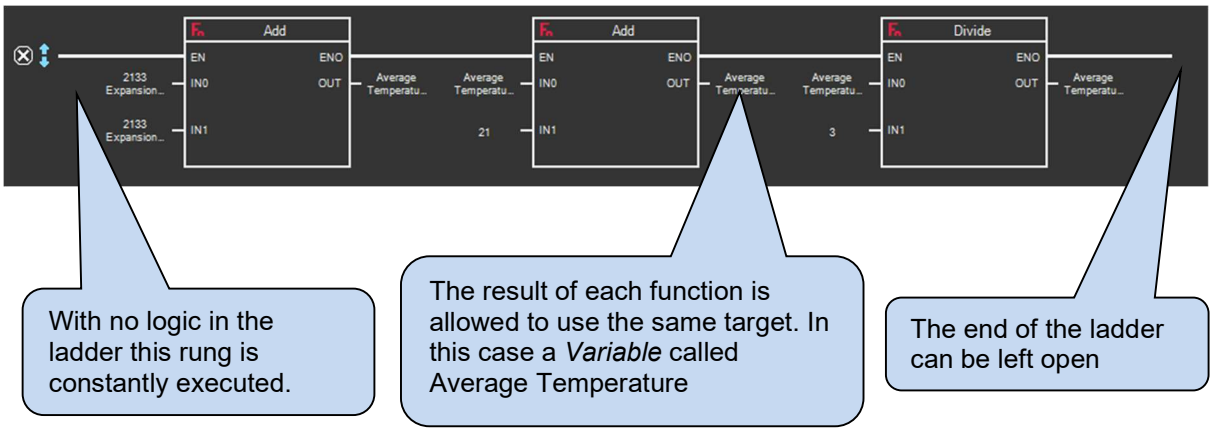
The below example, when *Digital Input A* is True, adds *2131 expansion module ID 0 input A* to *2131 expansion module ID 0 input B* and place the result in a *variable* called Total Fuel

When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Total Fuel remembers its last value

Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



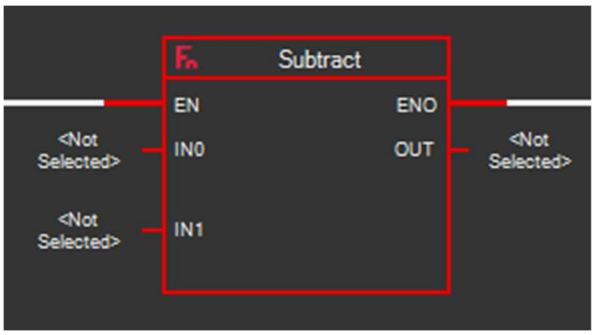
3.3.4.2.2 SUBTRACT

Allows a mathematical subtract function to be performed and the result stored in a *Variable* or *Persistent Variable*.

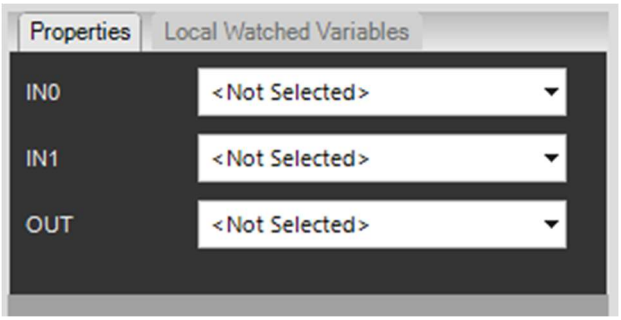
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



Drag the *Subtract* tool into the relevant place on the ladder rung and configure its properties.

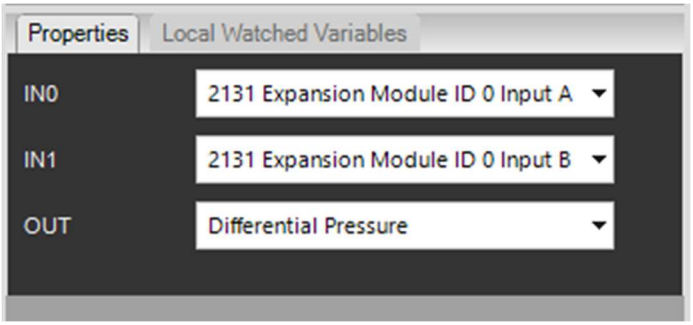


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is IN0 – IN1. If IN1 is larger than IN0 the result is a negative number.

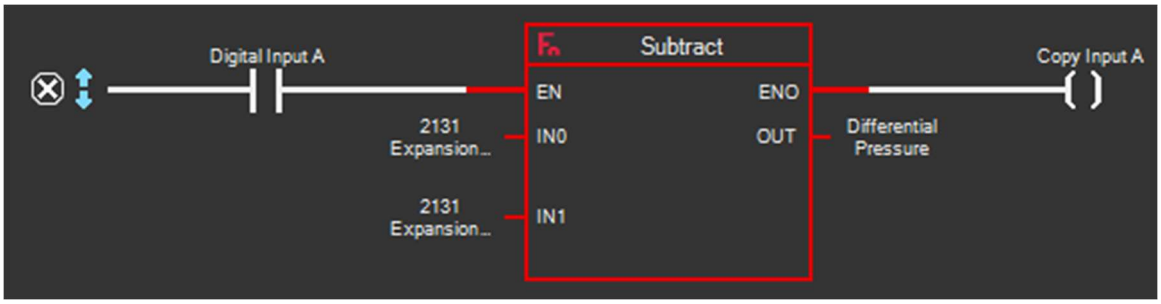
The below example shows a Subtract function configured to subtract *2131 expansion module ID 0 input B* from *2131 expansion module ID 0 input A* and place the result in a *variable* called Differential Pressure



The below example, when *Digital Input A* is True, subtracts *2131 expansion module ID 0 input B* from *2131 expansion module ID 0 input A* and place the result in a *variable* called Differential Pressure

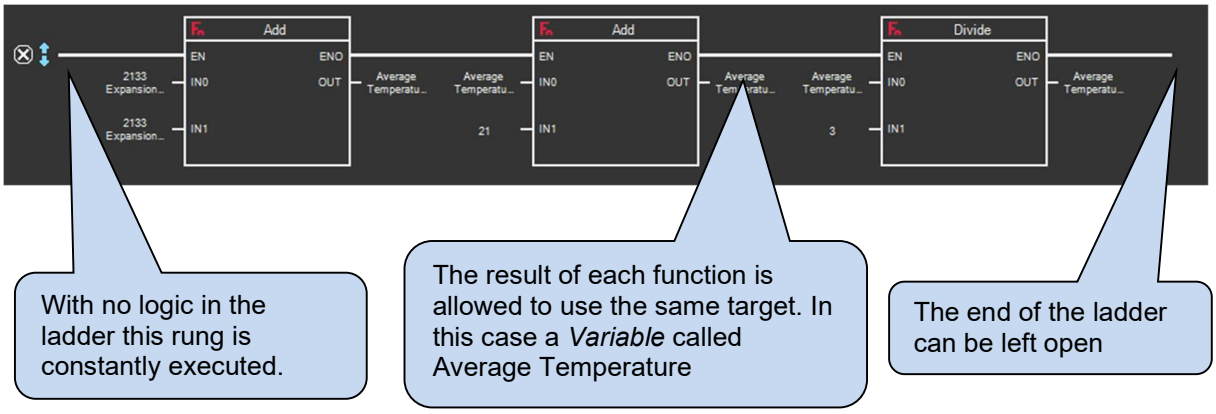
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also be *True*

When *Digital Input A* is *False* the *variable* called Differential Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



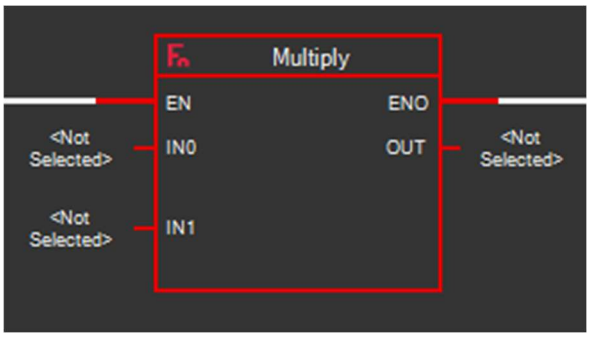
3.3.4.2.3 MULTIPLY

Allows a mathematical multiply function to be performed and the result stored in a *Variable* or *Persistent Variable*.

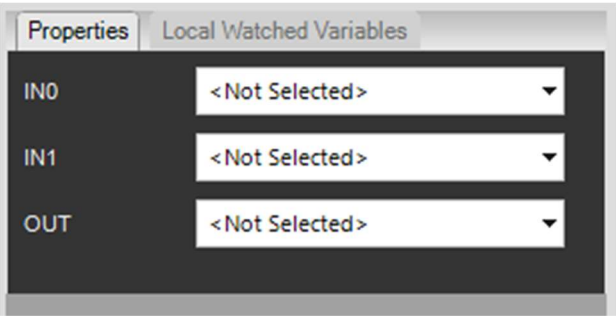
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



Drag the *Multiply* tool into the relevant place on the ladder rung and configure its properties.

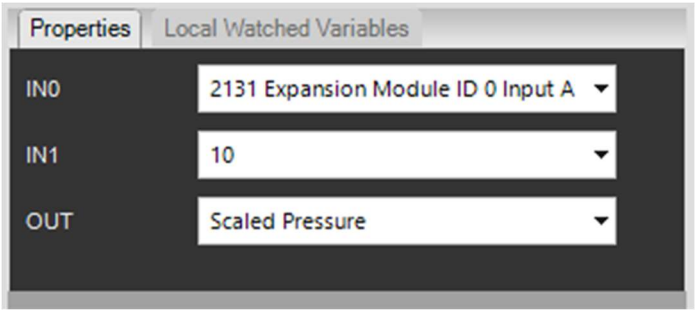


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is IN0 * IN1

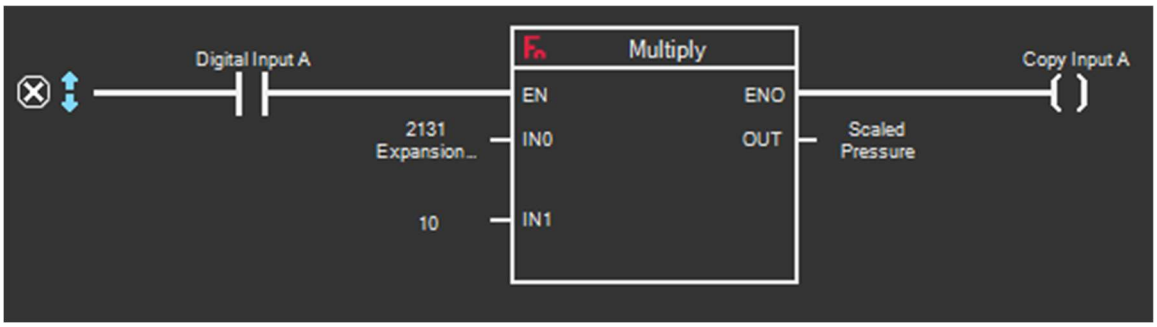
The below example shows a Multiply function configured to multiply *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Scaled Pressure



The below example, when *Digital Input A* is True, multiplies *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Scaled Pressure

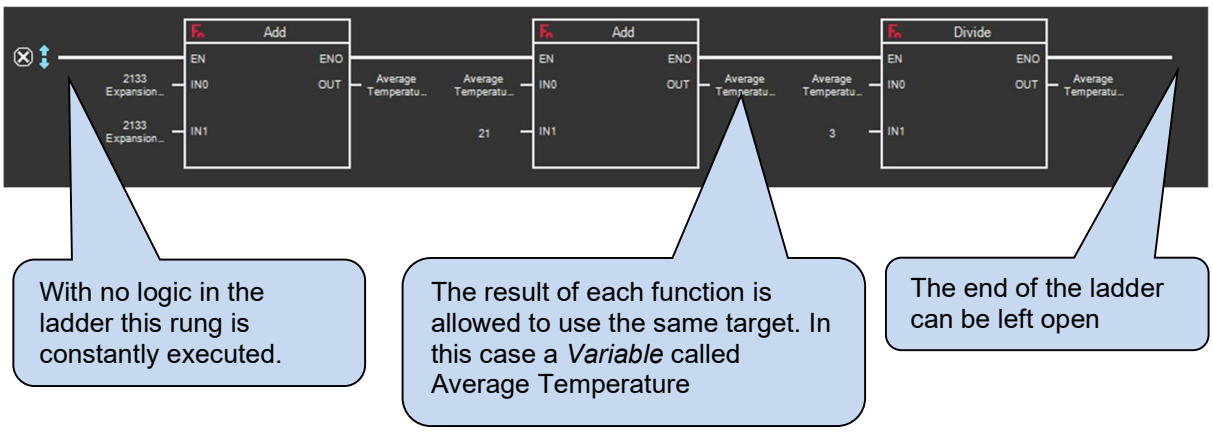
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also be *True*

When *Digital Input A* is *False* the *variable* called Scaled Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



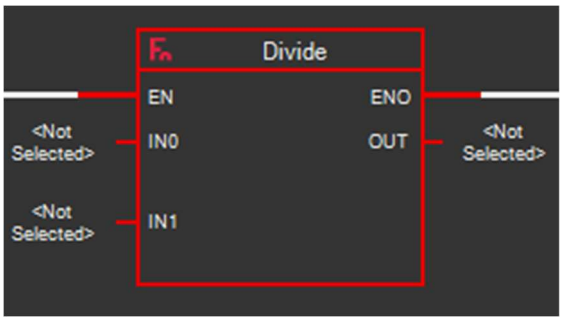
3.3.4.2.4 DIVIDE

Allows a mathematical divide function to be performed and the result stored in a *Variable* or *Persistent Variable*.

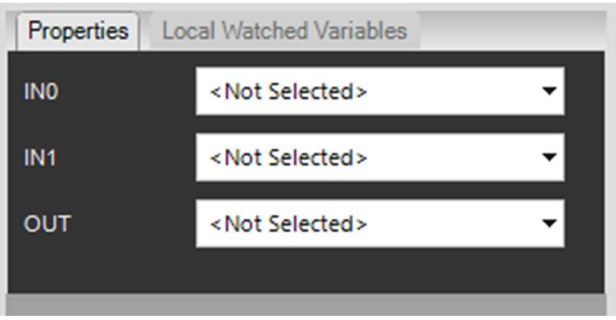
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



Drag the *Divide* tool into the relevant place on the ladder rung and configure its properties.

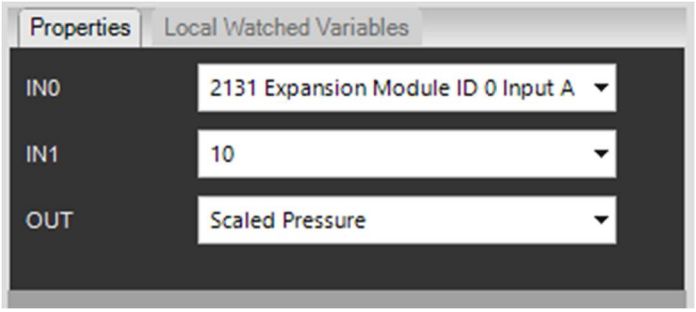


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is IN0 / IN1. Floating point maths is not supported, digits after the decimal point are lost. E.g. 86.9 is truncated to 86.

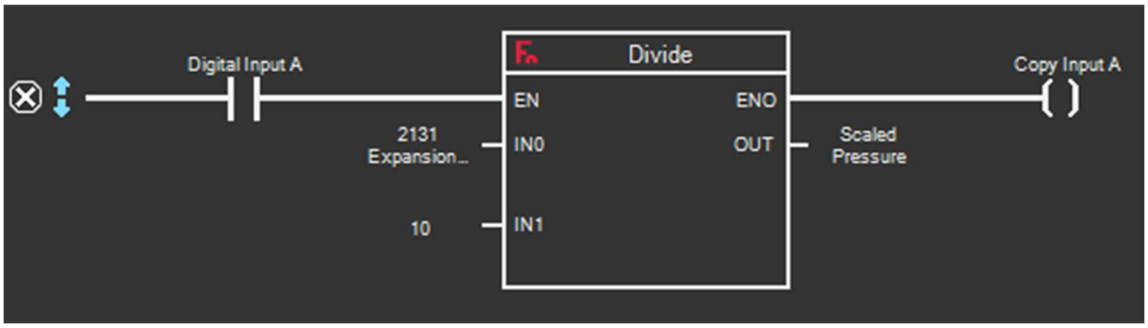
The below example shows a Divide function configured to divide *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Scaled Pressure



The below example, when *Digital Input A* is True, divides *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Scaled Pressure

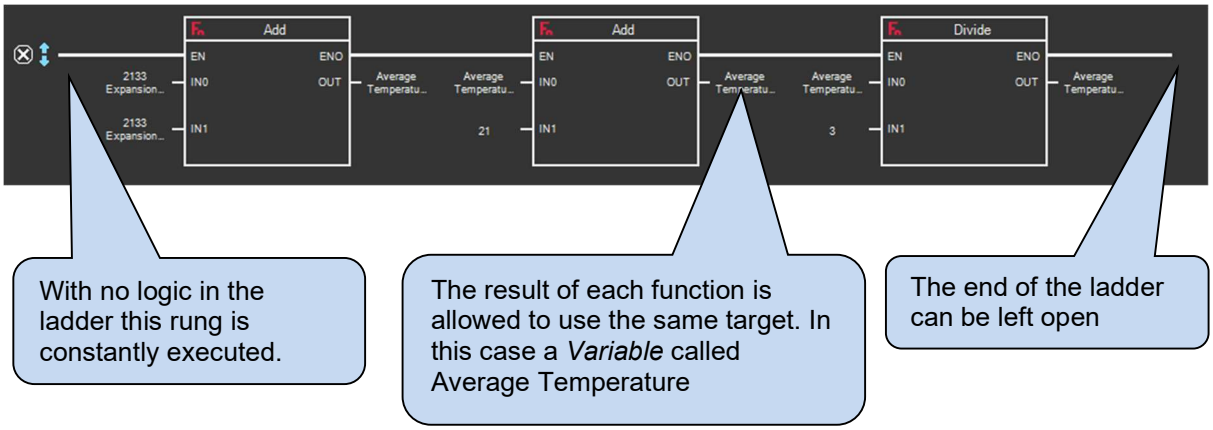
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Scaled Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



3.3.4.2.5 REMAINDER

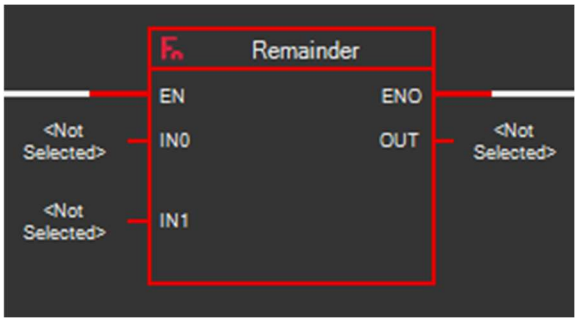
Allows a mathematical remainder function to be performed and the result stored in a Variable or Persistent Variable.

E.g. 100/46 = 2 remainder 8. Therefore 8 is stored

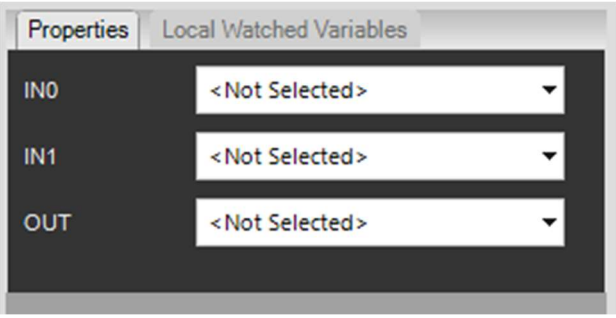
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



Drag the *Remainder* tool into the relevant place on the ladder rung and configure its properties.

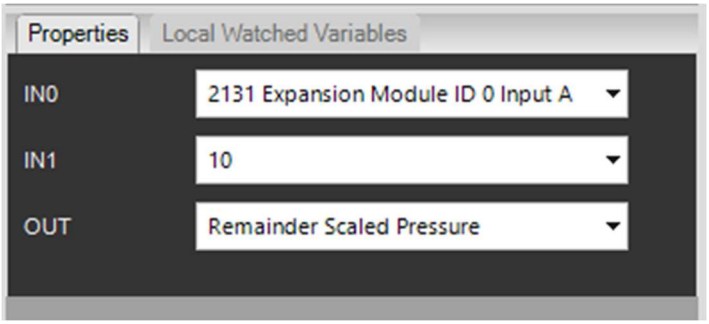


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is the remainder of IN0 / IN1. Floating point maths is not supported, digits after the decimal point are lost. E.g. 86.9 is truncated to 86.

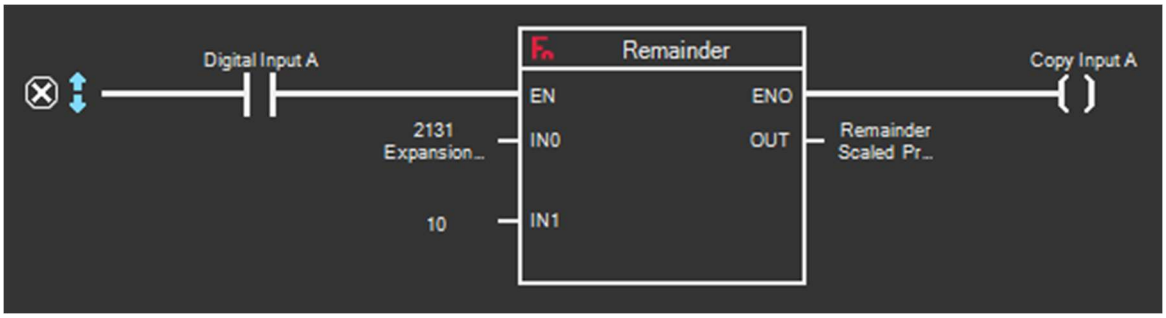
The below example shows a Remainder function configured to find the remainder of after dividing *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Remainder Scaled Pressure



The below example, when *Digital Input A* is True, divides *2131 expansion module ID 0 input A* by 10 and place the result in a *variable* called Remainder Scaled Pressure

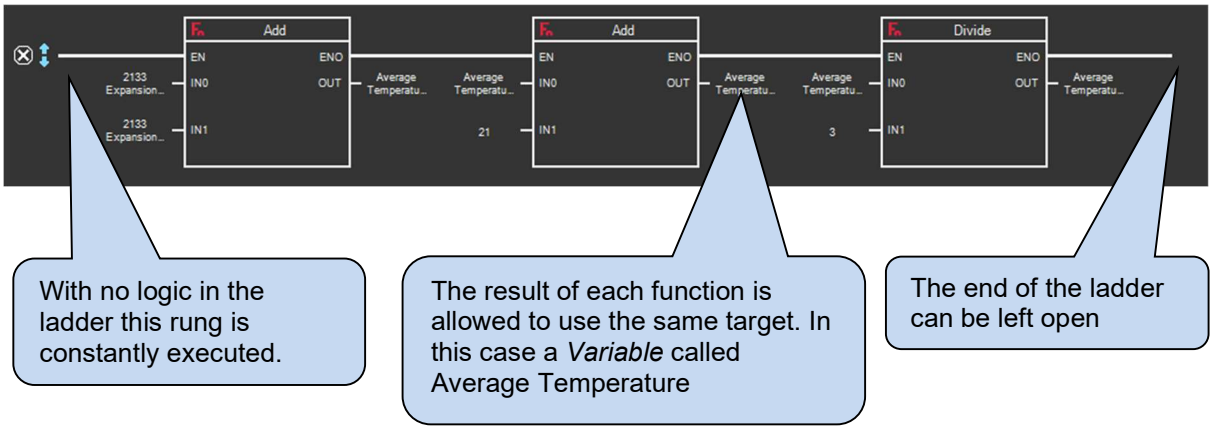
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Remainder Scaled Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



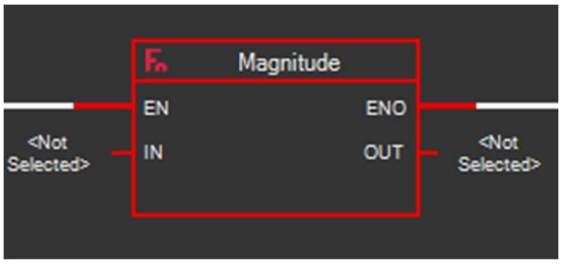
3.3.4.2.6 MAGNITUDE

Allows a mathematical magnitude function to be performed and the result stored in a Variable or Persistent Variable. Magnitude is the value with its 'sign' removed. For example, both 6 and -6 give a value of 6.

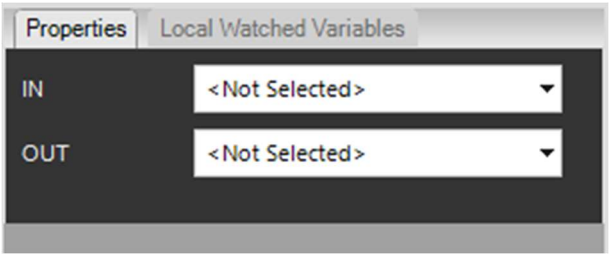
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



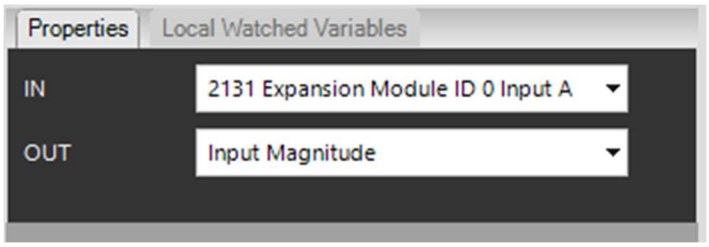
Drag the *Magnitude* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
IN	Select the instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

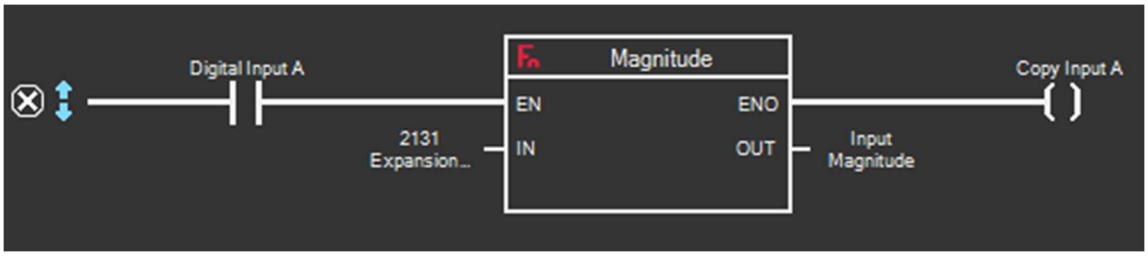
The below example shows a magnitude function configured to for *2131 expansion module ID 0 input A* place the result in a *variable* called Input Magnitude



The below example, when *Digital Input A* is True, remove the sign from *2131 expansion module ID 0 input A* and place the result in a *variable* called Input Magnitude

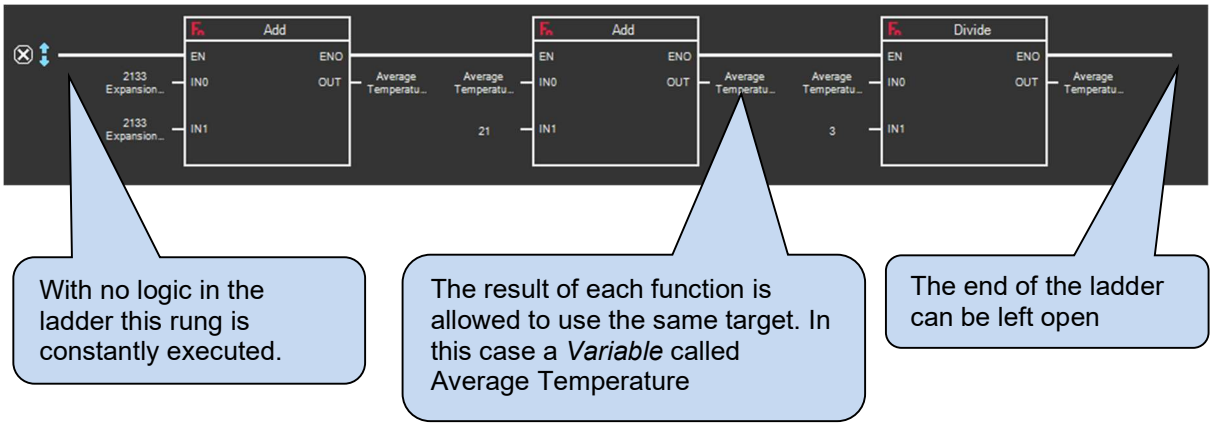
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Input Magnitude remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



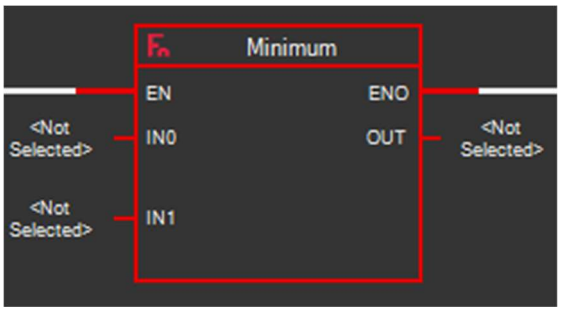
3.3.4.2.7 MINIMUM

Allows a mathematical minimum function to be performed and the result stored in a Variable or Persistent Variable.

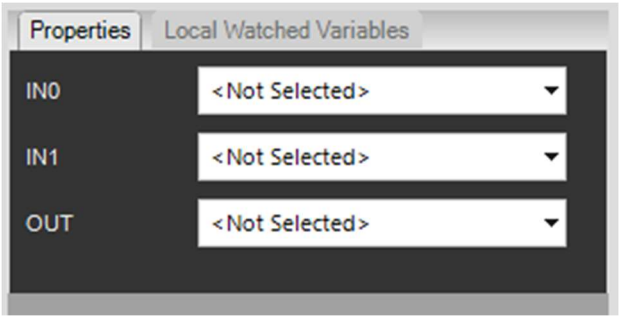
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



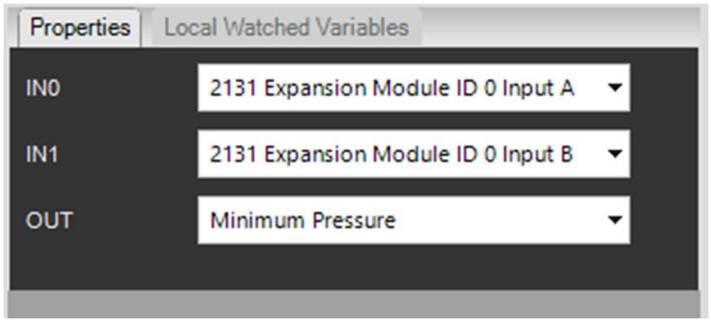
Drag the *Minimum* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

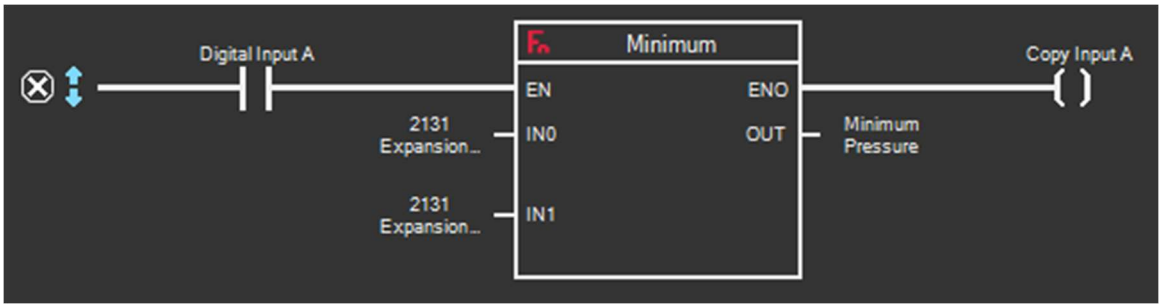
The below example shows a Minimum function configured to find the maximum value of 2131 expansion module ID 0 input A and 2131 expansion module ID 0 input B and place the result in a *variable* called Minimum Pressure



The below example, when *Digital Input A* is True, finds the Maximum of *2131 expansion module ID 0 input A* and *2131 expansion module ID 0 input B* and place the result in a *variable* called Minimum Pressure

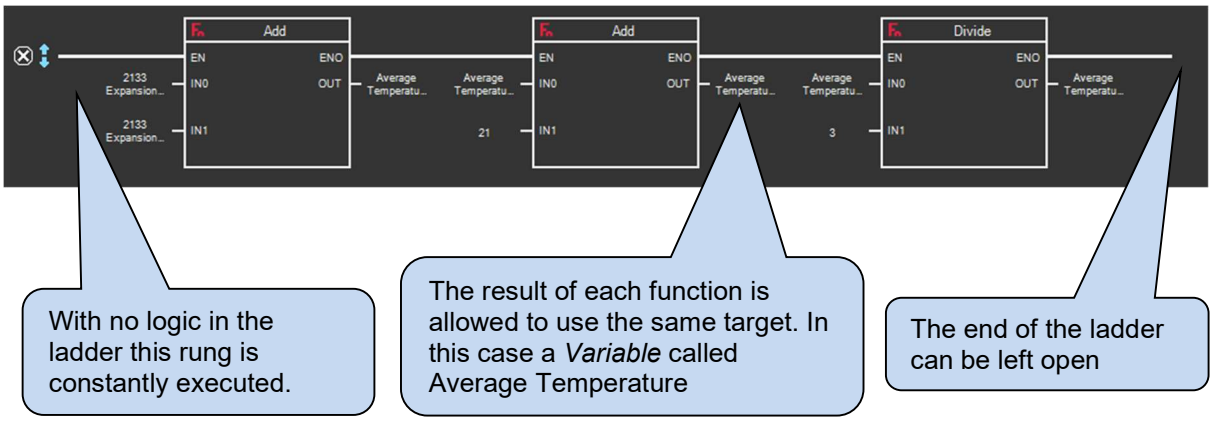
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Minimum Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



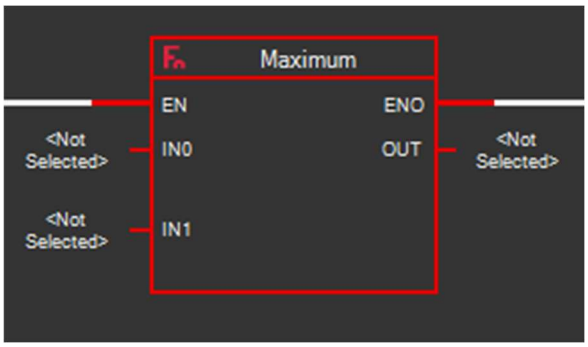
3.3.4.2.8 MAXIMUM

Allows a mathematical maximum function to be performed and the result stored in a Variable or Persistent Variable.

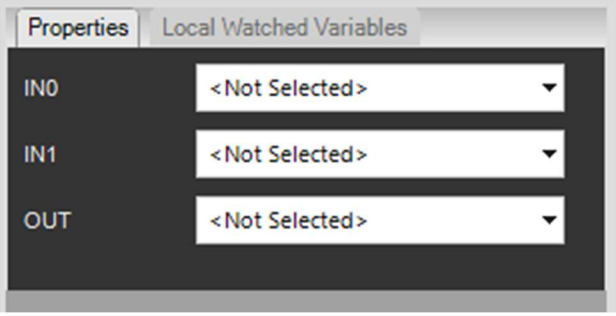
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



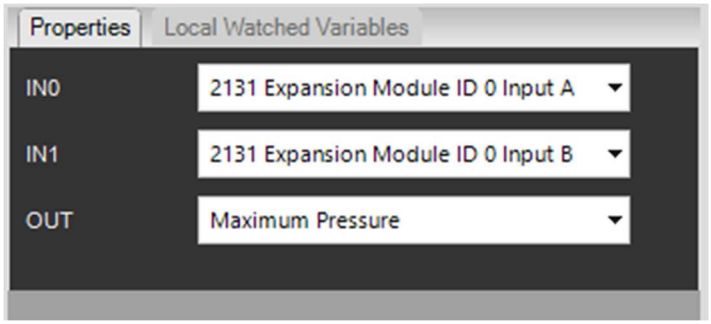
Drag the *Maximum* tool into the relevant place on the ladder rung and configure its properties.



Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

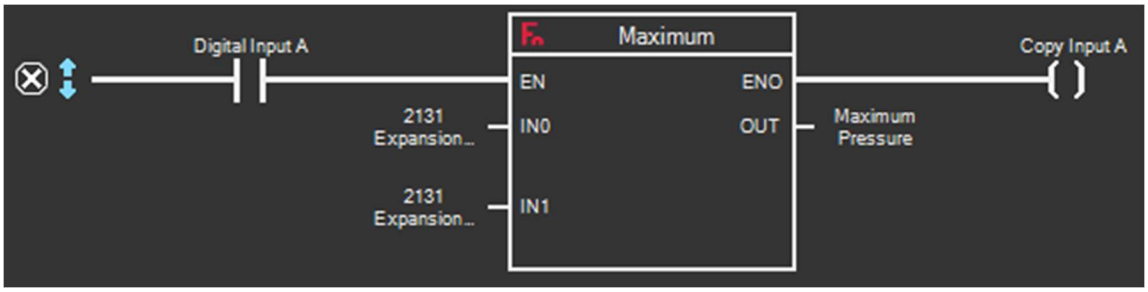
The below example shows a Maximum function configured to find the maximum value of 2131 expansion module ID 0 input A and 2131 expansion module ID 0 input B and place the result in a *variable* called Maximum Pressure



The below example, when *Digital Input A* is True, finds the Maximum of *2131 expansion module ID 0 input A* and *2131 expansion module ID 0 input B* and place the result in a *variable* called Maximum Pressure

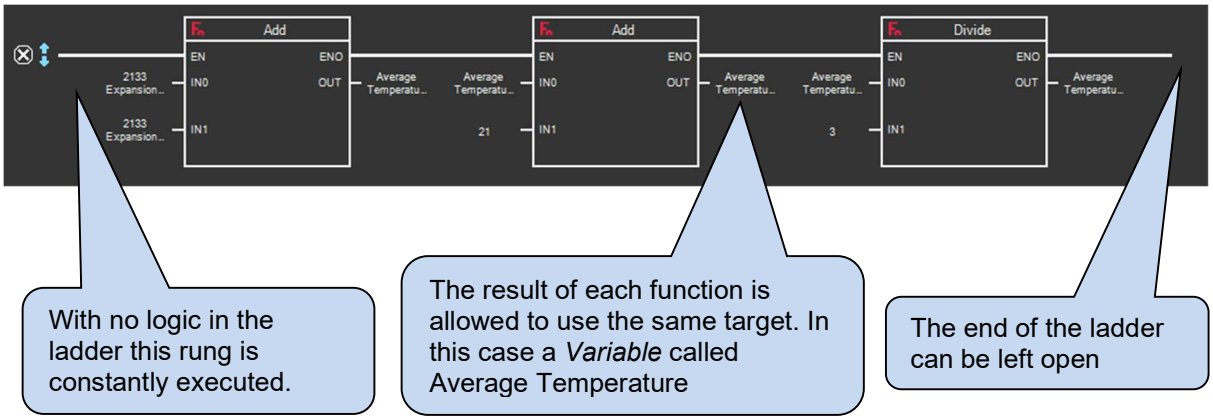
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Maximum Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



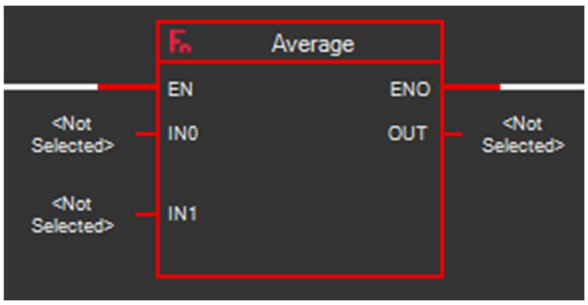
3.3.4.2.9 AVERAGE

Allows a mathematical average function to be performed and the result stored in a Variable or Persistent Variable.

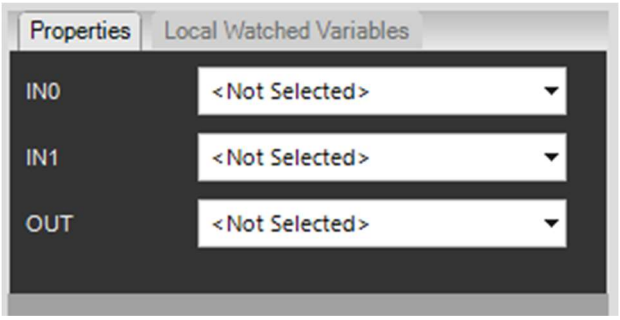
The mathematical function is performed when *EN (Input)* is *True*

ENO (pass through) is True when *EN (Input)* is *True*

Its symbol is shown below



Drag the *Average* tool into the relevant place on the ladder rung and configure its properties.

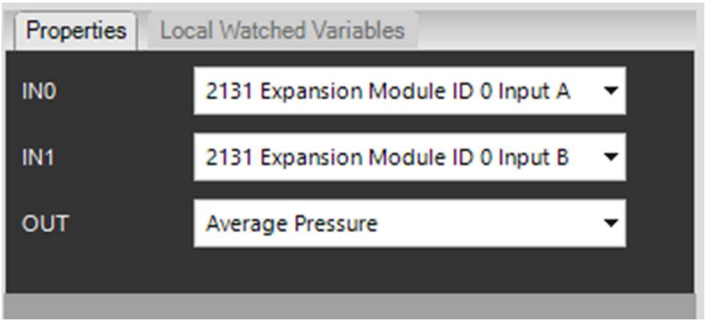


Parameter	Description
IN0	Select the first value to an instrument or constant
IN1	Select the second value to an instrument or constant
OUT	The location where the result of the mathematical operation is to be placed. ***Variable***: Values placed in the User Variables are lost when the module DC power is removed. ***Persistent Variable***: Values placed in the User Persistent Variables are maintained, even when the module DC power is removed. **MSC:** MSC Data A or MSC Data B are two registers that are transmitted over the MSC link and is read by every controller on the same MSC link. Described separately For further details of Variables and Persistent Variables, see section 3.1.6.1 in this document.

Example

NOTE: The equation is (IN0 + IN1) / 2

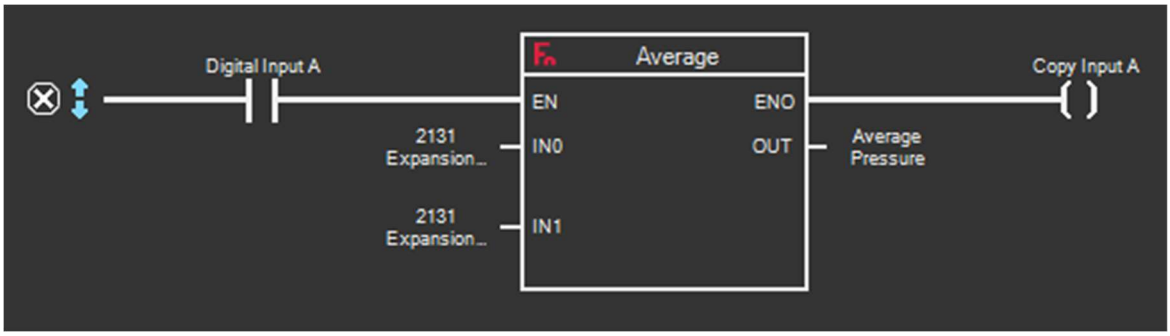
The below example shows an Average function configured to add 2131 expansion module ID 0 input A to 2131 expansion module ID 0 input B and place the result in a *variable* called Average Pressure



The below example, when *Digital Input A* is True, adds *2131 expansion module ID 0 input A* to *2131 expansion module ID 0 input B* and place the result in a *variable* called Average Pressure

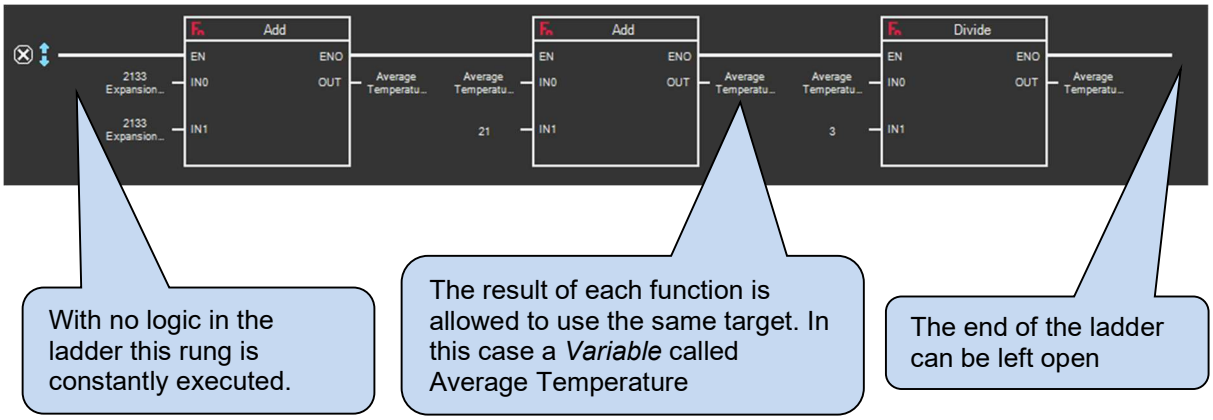
When *Digital Input A* is *True ENO* and therefore the coil called *Copy Input A* is also *True*

When *Digital Input A* is *False* the *variable* called Average Pressure remembers its last value



Example Of Multiple Mathematical Functions Connected Together

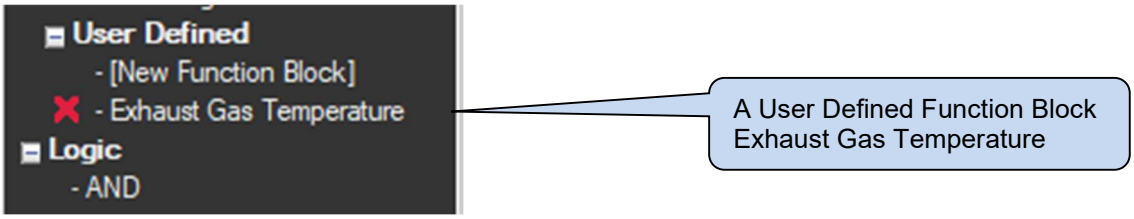
This example shows 3 mathematical functions on a single ladder rung. It is adding 3 temperatures together and dividing by 3 to find the average of the three. Ideal for alternator winding temperatures.



3.3.4.3 USER DEFINED

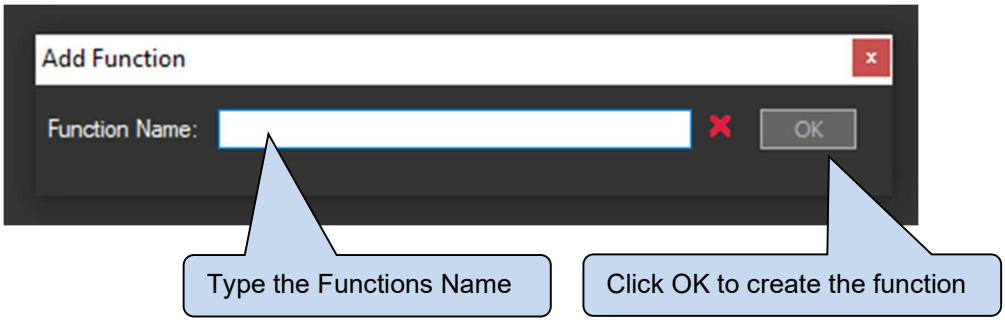
PLC programs usual contain more than one function. For example, a single PLC program could control the alternator heater, radiator cooling fan, exhaust gas temperatures and many more at the same time. Function Blocks allow these functions to be contained in their own function block, which improves readability, fault finding and repeatability.

All user defined Function Blocks are listed here so they can be dragged into the PLC editor.

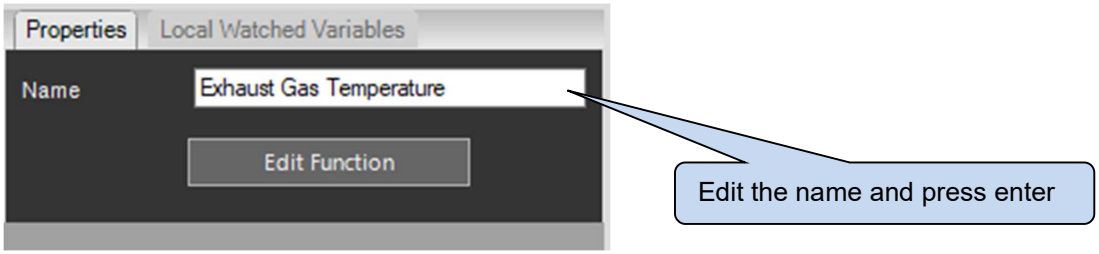


3.3.4.3.1 [NEW FUNCTION BLOCK]

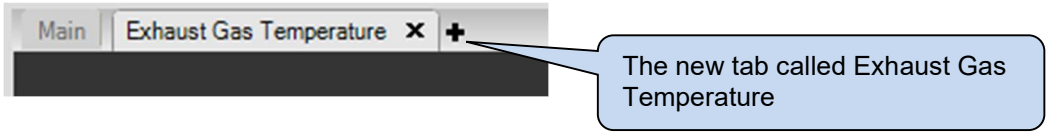
Drag *[New Function Block]* tool into the relevant place on the ladder rung, give it a name and click on OK



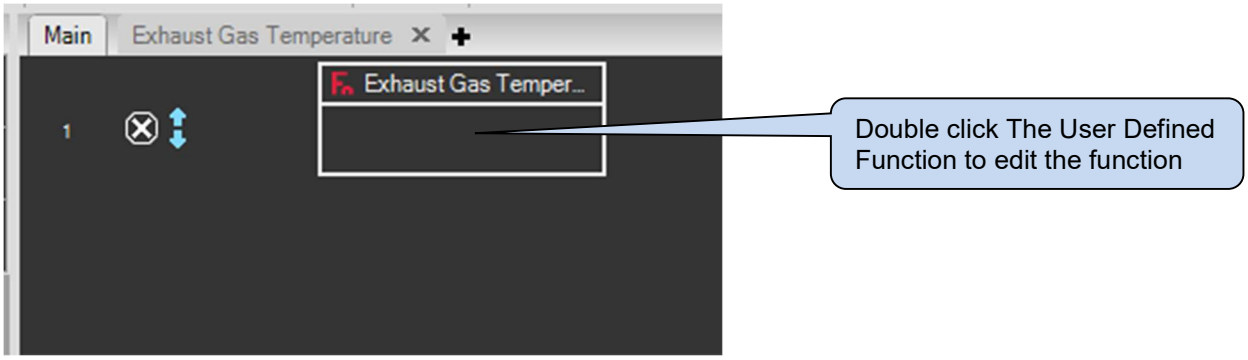
Its name can be changed using the properties box



Click on Edit Function and the editor opens a new tab. Create the PLC function in the new tab.



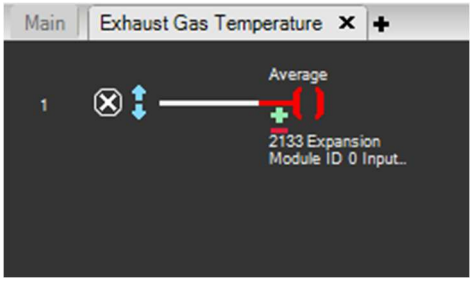
The Main tab contains a block representation of the *User Defined Function*.



Example 1

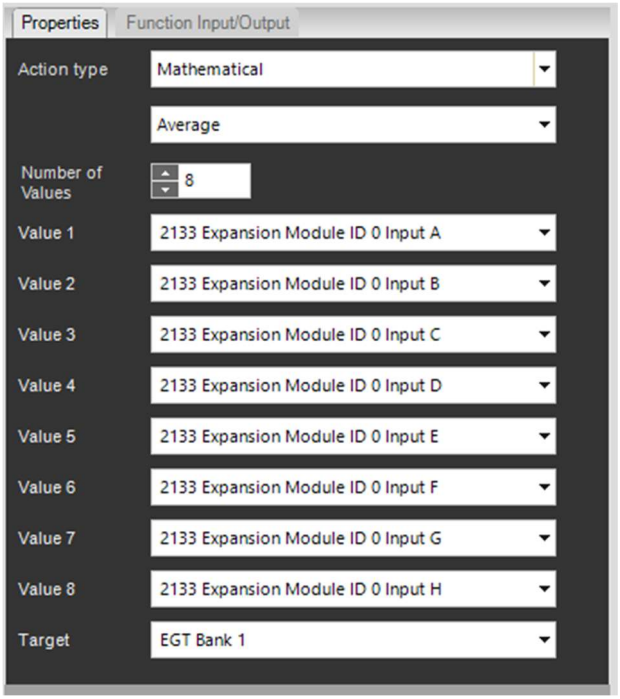
The below example is creating a User Defined Function to average the 8 exhaust gas temperatures of an 8-cylinder engine.

Create the User Defined Function as described above. Drag a mathematical Action into the editor and configure it for the average function of 8 inputs.



In this case the exhaust gas temperatures are monitored by 8 thermocouples connected to a 2133 expansion device 0.

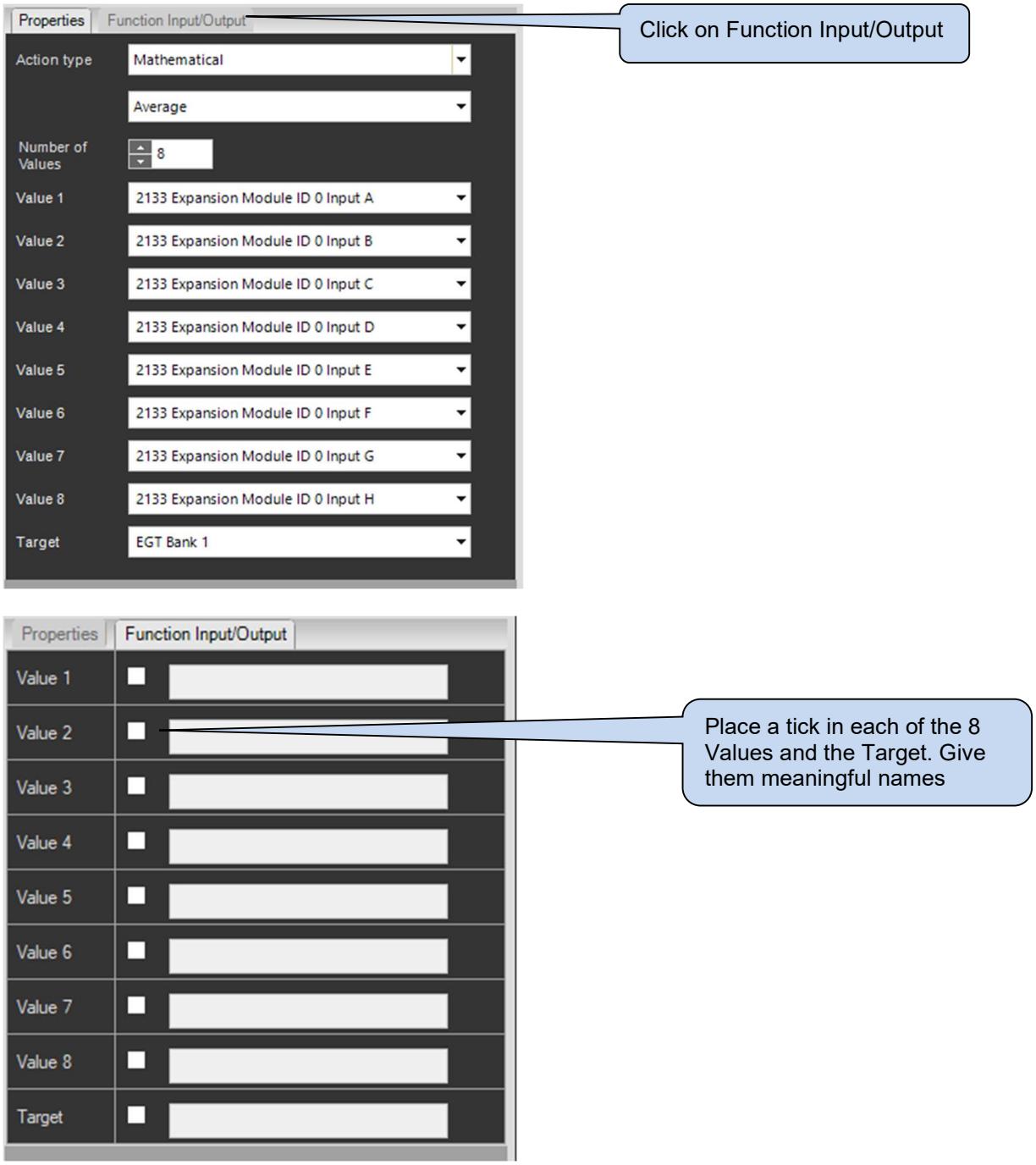
The output (the average of all 8 temperatures) is contained in a user *Variable* EGT Bank 1

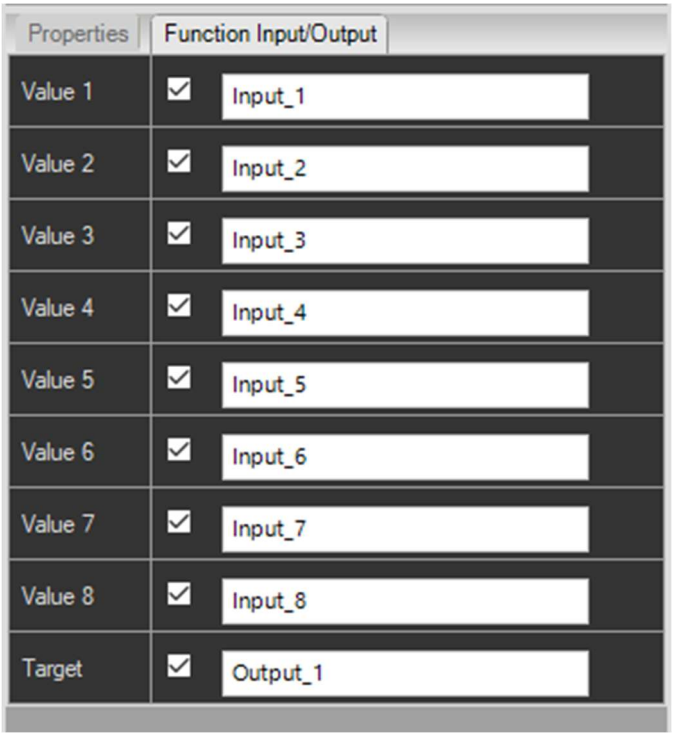


Example 2

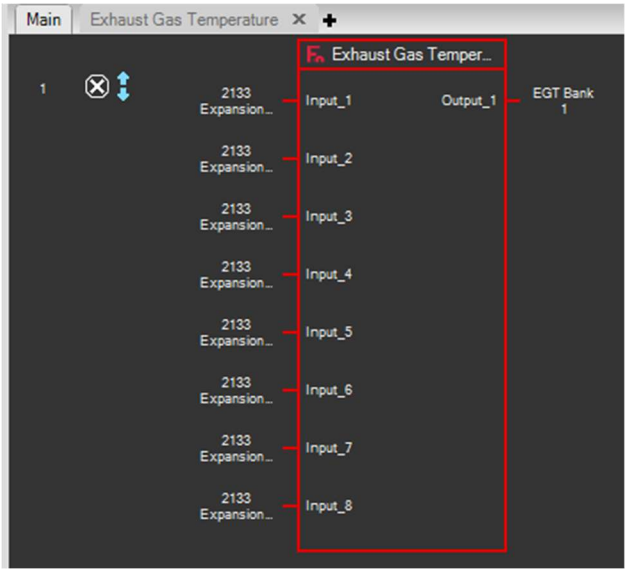
The above example shows a single use of the *User Defined Function.* However, it is often necessary to re-use existing function blocks, in which case it is necessary to pass information from the main PLC program into the function.

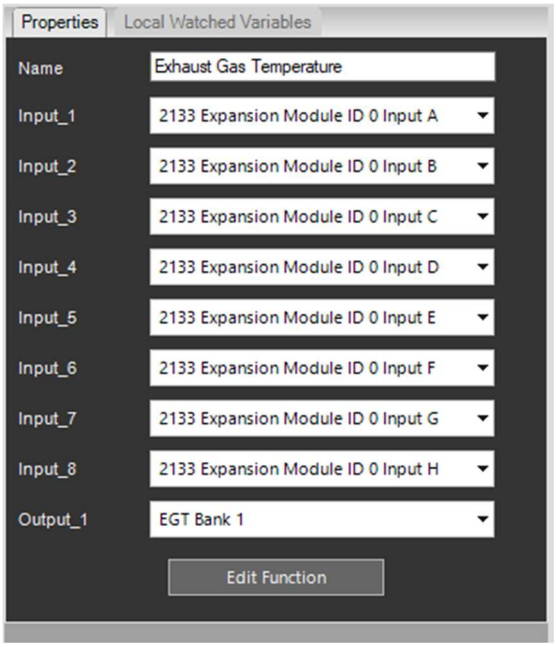
From the previous Properties box, click on Function Input/Output



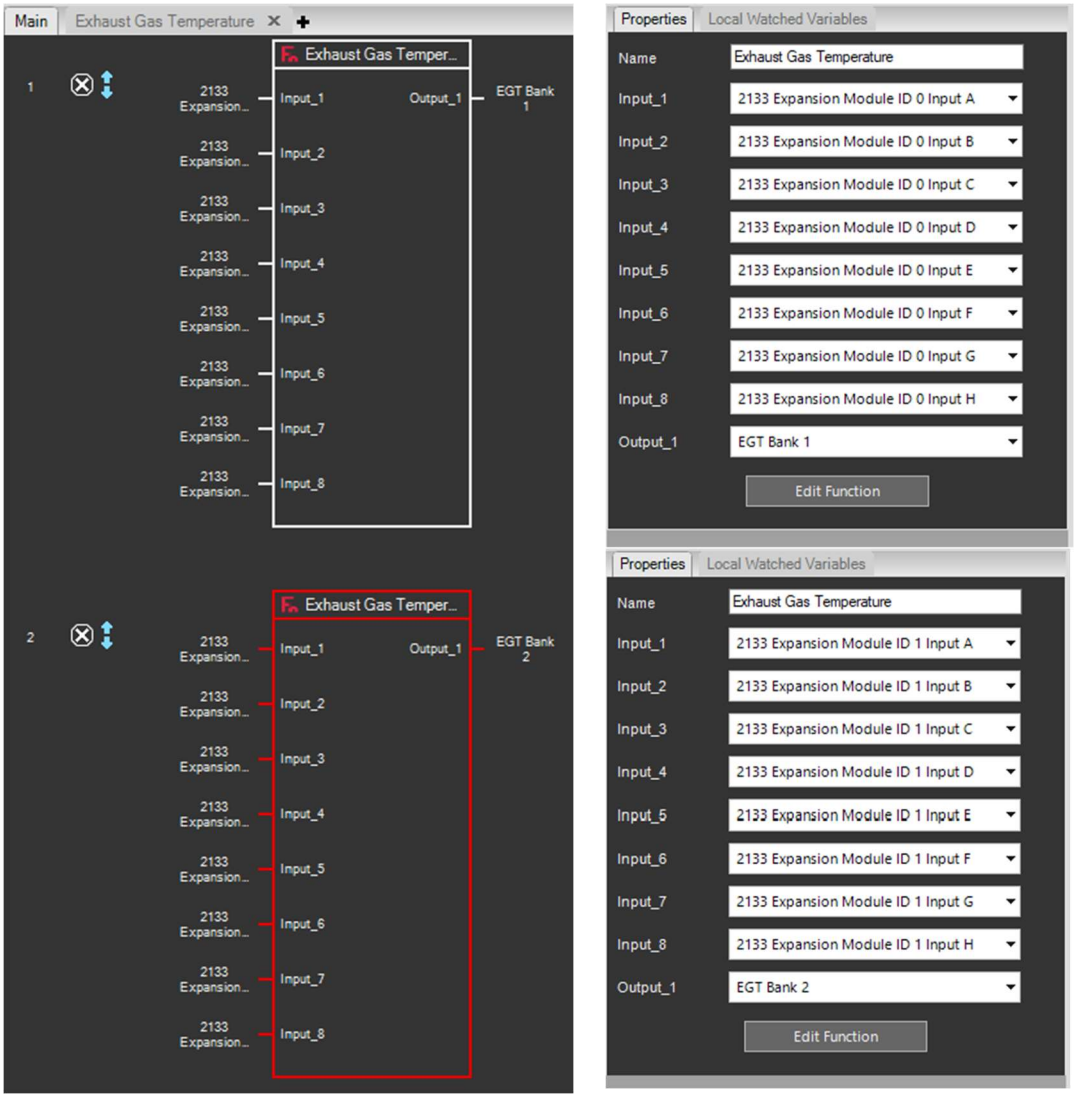


This changes the representation on the Main tab





It is now possible to drag the *User Defined Function* again into the Main Tab of the PLC editor and set the inputs and output to different instruments and *variables*.

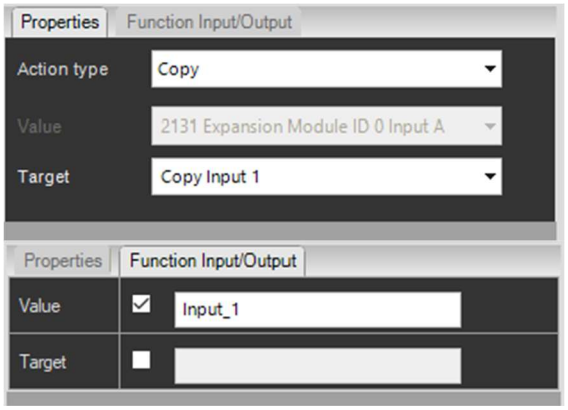


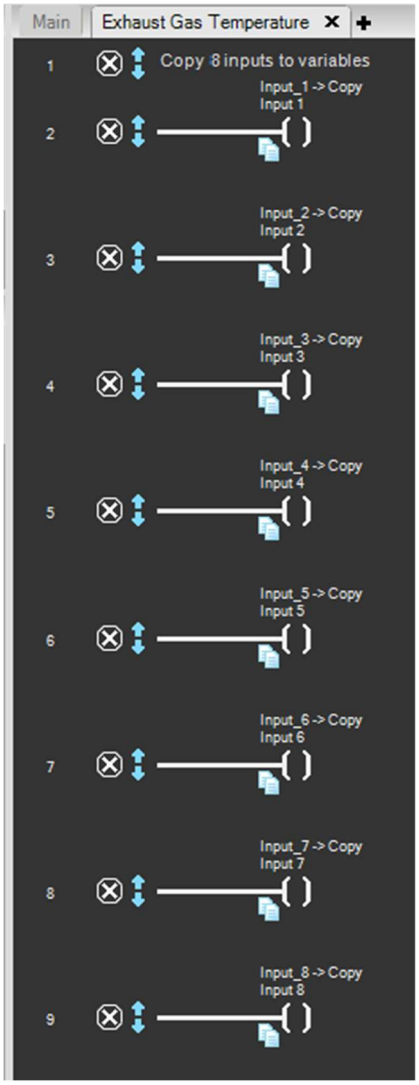
The second calling of the User Defined Function is measuring the second bank of exhaust gas temperatures using expansion 2133 ID 1

Example 3

To use multiple Mathematical functions in the same re-usable *User Defined Function* requires a different technique. Within the *User Defined Function,* all the inputs must first be copied into unique registers.

In the copy function properties box the *Value* must be set as a Function Input.





This example shows Input_1 copied to a *Variable* called *Copy Input 1*

This example shows Input_2 copied to a *Variable* called *Copy Input 2*

This example shows Input_3 copied to a *Variable* called *Copy Input 3*

This example shows Input_4 copied to a *Variable* called *Copy Input 4*

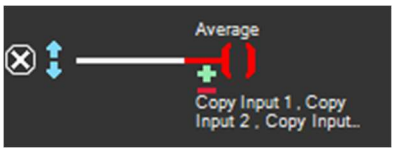
This example shows Input_5 copied to a *Variable* called *Copy Input 5*

This example shows Input_6 copied to a *Variable* called *Copy Input 6*

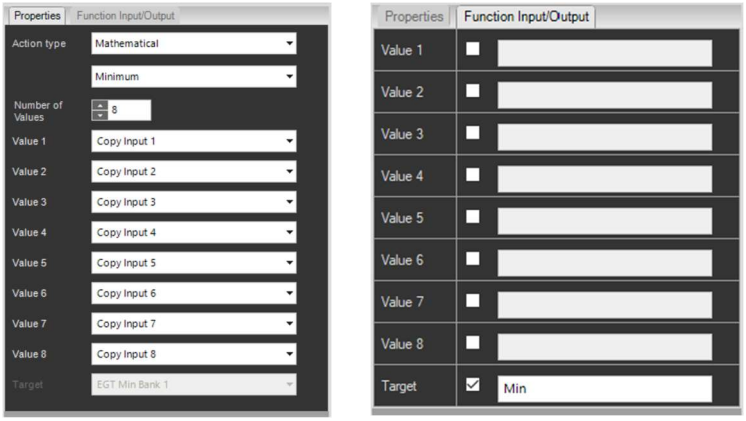
This example shows Input_7 copied to a *Variable* called *Copy Input 7*

This example shows Input_8 copied to a *Variable* called *Copy Input 8*

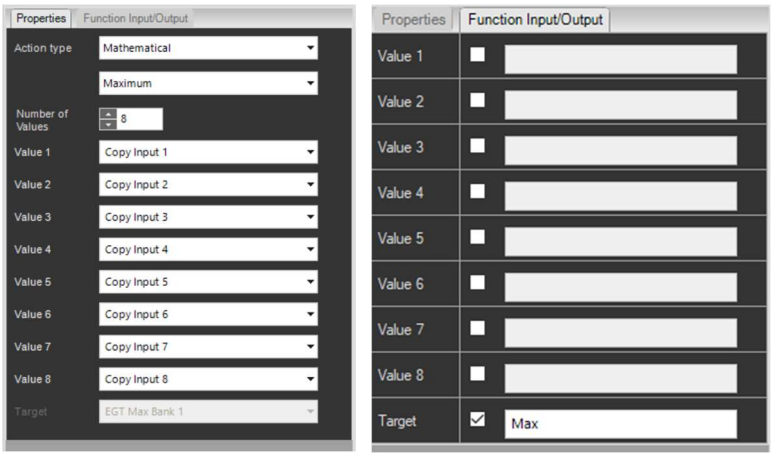
Using these 8 inputs drag the Mathematical function and configure to Average. The *Target* must be configured as a *Function Output.*



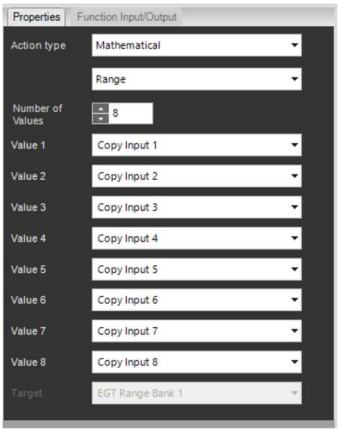
Using these 8 inputs drag the Mathematical function and configure to Minimum. The *Target* must be configured as a *Function Output.*

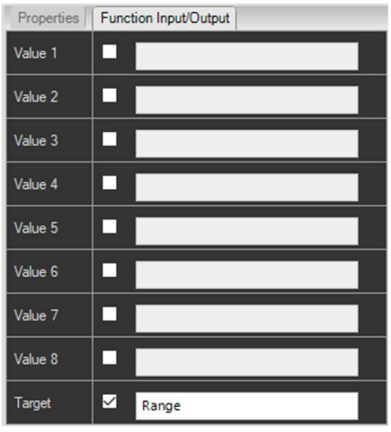


Using these 8 inputs drag the Mathematical function and configure to Maximum. The *Target* must be configured as a *Function Output.*

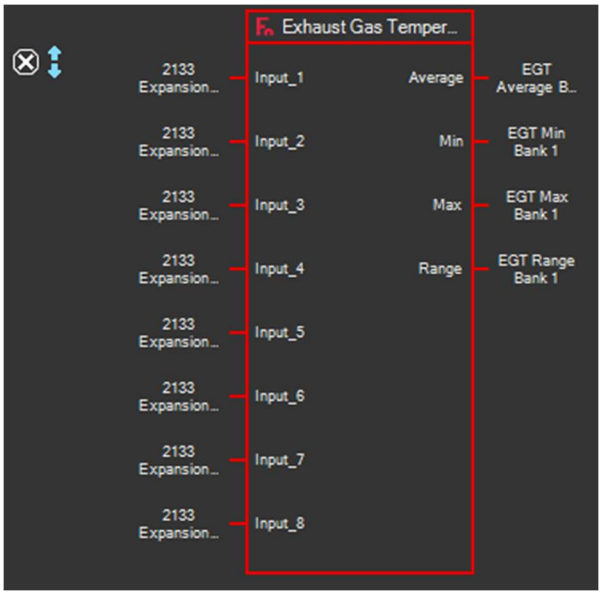


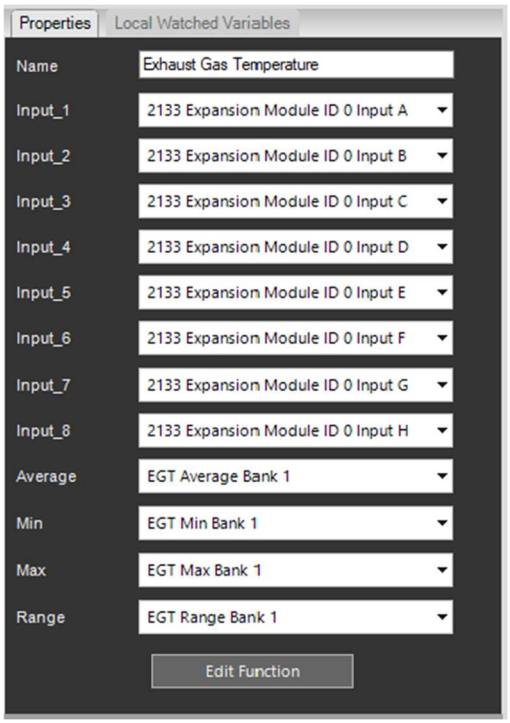
Using these 8 inputs drag the Mathematical function and configure to Range. The *Target* must be configured as a *Function Output.*



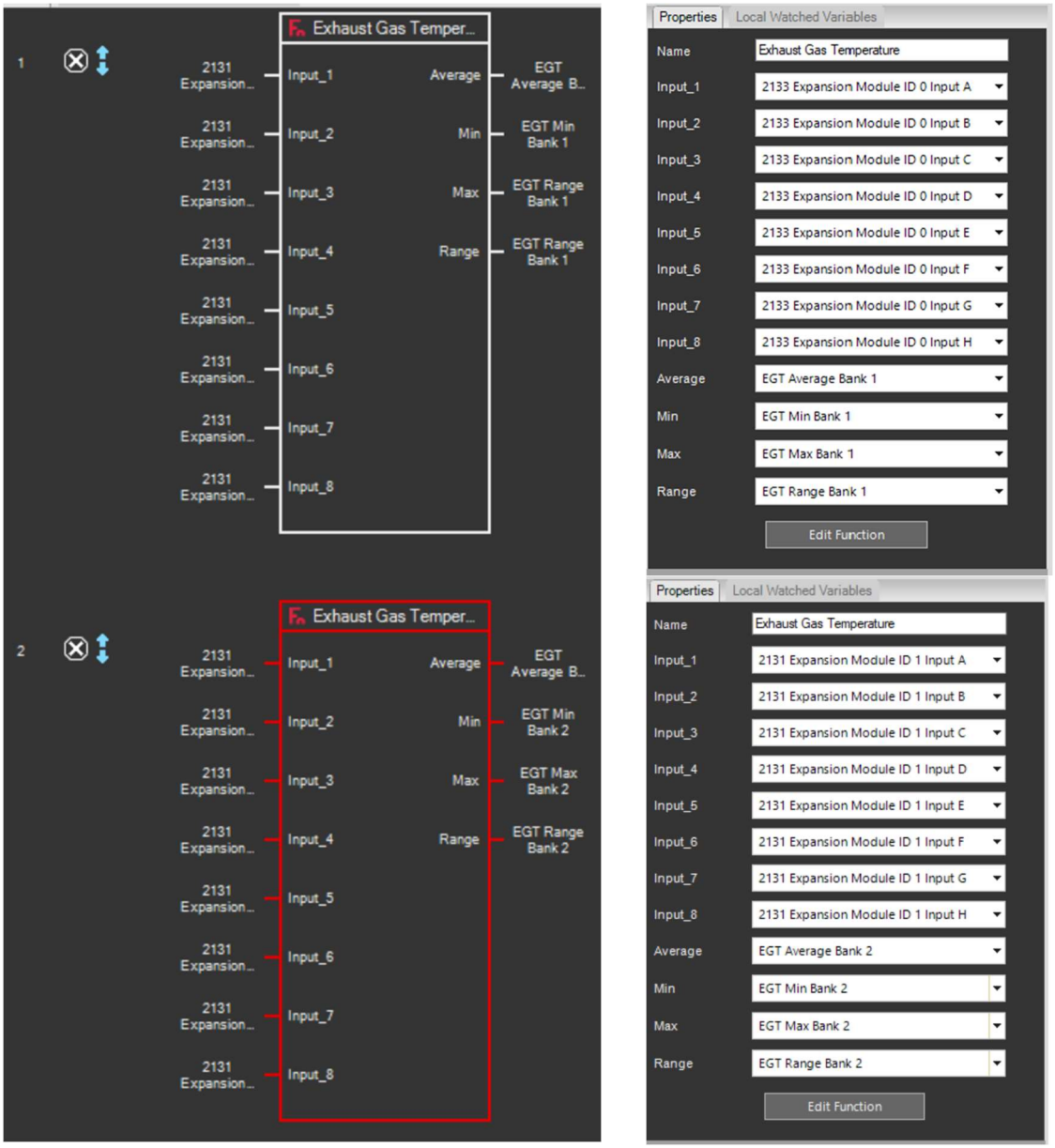


The representation on the main tab has now changed a little to include the extra 3 outputs





It is now possible to drag the *User Defined Function* again into the Main Tab of the PLC editor and set the inputs and outputs to different instruments and *variables*.



The function runs twice, working out the average, minimum value, maximum value, and the range of different sets of data.

Double clicking either box or clicking either of the *Edit Function* labels opens the common function ladder diagram.

The list of *variables* used in this example

Variables

Type ID	Name	Type	Scope	Watch	Initial Value
1	EGT Average Bank 1	Variable	Global	Not Watched	0
2	EGT Average Bank 2	Variable	Global	Not Watched	0
3	Copy Input 1	Variable	Local - Exhaust Gas Temperature	Not Watched	0
4	Copy Input 2	Variable	Local - Exhaust Gas Temperature	Not Watched	0
5	Copy Input 8	Variable	Local - Exhaust Gas Temperature	Not Watched	0
6	Copy Input 7	Variable	Local - Exhaust Gas Temperature	Not Watched	0
7	Copy Input 6	Variable	Local - Exhaust Gas Temperature	Not Watched	0
8	Copy Input 5	Variable	Local - Exhaust Gas Temperature	Not Watched	0
9	Copy Input 4	Variable	Local - Exhaust Gas Temperature	Not Watched	0
10	Copy Input 3	Variable	Local - Exhaust Gas Temperature	Not Watched	0
11	EGT Max Bank 1	Variable	Global	Not Watched	0
12	EGT Range Bank 2	Variable	Global	Not Watched	0
13	EGT Range Bank 1	Variable	Global	Not Watched	0
14	EGT Min Bank 2	Variable	Global	Not Watched	0
15	EGT Min Bank 1	Variable	Global	Not Watched	0
16	EGT Max Bank 2	Variable	Global	Not Watched	0

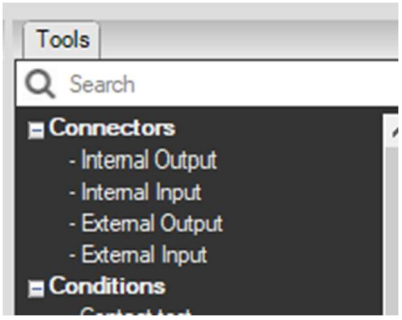
Variables 24 / 100 Persistent Variables 0 / 100 Timers 0 / 50 Strings 0 / 50 Counters 0 / 50 Coils 0 / 100 Add Remove

Consider making all the copy *variables* local to this function so that they can be watched individually. *Local Variables* cannot be used in different *User Defined Functions*.

Whilst there are only 16 *Variables* listed, all the *Copy Input #* registers are being used twice and are counted as such. Hence 24 *Variables* are in use.

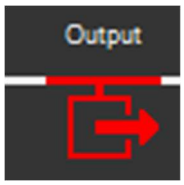
3.3.4.3.2 EXTERNAL OUTPUT

When using *User Defined Function Blocks, External Output* appears under *Connectors* in the *Tools* Menu

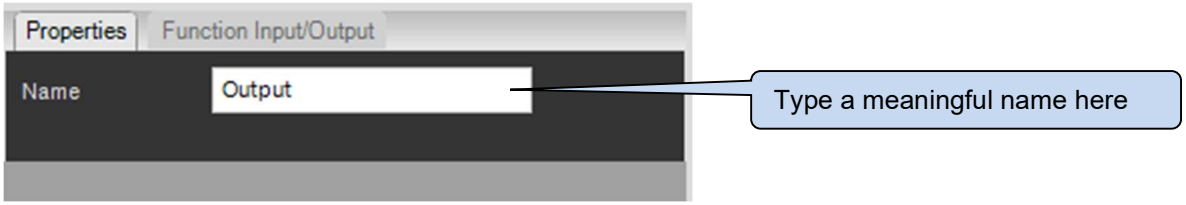


Allows a rung logic to be passed out of a *User Defined Function Block*. Adding one of these to the edited function tab, adds an output to the function.

Its symbol is shown below



Drag the *External Output* tool into the relevant place on the ladder rung *in a user defined function* and configure its properties.

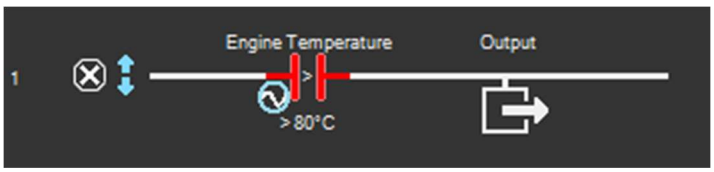


Parameter	Description
Name	The external output's name. Rename to a meaningful name

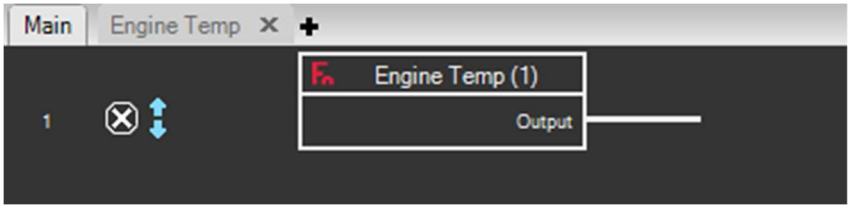
Example

The below example passes the result of an instrument test out to the main PLC program.

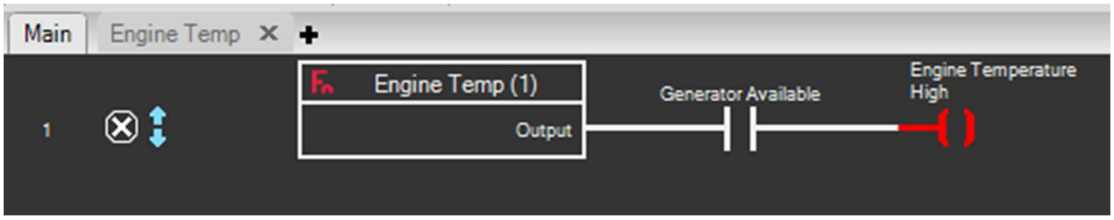
First create a *User defined function block* described in section 3.3.4.3 this manual. Within this function add an instrumentation value and configure it as required. This example is using engine temperature above 80°C. Add an external output after the instrumentation and give it meaningful name. This example uses the default name Output



Switch to the Main tab, the *User defined function block now has an output to it,*

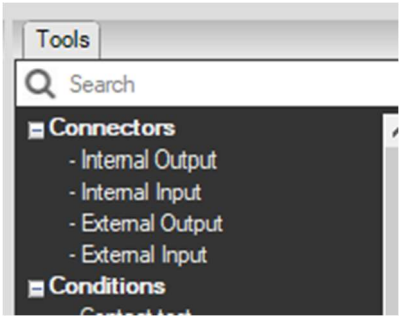


where extra logic can be added.



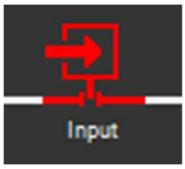
3.3.4.3.3 EXTERNAL INPUT

When using *User Defined Function Blocks External Input* appears under *Connectors* in the *Tools* Menu

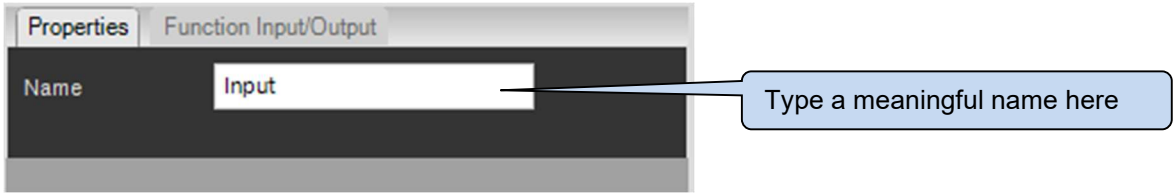


External Input allows a rung logic to be passed into a *User Defined Function Block*. Adding one of these to the edited function tab, adds an input to the function.

Its symbol is shown below



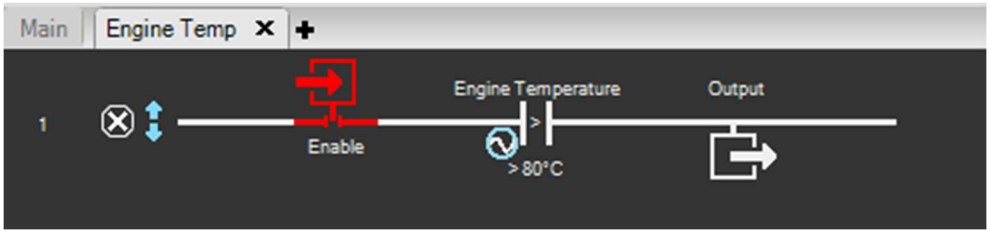
Drag the *External Input* tool into the relevant place on the ladder rung *in a user defined function block* and configure its properties.



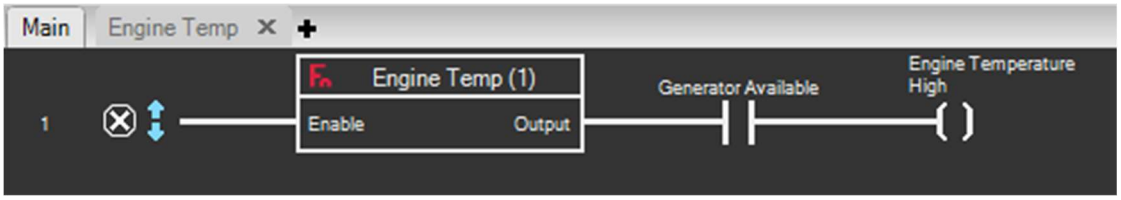
Parameter	Description
Name	The external input's name. Rename to a meaningful name

Example

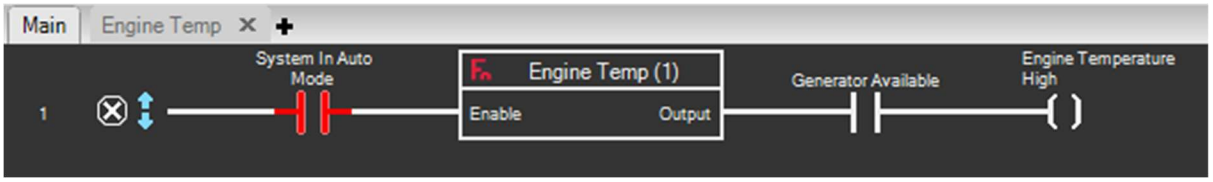
Continuing to use the example of an External Output, add an external input before the instrumentation and give it meaningful name. This example uses Enable.



Switch to the Main tab, the *User defined function block now has an input to it,*



where extra logic can be added.

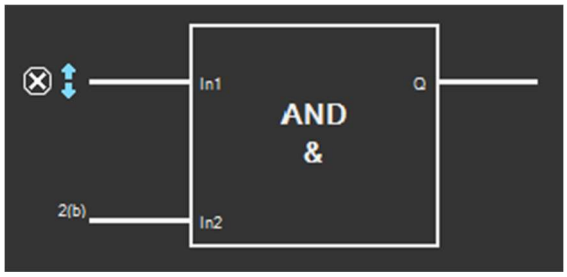


3.3.4.4 LOGIC

3.3.4.4.1 AND

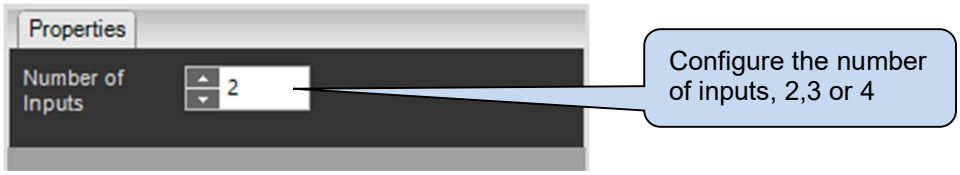
The output of an AND gate is true when all its inputs are true.

Its symbol is shown below



Q (Output) is True when both In1 and In2 are True

Drag the *AND* tool into the relevant place on the ladder rung and configure its properties.

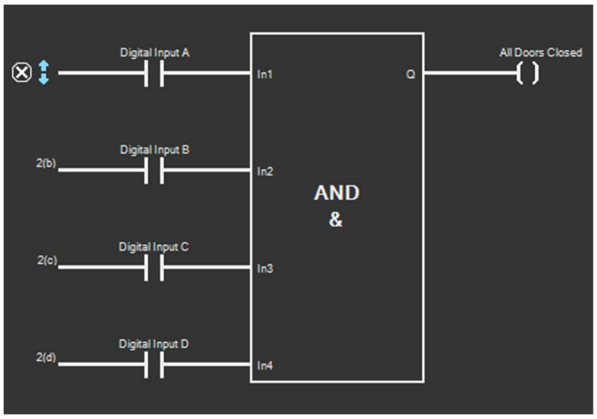


Example

The below example shows an *AND gate configured for 4 inputs*.

If *Digital Input A, Digital Input B, Digital Input C* and *Digital Input D* are all True at the same time *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named All Doors Closed is driven to True.

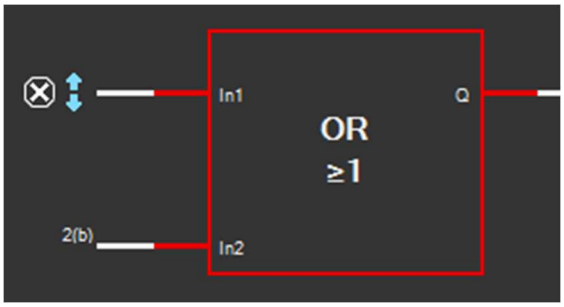
If any one of the inputs is not true, *Q (Output)* is driven to False. As *Q (Output)* is driven to False, the *Coil* named All Doors Closed is driven to be False.



3.3.4.4.2 OR

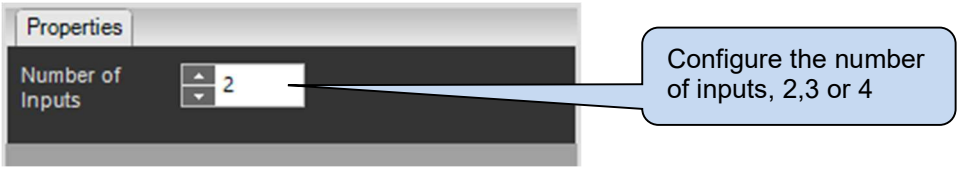
The output of an OR gate is true when any of its inputs are true. The output is also true if there are multiple inputs true

Its symbol is shown below



Q (Output) is True when either In1 or In2 are True. Q (Output) is also True when all inputs are True

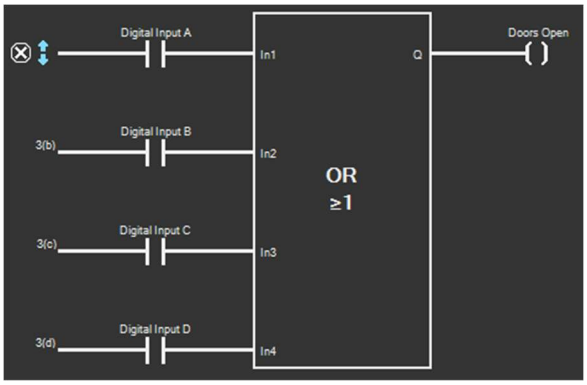
Drag the *OR* tool into the relevant place on the ladder rung and configure its properties.



Example

The below example shows an *OR gate configured for 4 inputs*.

If *Digital Input A,* or *Digital Input B,* or *Digital Input C* or *Digital Input D* is True, *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named Doors Open is driven to True. If more than 1 input is True, *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named Doors Open is driven to True



3.3.4.4.3 NOT

The output of a NOT gate is the inverse of its input. So, if the input is True the output is False. If the input is False, the output is True

Its symbol is shown below

Drag the *NOT* tool into the relevant place on the ladder rung. It does not have any properties.

Example

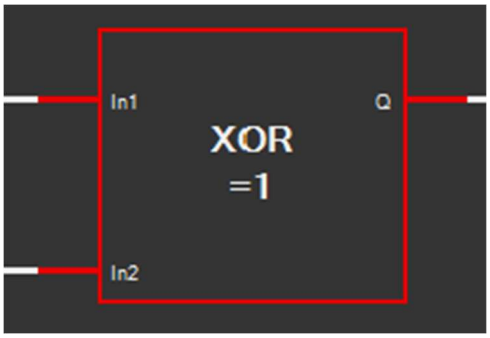
The below example shows a *NOT gate*.

If *Engine Temperature* is below 65°C In1 (*Input*) is True, *Q (Output)* is driven to False. As *Q (Output)* is driven to False, the *Coil* named Temp High is driven to False. If *Engine Temperature* is 65°C or higher In1 (*Input*) is False, *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named Temp High is driven to True

3.3.4.4.4 XOR

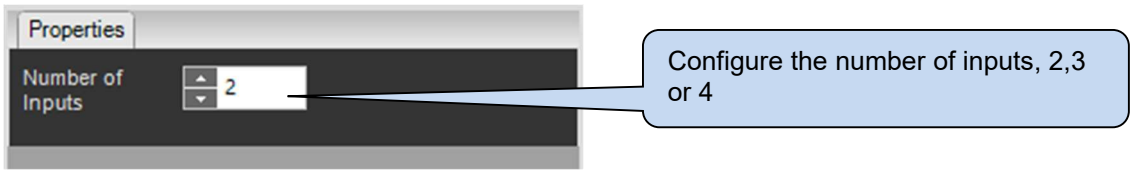
The output of an XOR gate is True when only one of its inputs is True. The output is False if there are multiple inputs True

Its symbol is shown below



Q (Output) is be True when either In1 or In2 are True. Q (Output) is False when both inputs are True

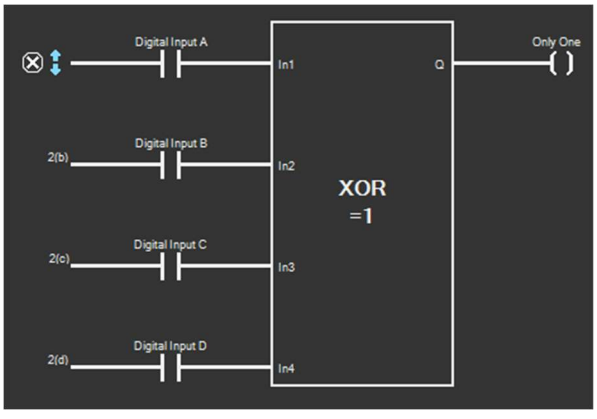
Drag the *XOR* tool into the relevant place on the ladder rung and configure its properties.



Example

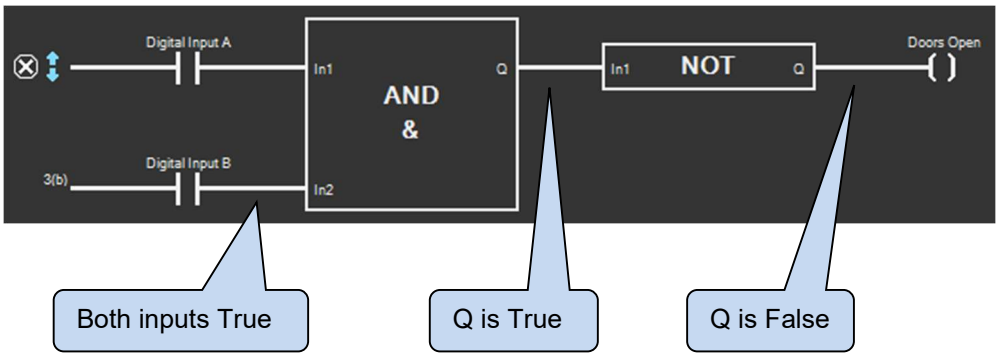
The below example shows an X*OR gate configured for 4 inputs*.

If *Digital Input A,* or *Digital Input B,* or *Digital Input C* or *Digital Input D* is True *Q (Output)* is driven to True. As *Q (Output)* is driven to True, the *Coil* named Only One is driven to True. If more than one input is True, *Q (Output)* is driven to False. As *Q (Output)* is driven to False, the *Coil* named Only One is driven to False

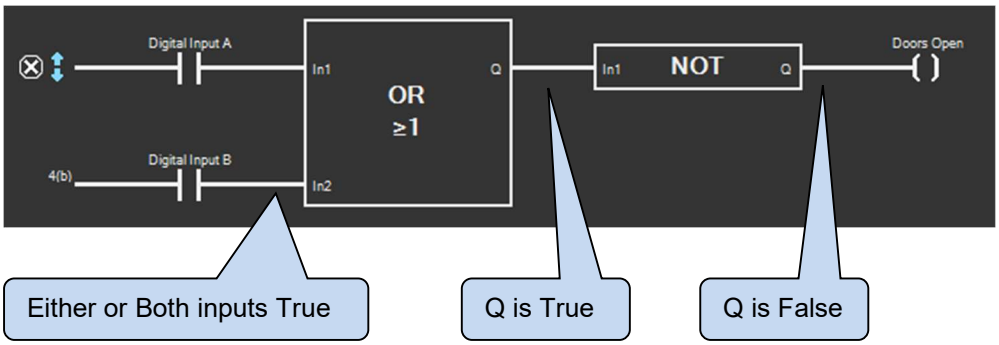


3.3.4.4.5 COMBINING LOGIC GATES

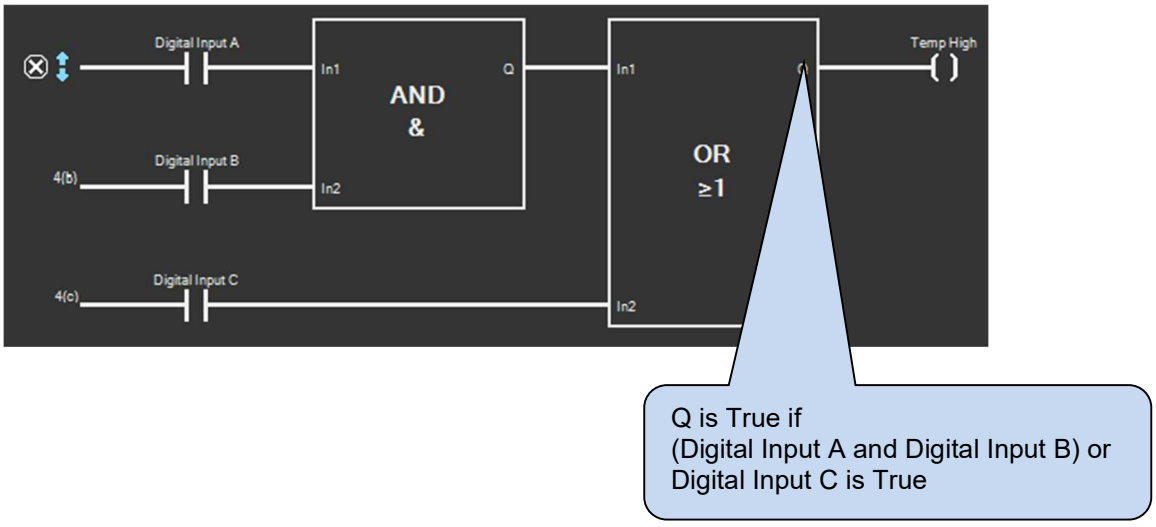
To create a NAND gate, combine an AND with a NOT gate



To create a NOR gate, combine an OR with a NOT gate



Combine an AND with an OR



4 HOW TO CREATE A PLC PROGRAM

4.1 DESIGNING THE PROGRAM

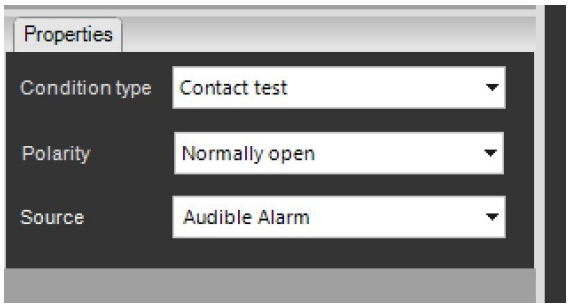
The fundamental issue in creating a PLC program is knowing exactly what is required!
It is usually best to start with a written description. As the DSE PLC is based around the internal system of inputs and outputs, keep these functions in mind when designing the solution.

4.2 USING THE PLC EDITOR TO MAKE AN EXAMPLE PROGRAM

For example, if a function is required to silence the module's alarm 30 seconds after it begins, the designer must consider how to determine if the alarm is active and then how to perform the silencing function.

Looking through the list of *Coils* we find *Audible Alarm.* This flag is set whenever the module's internal audible alarm is active.

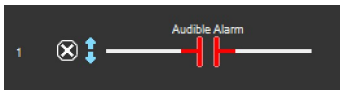
Therefore, in the *PLC Logic* screen we can 'drag' the Contact Test icon to the Program Area, and select the appropriate source in the properties box to the right of the program area



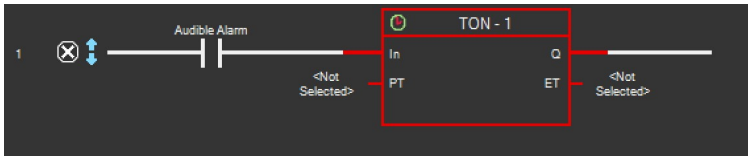
Parameter	Description
Condition Type	The type of condition to check for. This is automatically set when the icon is dragged from the Condition toolbar but can be changed if the wrong one is inadvertently selected.
Polarity	***Normally Open***: Tests if the condition is True. ***Normally Closed***: Tests if the condition is False.
Source	Contains the list of available Coils. This list differs between module types. A full list of sources is included in the relevant module's Configuration Suite manual.

In our example we select *Contact Test*, *Normally Open and Audible Alarm*.

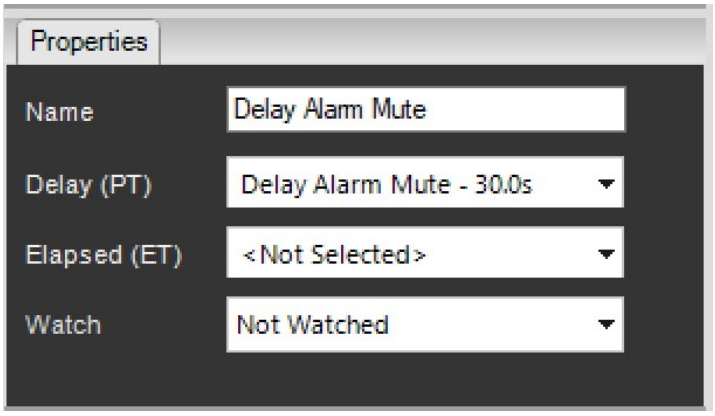
Our example program is now as below. If it is not, then go back and check what went wrong.



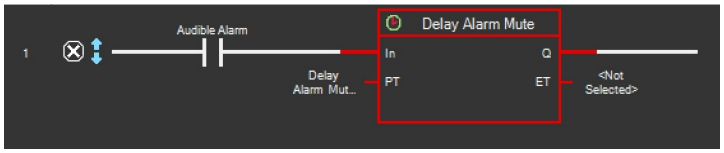
The next thing we need to do is set a timer that expires in 30 seconds after the audible alarm begins. Drag the Timer On icon from the *Function Block Toolbar* and drop it just to the right of the white line to the right of the *Audible Alarm* symbol that has just been placed.



When the timer has been placed, complete the properties box

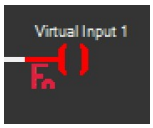


Parameter	Description
Name	Give the timer a meaningful name.
Delay (PT)	The timer setting. Selected from the variables table
Elapsed (ET)	The current running time of the timer. Selected from the variables table. This can be left as not selected
Watched	Select whether the timer is watched or not. Refer to section entitled *Watched Items* else where in this document.

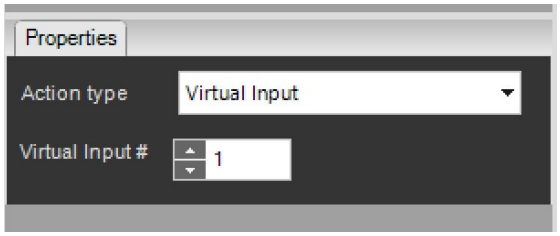


This program now checks for an audible alarm and starts a timer that runs for 30 seconds (so long as the audible alarm remains active). Next, we need to configure what happens when the timer expires.

Drag a virtual input just to the right of the white line to the right of the timer

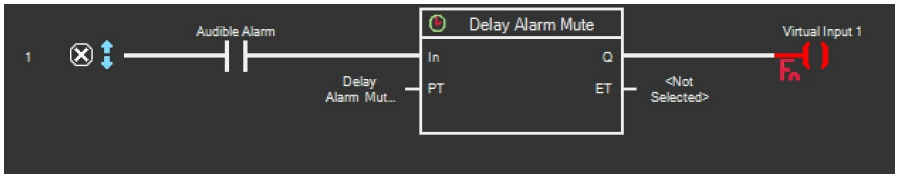


Complete the properties box



Parameter	Description
Action Type	The type of action. This is automatically set when the icon is dragged from the Action toolbar but can be changed if the wrong one is inadvertently selected.
Virtual Input	The number of the virtual input to be used

Our program is now as shown below:

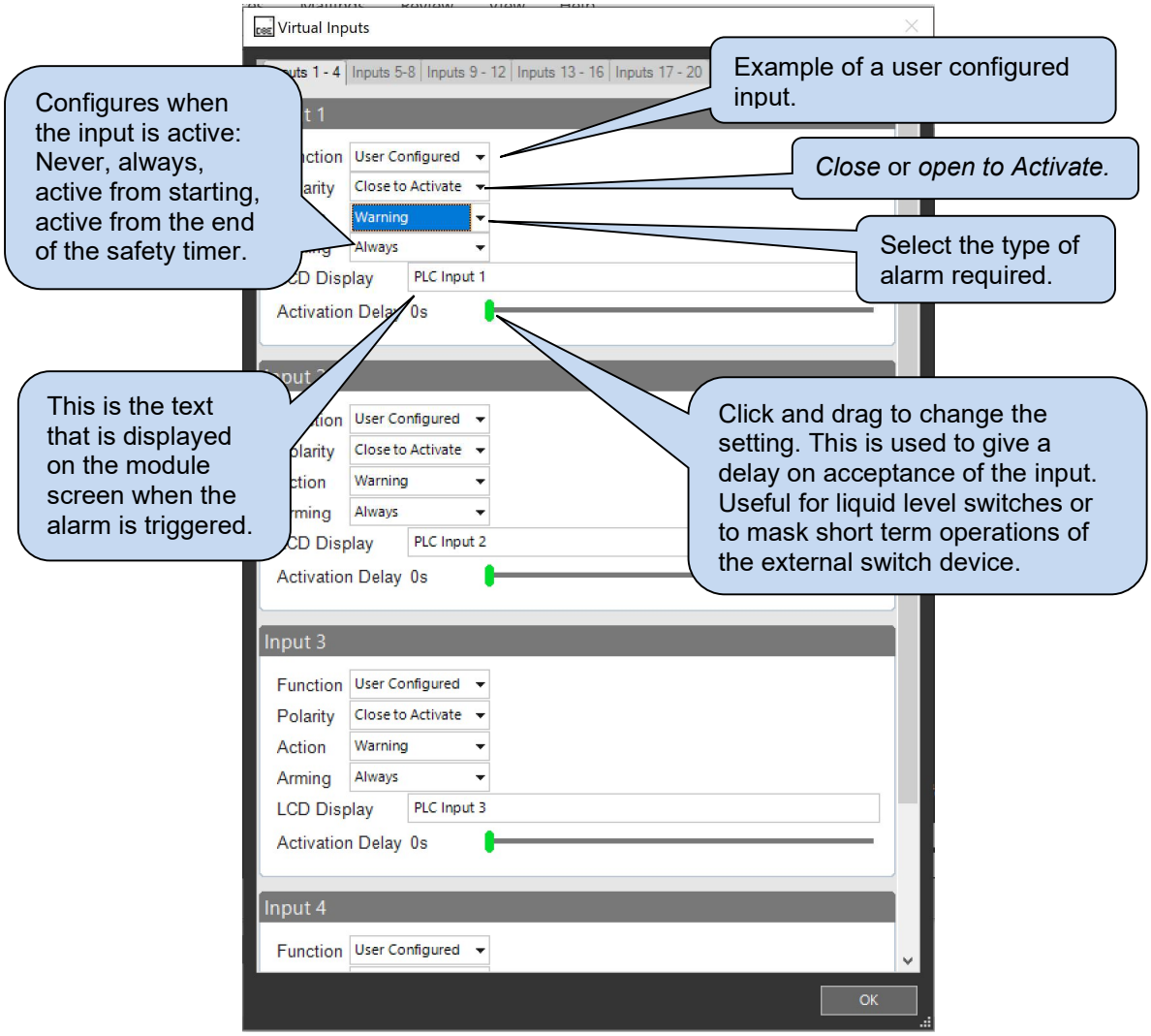


For our example, select *Virtual Input # 1*.

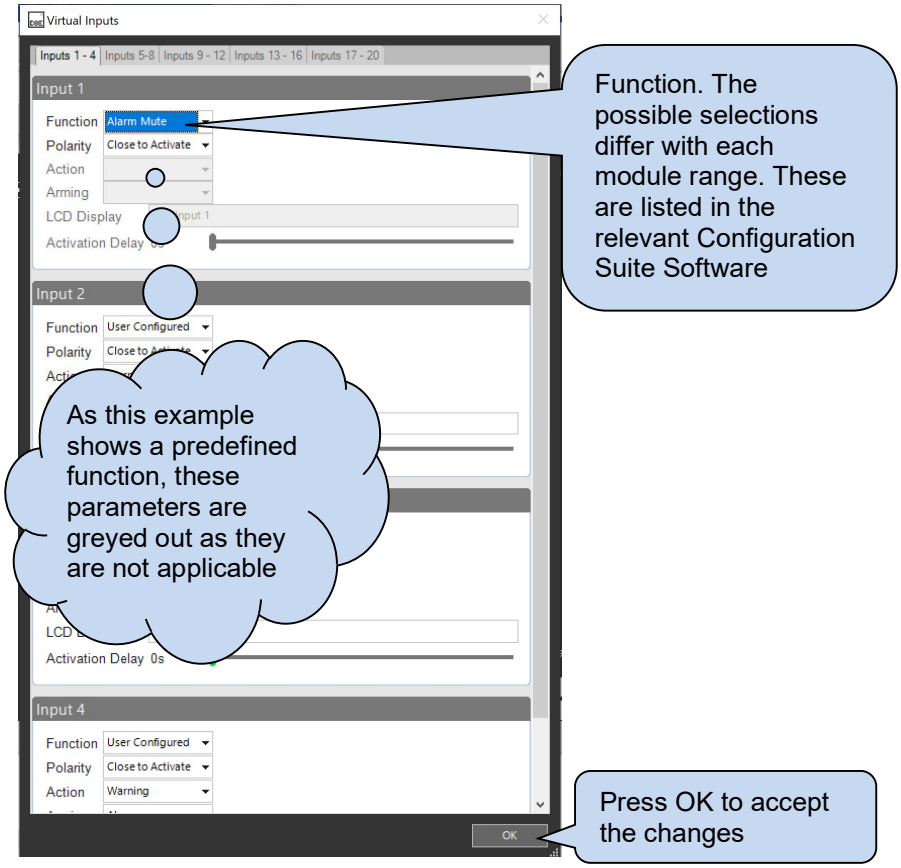
This program now checks for an audible alarm and starts a timer that runs for 30 seconds (so long as the audible alarm remains active).
When the timer expires, Virtual Input 1 is triggered.

Finally, we must configure what Virtual Input 1 is used for.

Select the Virtual Inputs Icon  which opens the virtual input settings window. PLC Functions have exactly the same choices as module digital inputs.



For our example PLC program, we need to select *Alarm Mute*, a predefined function that silences the alarm:



This now results in a program that checks for an audible alarm and starts a timer that runs for 30 seconds (so long as the audible alarm remains active).
When the timer expires, Alarm Mute (Virtual Input 1) is triggered silencing the alarm. If another alarm occurs, the audible alarm restarts, starting our 30 second timer again.

Remember to click (Save) to save a copy of the configuration file. The PLC program is contained within the configuration file.
Also remember to click (Write to module) to upload the configuration file to the connected module.

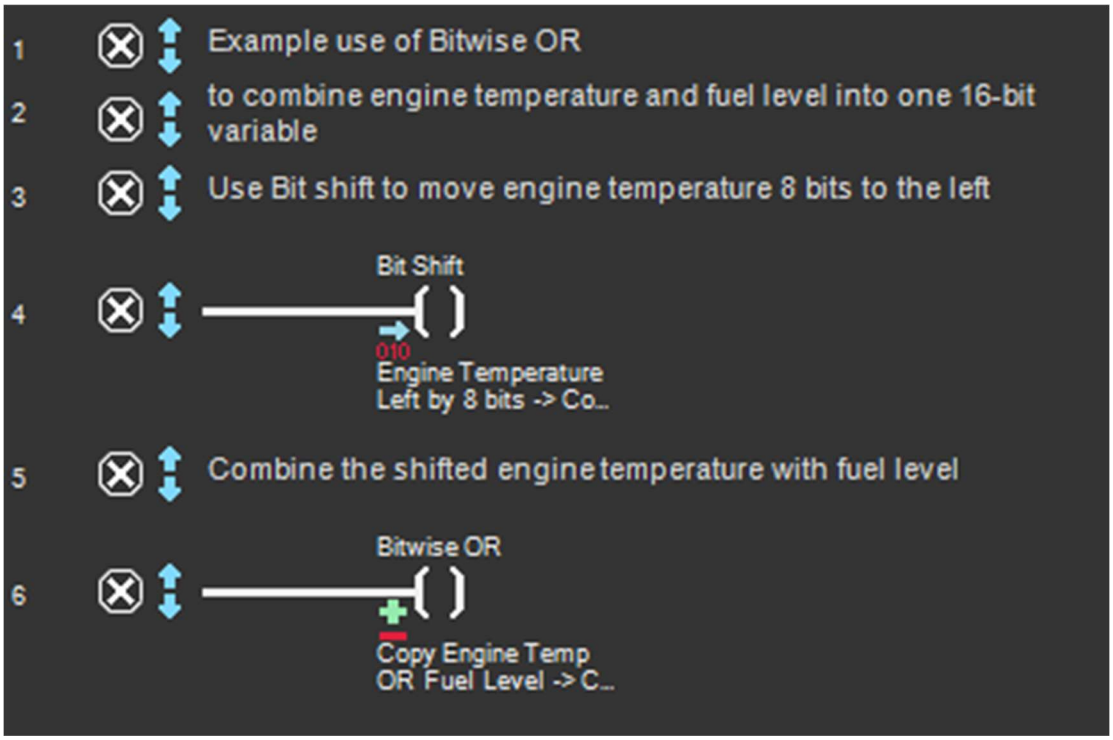
4.3 EXAMPLE PLC PROGRAMS

4.3.1 USING BITWISE FUNCTIONS

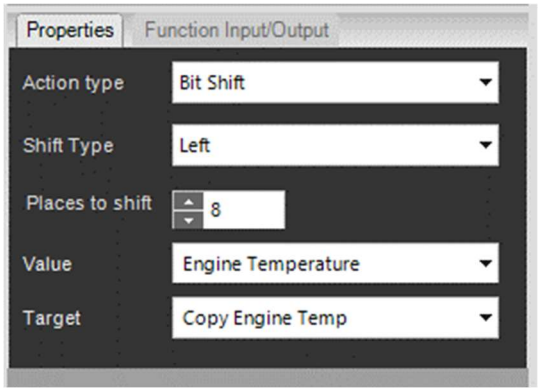
Example 1

Bitwise Or to combine two separate instruments into one variable

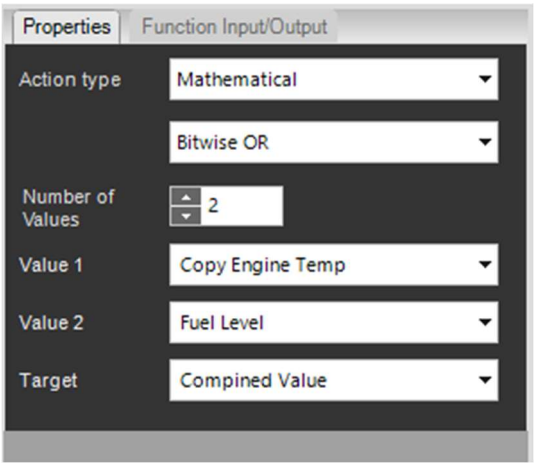
This example shows how to combine engine temperature and fuel level into one variable.



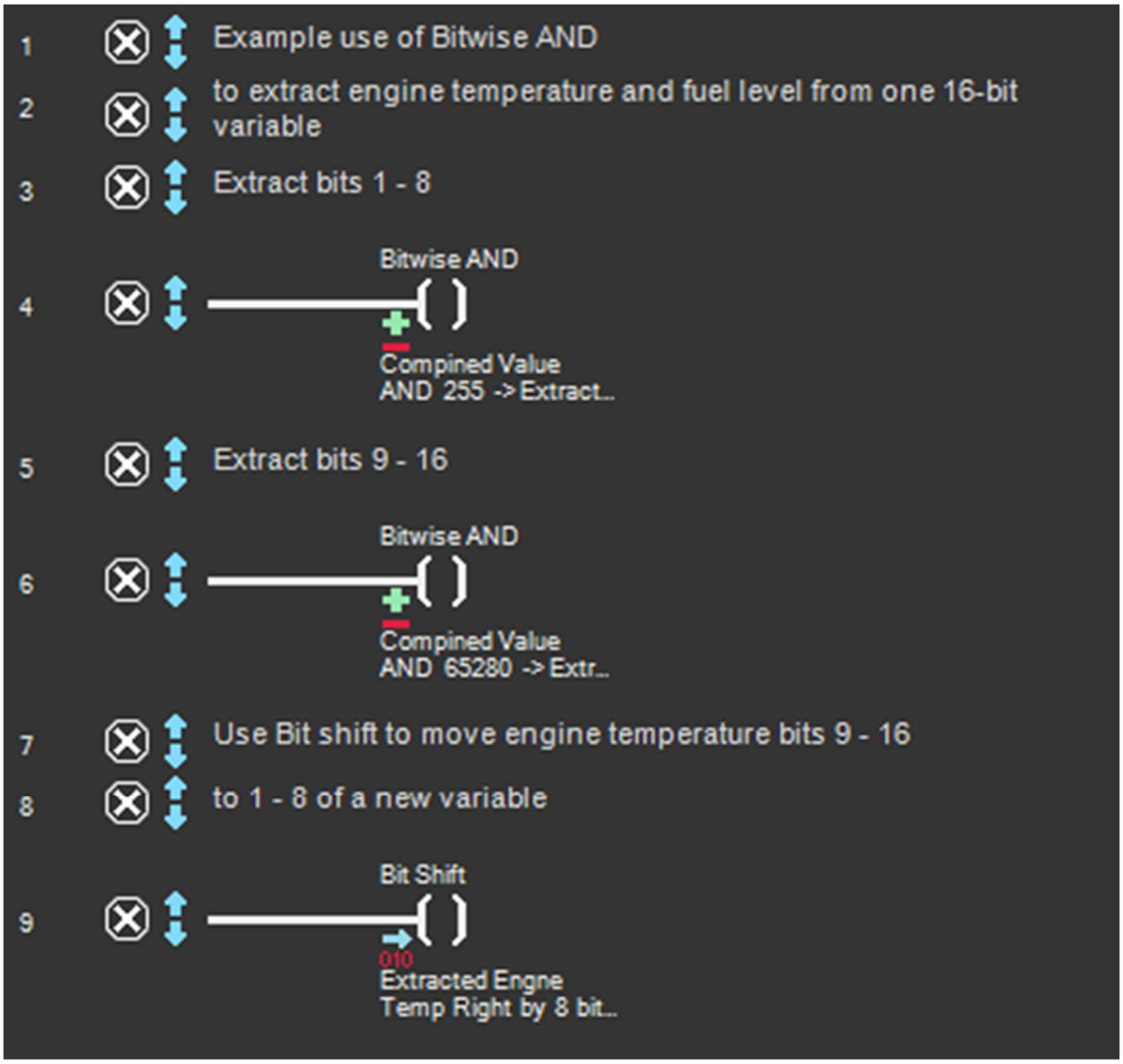
Properties For Bit Shift



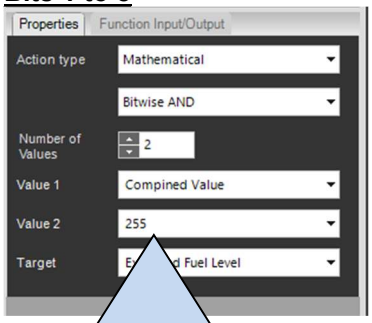
Properties For Bitwise Or



This example shows how to extract engine temperature and fuel level from one variable.

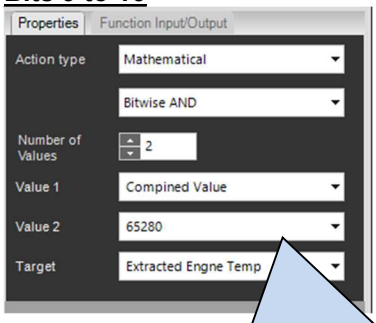


Properties For Bitwise AND Bits 1 to 8



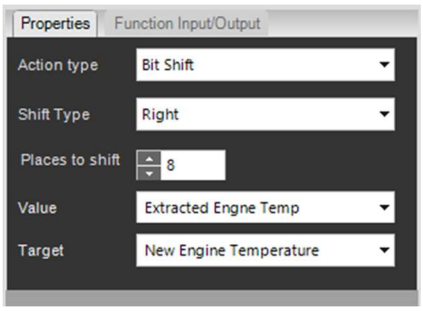
255 isolates bits 1 to 8. The binary for 255 is 0000 0000 1111 1111

Properties for Bitwise AND Bits 9 to 16



65280 isolates bits 9 to 16. The binary for 65280 is 1111 1111 0000 0000

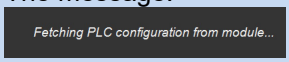
Properties For Bit Shift



5 TESTING AND DIAGNOSING A PLC PROGRAM

Many PLC programs are very simple, having only one or two rungs (lines) in the ladder. Often, the easiest way to test these is to simply use the module and test that the actions are as required. However sometimes this leads to us finding that the PLC program is not acting as required. This is usually caused by an error in the design of the program.
DSE have provided a diagnostic monitor to aid the fault-finding process.

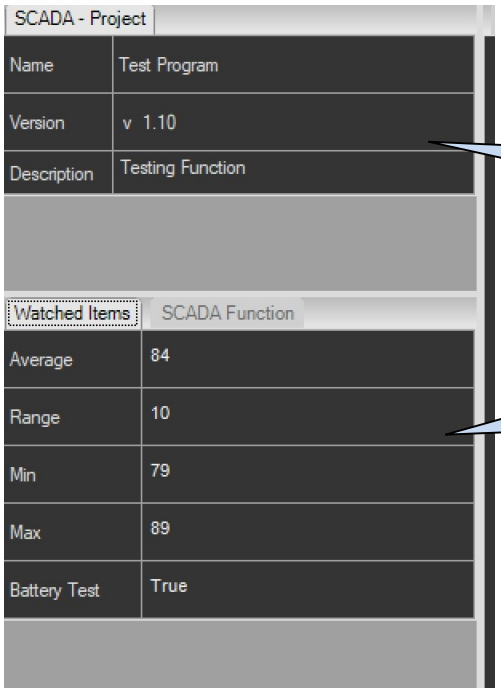
Select *Connect* SCADA from the PLC Tool Bar of the PLC editor. when connected to the module containing the PLC ladder to be viewed
The message:



is shown until the entire PLC ladder has been downloaded for viewing from the connected module.

This icon at the bottom of the window indicates that the PLC SCADA is not connected to a module

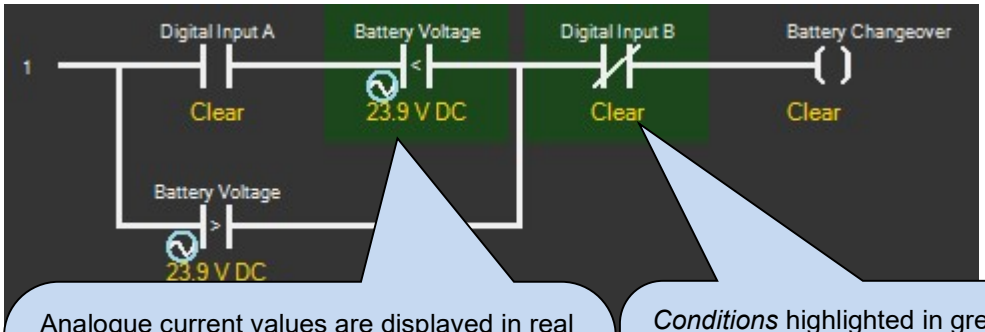
This icon at the bottom of the window indicates that the PLC SCADA is not connected to a module



Once the configuration has been loaded. The project box is populated.

Any watched items are displayed on the left-hand side of the SCADA viewer.

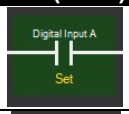
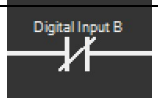
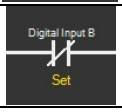
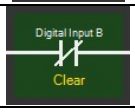
Example showing PLC SCADA:



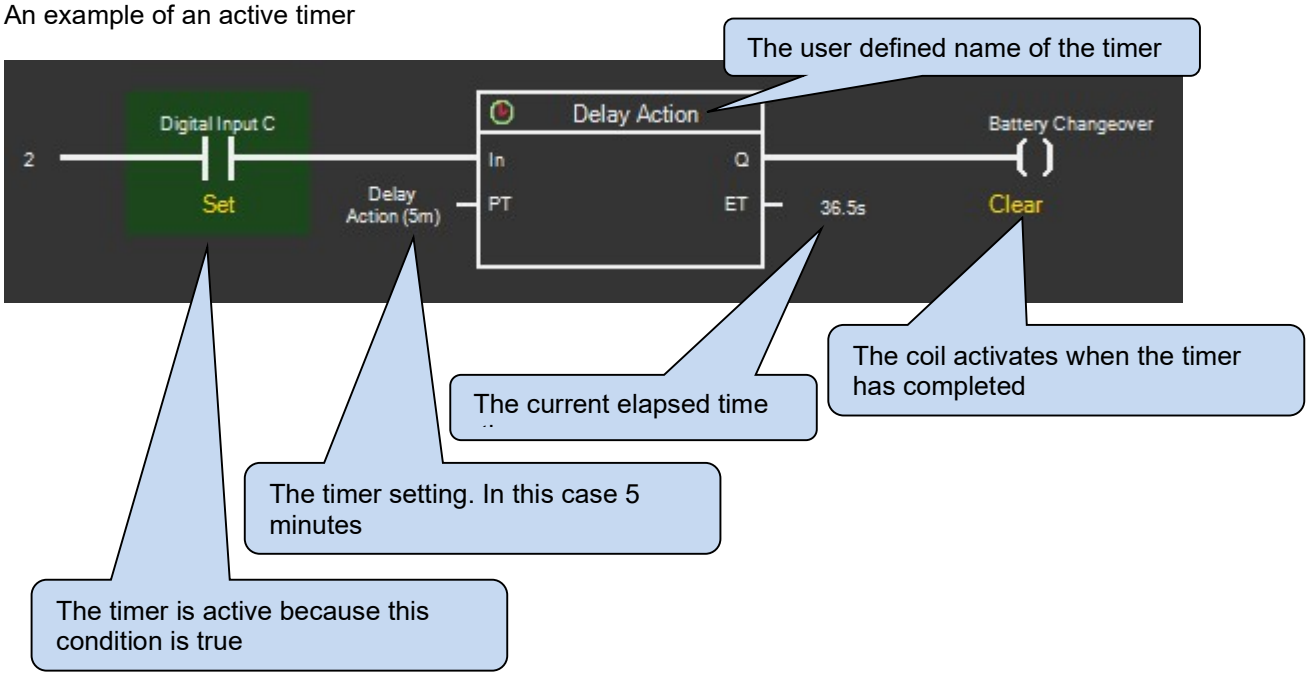
Analogue current values are displayed in real time. As do counters' timers, variables and persistent variables

Note that depending upon update time of the SCADA screen, this may not be in 'real time'. Fast connection methods such as USB or direct Ethernet connection (no internet) give much quicker update rates.

Conditions highlighted in green show ACTIVE (true) conditions.
Conditions with no highlight are INACTIVE (false)
'*Clear*' means that the coil is not set.
'*Set*' means that the coil is set.
See below for further details.

PLC Editor Symbol	Description	PLC SCADA Symbol When Active (True)	PLC SCADA Symbol When Inactive (False)
	Normally Open Gate		
	Normally Closed Gate		

An example of an active timer



Page is Left Intentionally Blank

Page is Intentionally Left Blank