Your Global Automation Partner

TURCK

ARGEE 2
Reference Manual

MA1019
Last Updated 7/30/2018

1 General Information

1.1 About these instructions

The following user manual describes the setup, functions, and use of the system. It helps you to plan, design, and implement the system for its intended purpose.

Note*: Please read this manual carefully before using the system. This will prevent the risk of personal injury or damage to property or equipment. Keep this manual safe during the service life of the system. If the system is passed on, be sure to transfer this manual to the new owner as well.

1.2 Explanation of symbols used

1.2.1 Warnings

Action-related warnings are placed next to potentially dangerous work steps and are marked by graphic symbols. Each warning is initiated by a warning sign and a signal word that expresses the gravity of the danger. The warnings have absolutely to be observed:

⚠ DANGER!

DANGER indicates an immediately dangerous situation, with high risk, the death or severe injury, if not avoided.

⚠ WARNING!

WARNING indicates a potentially dangerous situation with medium risk, the death or severe injury, if not avoided.

❗ ATTENTION!

ATTENTION indicates a situation that may lead to property damage, if it is not avoid-ed.

ℹ NOTE

In NOTES you find tips, recommendations and important information. The notes facilitate work, provide more information on specific actions and help to avoid overtime by not following the correct procedure.

➢ CALL TO ACTION

This symbol identifies steps that the user has to perform.

➔ RESULTS OF ACTION

This symbol identifies relevant results of steps

1.3 Contents

Enter the contents of this manual/guide.

- Overview of the ARGEE Manual content
- How to access the ARGEE Environment
- An explanation of the ARGEE Menu Bar
- A general overview and walkthrough of the ARGEE Flow Chart
- A general overview and walkthrough of ARGEE PRO
- A detailed explanation of the ARGEE Condition & Action Statements
- A detailed explanation of the Operations offered in ARGEE PRO
- A general overview and walkthrough of the ARGEE Simulation Mode
- A detailed explanation about ARGEE Security
- A general overview of ARGEE system behavior
- Sample code for many common applications
- Defining I/O Variable Names

1.4 Feedback about these instructions

We make every effort to ensure that these instructions are as informative and as clear as possible. If you have any suggestions for improving the design or if some information is missing in the document, please send your suggestions to **techdoc@turck.com**.

1.5 Technical support

For additional support, email inquiries to appsupport@turck.com, or call Application Support at 763-553-7300, Monday-Friday 8AM-5PM CST.

2 Preface

Read this preface to familiarize yourself with the rest of the manual. It provides answers to the following questions:

- Why use ARGEE?
- What are ARGEE’s advantages and limitations?
- Who should use this manual?
- What is the purpose of this manual?
- What content is in the ARGEE reference manual?

2.1 Why use ARGEE?

Imagine that a customer is trying to solve a simple application. This customer does not need a PLC, but they do need some logic. ARGEE was created specifically to solve this problem.

2.2 What are ARGEE’s advantages and limitations?

2.2.1 ARGREE Advantages

- ARGEE stands alone
  - Standalone application (No PLC needed to perform logic)
- ARGEE backs up the PLC
  - PLC back-up (If the application loses communication with the PLC, ARGEE can take over and safely shut down the process)
- ARGEE and the PLC work together
  - Local Control (ARGEE can monitor an application and send updates back to the PLC)

2.2.2 ARGEE Limitations

- One ARGEE block cannot control another ARGEE block
- ARGEE is not suited for motion applications

2.3 Who should use this manual?

Use this manual if you are responsible for designing, installing, programming or trouble shooting a Turck multiprotocol block that is using the ARGEE programmable functionality.

You should have a basic understanding of networking knowledge, Boolean algebra, and ladder logic. If you do not possess these skills, contact your local Turck representative for proper training before using ARGEE.

2.4 What is the purpose of this manual?

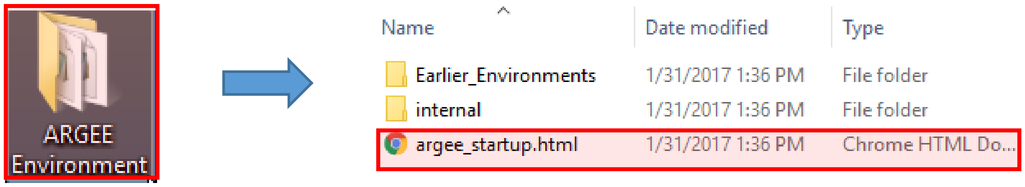
This manual is a reference guide for the ARGEE Programing Environment. This manual:

- Teaches the user how to use the ARGEE Flow Chart
- Teaches the user about syntax in ARGEE PRO
- Provides code for common applications
- Defines all the tag names associated with Turck I/O cards

3 Logging into ARGEE

3.1 Opening the Environment

➢ Open the ARGEE Environment and double click on argee_startup.html.



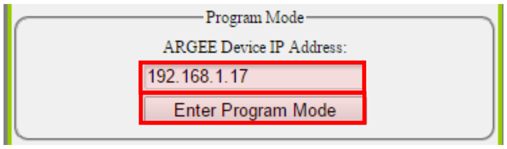
Name	Date modified	Type
Earlier_Environments	1/31/2017 1:36 PM	File folder
internal	1/31/2017 1:36 PM	File folder
argee_startup.html	1/31/2017 1:36 PM	Chrome HTML Do...

NOTE

ARGEE only opens up in HTML 5 compliant web browsers such as Google Chrome or Firefox.

3.2 Logging into the Program Mode

➢ Type **your** device's IP Address into the ARGEE Device IP Address text box, and then click Enter Program Mode.



ARGEE Device IP Address: 192.168.1.17

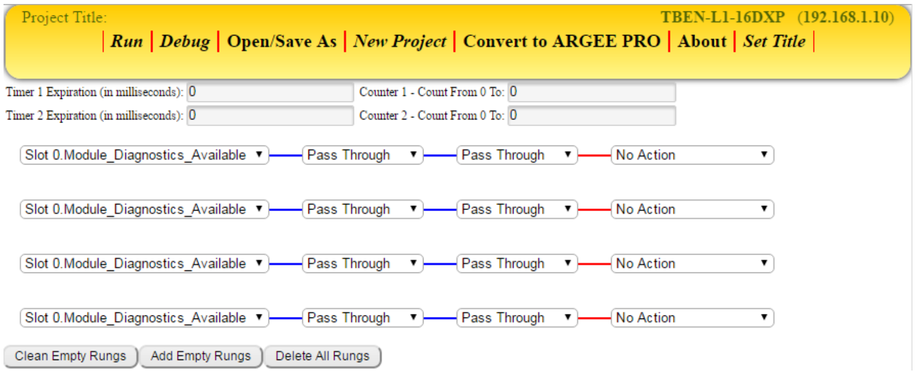
NOTE

Simulation Mode is explained later in chapter 10 ARGEE Simulation Mode.

NOTE

The user can find their device's IP Address on the block itself, located in the hatch, set by rotary dials, or by using the Turck Service Tool application.

3.3 Welcome to Flow Chart



Project Title:

TBEN-L1-16DXP (192.168.1.10)

Timer 1 Expiration (in milliseconds): 0

Counter 1 - Count From 0 To: 0

Timer 2 Expiration (in milliseconds): 0

Counter 2 - Count From 0 To: 0

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

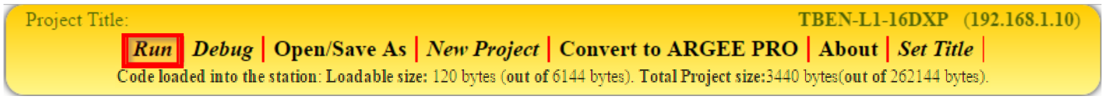
Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

4 ARGEE Menu Bar (Flow Chart)

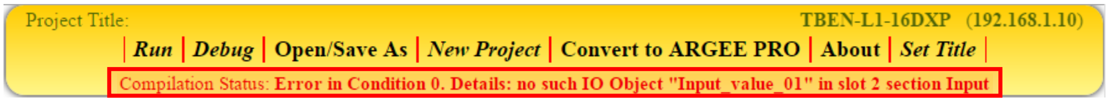
4.1 Run

When the user clicks *Run*, several things happen. First, ARGEE checks the code for errors. If the code has no errors, ARGEE downloads the code to the block. It also calculates and displays how much memory the code has used and how much memory is still available. Lastly, ARGEE transitions over to the Debug screen.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Open/Save As | New Project | Convert to ARGEE PRO | About | Set Title

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:3440 bytes(out of 262144 bytes).

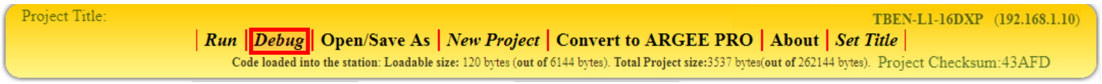
If the code has errors, ARGEE will display an error message and tell the user where the error is located in the code.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Open/Save As | New Project | Convert to ARGEE PRO | About | Set Title

Compilation Status: Error in Condition 0. Details: no such IO Object "Input_value_01" in slot 2 section Input

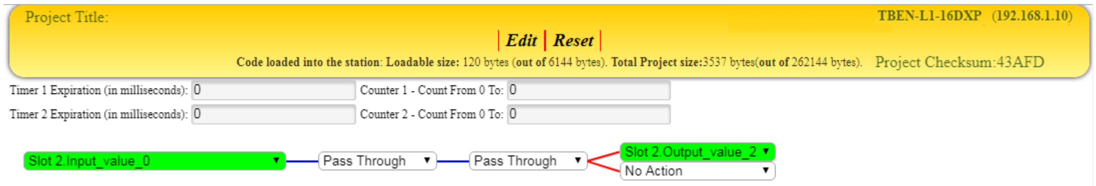
4.2 Debug (ARGEE Flow)

When the user clicks *Debug*, different things happen depending on whether the user is in Flow Chart or ARGEE PRO.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Open/Save As | New Project | Convert to ARGEE PRO | About | Set Title

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:3537 bytes(out of 262144 bytes). Project Checksum:43AFD

If the user clicks *Debug* while in Flow Chart, the first thing the user will notice is that the Flow Chart will enter *Debug* mode. As conditions become true, the user can visually observe code progression.



Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:3537 bytes(out of 262144 bytes). Project Checksum:43AFD

Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

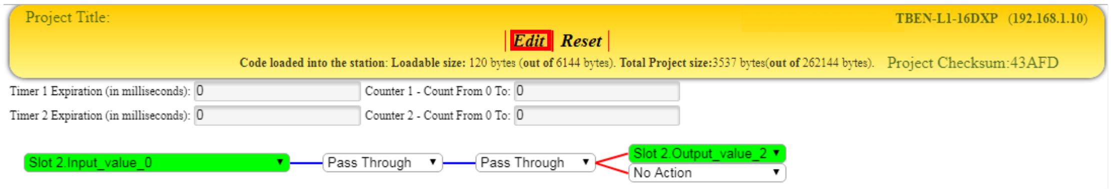
The highlighted green indicates that Slot 2 Input_value_0 on the block has gone true and has therefore triggered the Slot 2 Output_value_2 (also green) to go true.

NOTE

Debug behavior in ARGEE PRO is discussed later in section 5.2 Debug (ARGEE PRO).

4.2.1 Edit

The user can find the *Edit* tab on many screens in the ARGEE 2 Flow environment. The user will click *Edit* when they want to leave their current location and return to the ARGEE Flow programming page.



Project Title: TBEN-L1-16DXP (192.168.1.10)

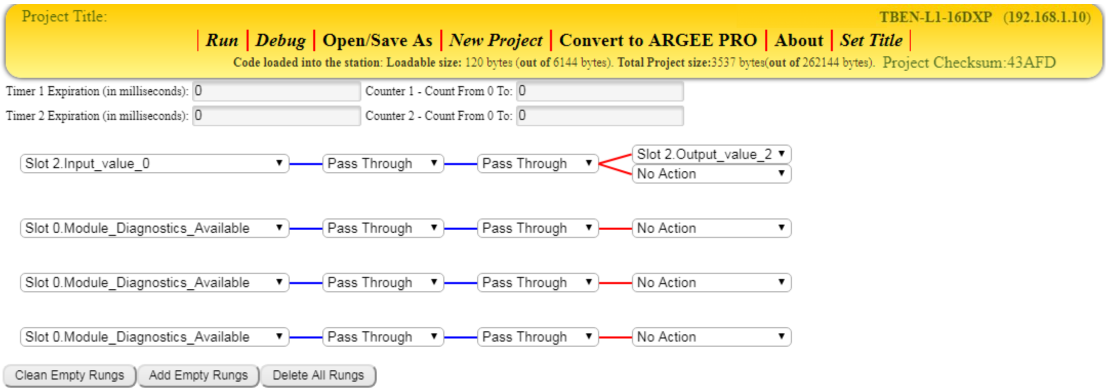
| *Edit* | *Reset* |

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:3537 bytes(out of 262144 bytes). Project Checksum:43AFD

Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 0

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

Slot 2.Input_value_0 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 2.Output_value_2 ▾ / No Action ▾



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Run* | *Debug* | Open/Save As | *New Project* | Convert to ARGEE PRO | About | *Set Title* |

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:3537 bytes(out of 262144 bytes). Project Checksum:43AFD

Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 0

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

Slot 2.Input_value_0 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 2.Output_value_2 ▾ / No Action ▾

Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

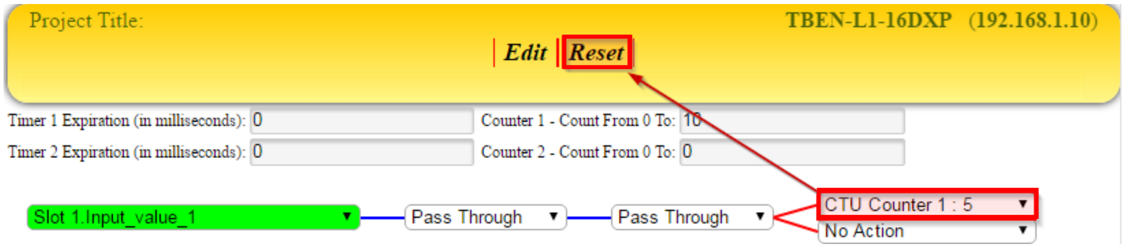
Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

Clean Empty Rungs | Add Empty Rungs | Delete All Rungs

4.2.2 Reset

The user can view the *Reset* button while in *Debug* mode or while viewing an HMI screen. Reset sets the program’s timers and counters to zero.

If the user clicks Reset while in Flow Chart, the user can visually observe the timers and counters being reset.



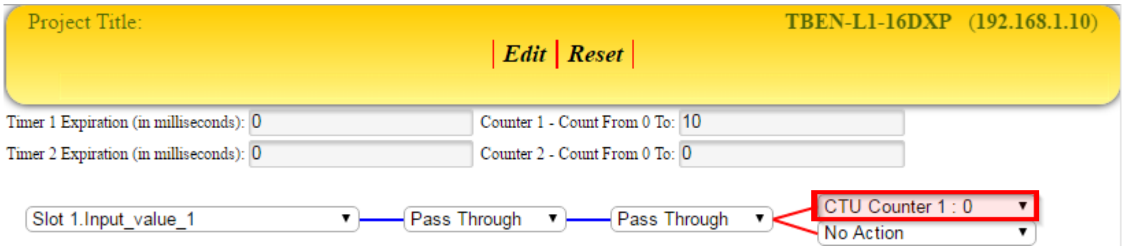
Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit* | *Reset* |

Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 10

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — CTU Counter 1 : 5 ▾ / No Action ▾



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit* | *Reset* |

Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 10

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

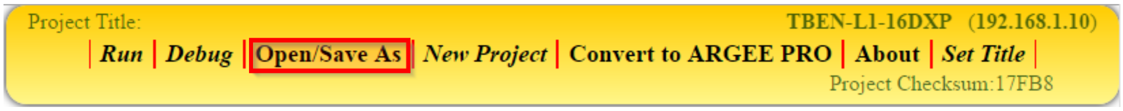
Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — CTU Counter 1 : 0 ▾ / No Action ▾

NOTE

Reset in ARGEE PRO is discussed later in section 5.2.3 Reset.

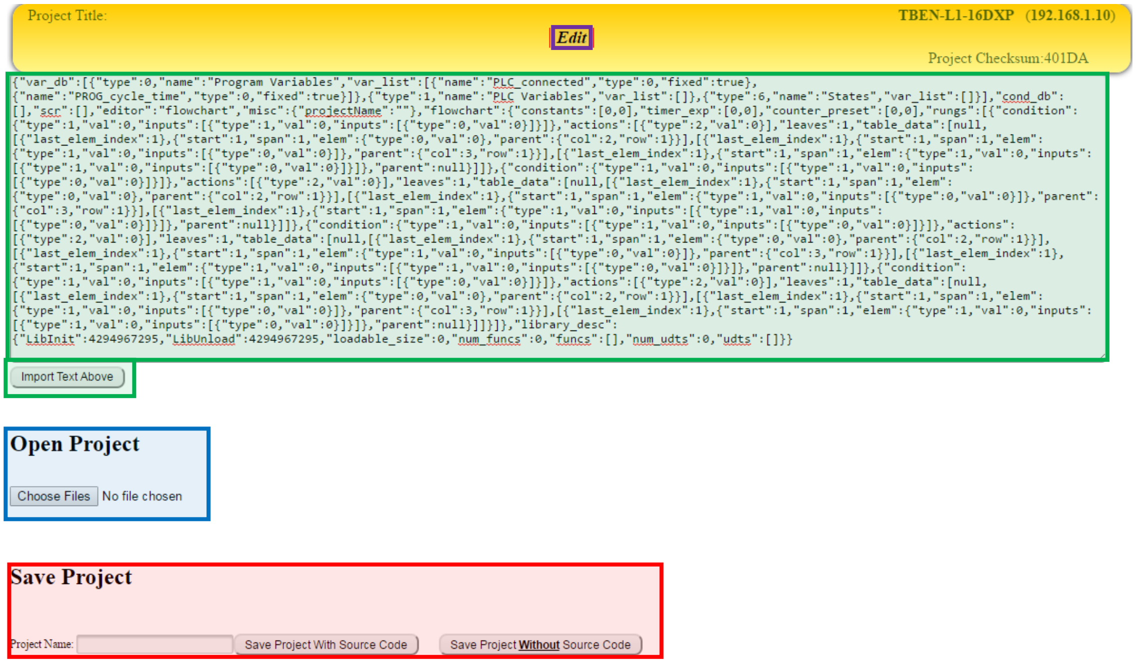
4.3 Open/Save As

The Open/Save feature allows the user to save a current project or load a previous project.



On the Open/Save As screen, the user can perform several actions:

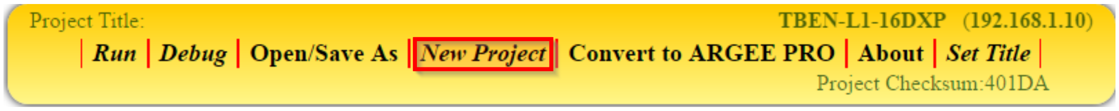
- The *Import Text Above* button imports the above text into the project (highlighted green).
- Under the *Open Project* text, the *Choose Files* button allows the user to browse their computer for a previously saved project (highlighted in blue). Once the file is selected, ARGEE will automatically load the project.
- Under the *Save Project* text, the *Save Project With Source* button allows the user to save their project as an .arg file. The user also has the option to *Save Project **Without** Source Code*. Both highlighted in red.
- The *Edit* tab brings the user back to ARGEE Flow Chart (highlighted in purple).



```
{"var_db":[{"type":0,"name":"Program Variables","var_list":[{"name":"PLC_connected","type":0,"fixed":true},
{"name":"PROG_cycle_time","type":0,"fixed":true}]},{"type":1,"name":"PLC Variables","var_list":[]},{"type":6,"name":"States","var_list":[]}],"cond_db":
[],"scr":[],"editor":"flowchart","misc":{"projectName":""},"flowchart":{"constants":[0,0],"timer_exp":[0,0],"counter_preset":[0,0],"rungs":[{"condition":
{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"actions":[{"type":2,"val":0}],"leaves":1,"table_data":[null,
[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":0,"val":0},"parent":{"col":2,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":
{"type":1,"val":0,"inputs":[{"type":0,"val":0}]},"parent":{"col":3,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":
[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"parent":null}]]},{"condition":{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":
[{"type":0,"val":0}]}]},"actions":[{"type":2,"val":0}],"leaves":1,"table_data":[null,[{"last_elem_index":1},{"start":1,"span":1,"elem":
{"type":0,"val":0},"parent":{"col":2,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":[{"type":0,"val":0}]},"parent":
{"col":3,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":
[{"type":0,"val":0}]}]},"parent":null}]]},{"condition":{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"actions":
[{"type":2,"val":0}],"leaves":1,"table_data":[null,[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":0,"val":0},"parent":{"col":2,"row":1}}],
[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":[{"type":0,"val":0}]},"parent":{"col":3,"row":1}}],[{"last_elem_index":1},
{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"parent":null}]]},{"condition":
{"type":1,"val":0,"inputs":[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"actions":[{"type":2,"val":0}],"leaves":1,"table_data":[null,
[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":0,"val":0},"parent":{"col":2,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":
{"type":1,"val":0,"inputs":[{"type":0,"val":0}]},"parent":{"col":3,"row":1}}],[{"last_elem_index":1},{"start":1,"span":1,"elem":{"type":1,"val":0,"inputs":
[{"type":1,"val":0,"inputs":[{"type":0,"val":0}]}]},"parent":null}]]}]},"library_desc":
{"LibInit":4294967295,"LibUnload":4294967295,"loadable_size":0,"num_funcs":0,"funcs":[],"num_udts":0,"udts":[]}}
```

4.4 New Project

The user clicks on New Project to start a new project. They will be brought to the ARGEE Flow programming page if a new project is started.



NOTE

Starting a new project does not erase the code on the user’s block. If the user wants to erase the code on the block, the user has two ways of doing it. These are shown below.

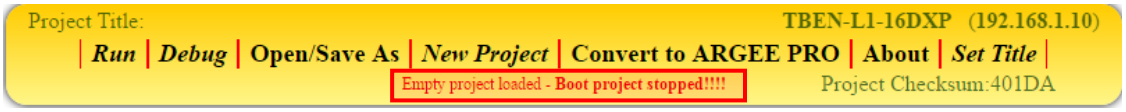
4.4.1 How to Erase a Project from a Device

4.4.1.1 Running an Empty Project

If the user wants to erase the code on the block, one way of doing that is by running an empty project.

➢ The user needs to first start a New Project and then click Run.

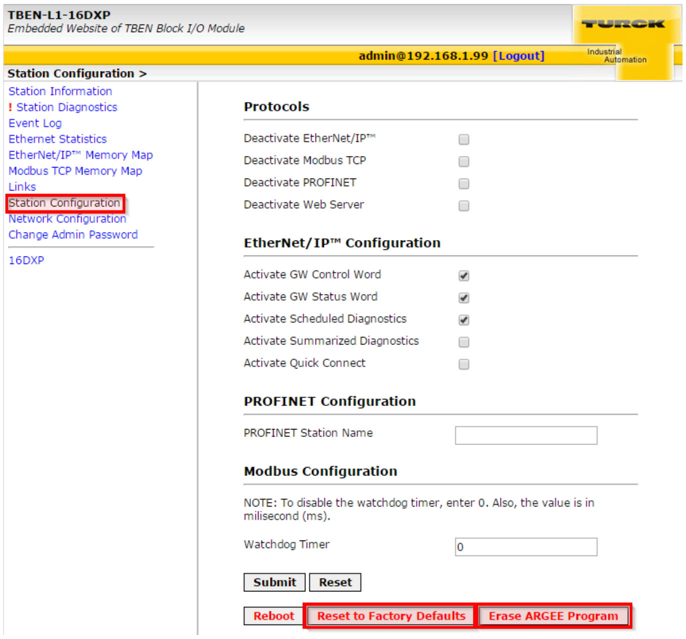
➔ This action will load an empty project into the block.



Empty project loaded - Boot project stopped!!!!

4.4.1.2 Using the Webserver to delete ARGEE code

The user can also remove the ARGEE code by selecting *Erase ARGEE Program* or *Reset to Factory Defaults* inside the webserver page.



TBEN-L1-16DXP
Embedded Website of TBEN Block I/O Module

admin@192.168.1.99 [Logout]

Station Configuration >

Protocols

Deactivate EtherNet/IP™ □
Deactivate Modbus TCP □
Deactivate PROFINET □
Deactivate Web Server □

EtherNet/IP™ Configuration

Activate GW Control Word ☑
Activate GW Status Word ☑
Activate Scheduled Diagnostics ☑
Activate Summarized Diagnostics □
Activate Quick Connect □

PROFINET Configuration

PROFINET Station Name

Modbus Configuration

NOTE: To disable the watchdog timer, enter 0. Also, the value is in milisecond (ms).

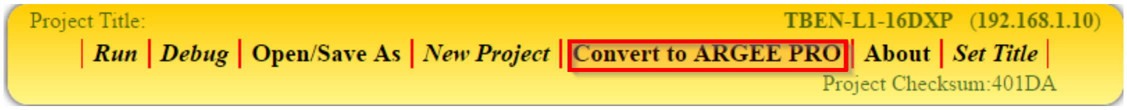
Watchdog Timer 0

NOTE

Getting to the webserver is discussed in section 7.4 Embedded Webserver.

4.5 Convert to ARGEE PRO

The user will click *Convert to ARGEE PRO* when they want to leave the Flow Chart mode and enter the ARGEE PRO Programming Environment. ARGEE PRO is discussed in chapter 5 ARGEE PRO Menu Bar.

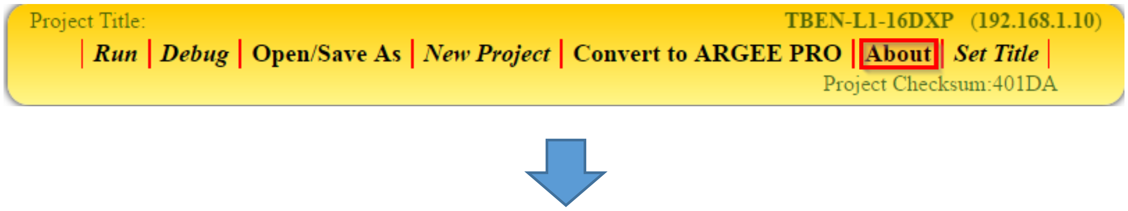


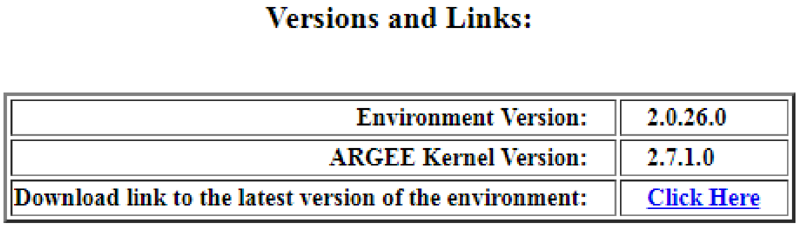
NOTE

Once the user selects *Convert to ARGEE PRO*, they cannot convert back to Flow Chart.

4.6 About

The user can click *About* if they want to view the ARGEE environment and kernel firmware revisions.





Versions and Links:

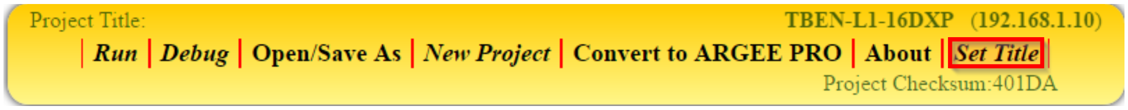
Environment Version:	2.0.26.0
ARGEE Kernel Version:	2.7.1.0
Download link to the latest version of the environment:	Click Here

NOTE

The user can use the **Click Here** hyperlink to download the latest ARGEE environment.

4.7 Set Title

The user can click *Set Title* to add a name to the project.

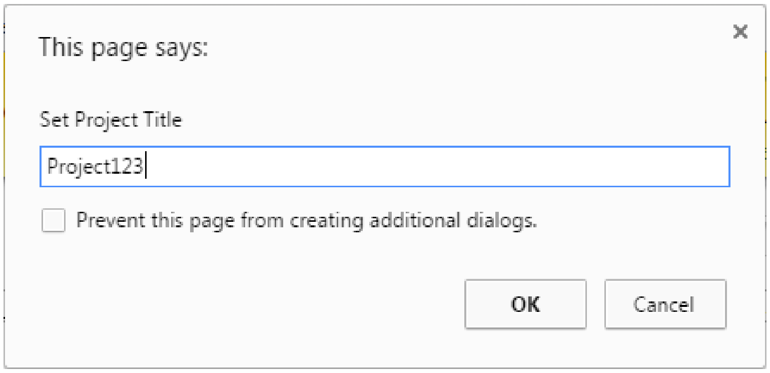


Project Title:

TBEN-L1-16DXP (192.168.1.10)

| Run | Debug | Open/Save As | New Project | Convert to ARGEE PRO | About | Set Title |

Project Checksum:401DA

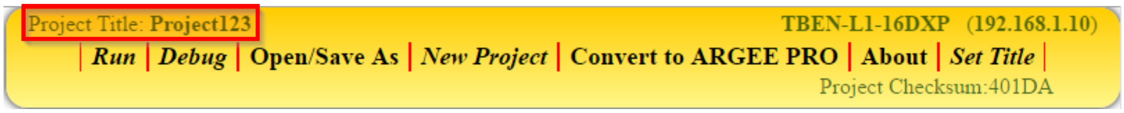


This page says:

Set Project Title

Project123

□ Prevent this page from creating additional dialogs.



Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

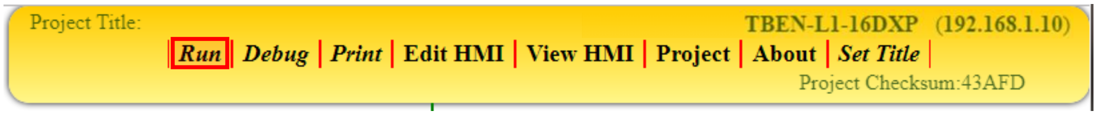
| Run | Debug | Open/Save As | New Project | Convert to ARGEE PRO | About | Set Title |

Project Checksum:401DA

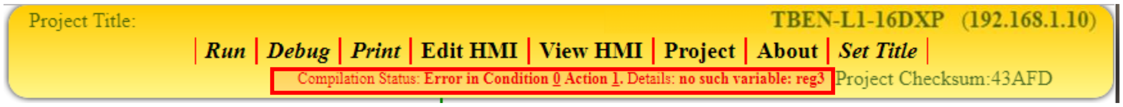
5 ARGEE PRO Menu Bar

5.1 Run

When the user clicks *Run*, several things happen. First, ARGEE checks the code for errors. If the code has no errors, ARGEE downloads the code to the block. It also calculates and displays how much memory the code has used and how much memory is still available. Next, ARGEE transitions over to the Debug screen.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Project Checksum:43AFD

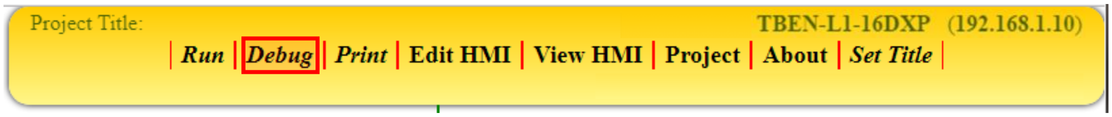
If the code has errors, ARGEE will display an error message and tell the user where the error is located in the code.

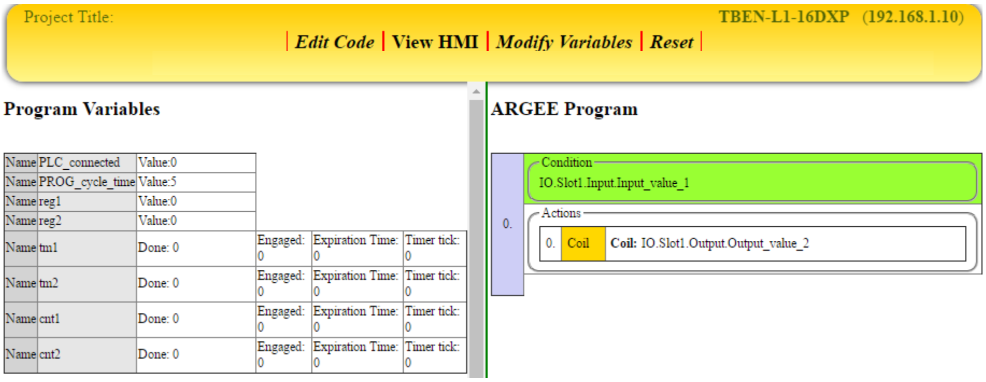

Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Project Checksum:43AFD

Compilation Status: Error in Condition 0 Action 1. Details: no such variable: reg3

5.2 Debug (ARGEE PRO)

When the user clicks *Debug* while in the ARGEE PRO, the user can visually observe code progression from a more advanced screen, compared to ARGEE Flow. Sections highlighted in green indicate they are true/triggered.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Edit Code | View HMI | Modify Variables | Reset

Program Variables

Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0

ARGEE Program

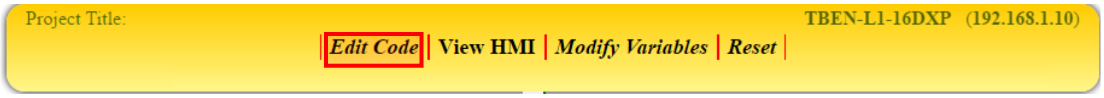
0.
Condition
IO.Slot1.Input.Input_value_1
Actions
0. Coil **Coil:** IO.Slot1.Output.Output_value_2

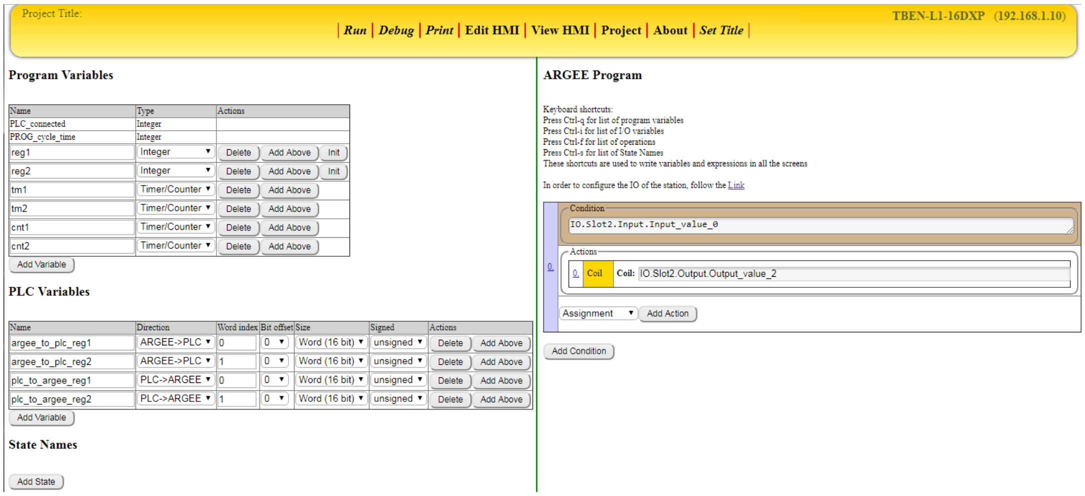
NOTE

The View HMI tab can be accessed from multiple different screens in ARGEE 2 PRO. The View HMI function is discussed in section 5.5 View HMI.

5.2.1 Edit Code

The user can find the *Edit Code* tab on many screens in the ARGEE 2 Pro environment. The user will click *Edit Code* when they want to leave their current location and return to the ARGEE Pro programming page.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Edit Code | View HMI | Modify Variables | Reset


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Add Variable
Add Variable
Add State
Add Condition
Assignment
Add Action

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

State Names

ARGEE Program

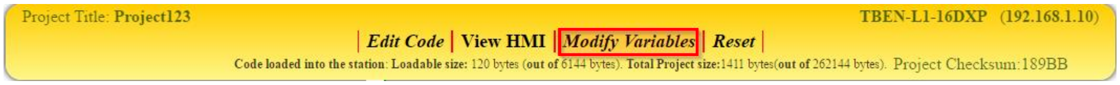
Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

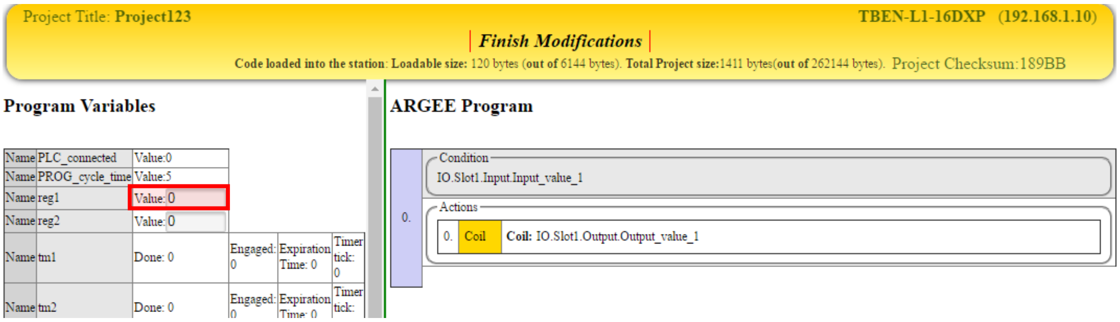
0.
Condition: `IO.Slot2.Input.Input_value_0`
Actions: 0. Coil **Coil:** IO.Slot2.Output.Output_value_2

5.2.2 Modify Variables

The *Modify Variables* tab is available in ARGEE PRO and in the ARGEE Simulation Mode. It only becomes visible when the user is in Debug mode. From the *Modify Variables* screen, the user can manually change register and variable values.


Project Title: Project123
TBEN-L1-16DXP (192.168.1.10)
Edit Code | View HMI | Modify Variables | Reset

Code loaded into the station: **Loadable size:** 120 bytes (**out of** 6144 bytes). **Total Project size:**1411 bytes(**out of** 262144 bytes). Project Checksum:189BB



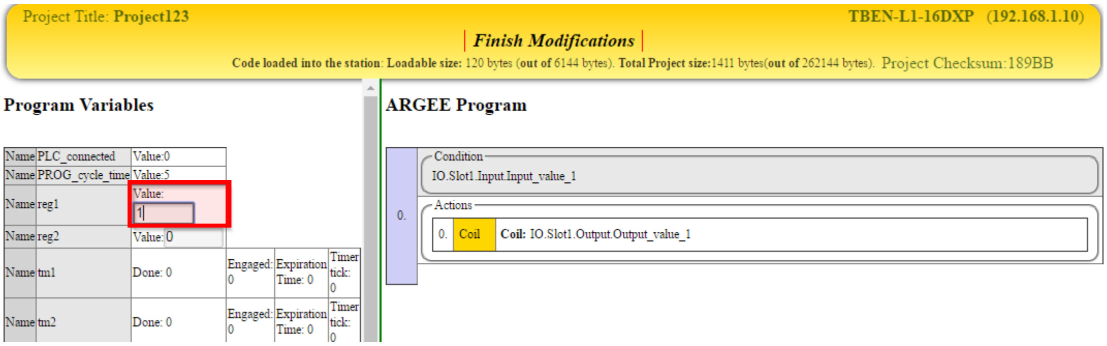
Code loaded into the station: **Loadable size:** 120 bytes (**out of** 6144 bytes). **Total Project size:**1411 bytes(**out of** 262144 bytes). Project Checksum:189BB

Program Variables

Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value: 0			
Name	reg2	Value: 0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick:

ARGEE Program

0.
Condition: IO.Slot1.Input.Input_value_1
Actions: 0. Coil **Coil:** IO.Slot1.Output.Output_value_1



Project Title: Project123 TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:1411 bytes(out of 262144 bytes). Project Checksum:189BB

Program Variables

Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value: 1			
Name	reg2	Value: 0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0

ARGEE Program

0.

Condition

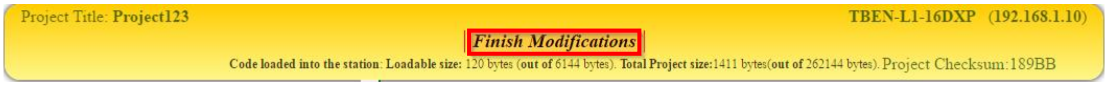
IO.Slot1.Input.Input_value_1

Actions

0. Coil **Coil:** IO.Slot1.Output.Output_value_1

5.2.2.1 Finish Modifications

The user can select *Finish Modification* to exit *Modify Variables* mode.

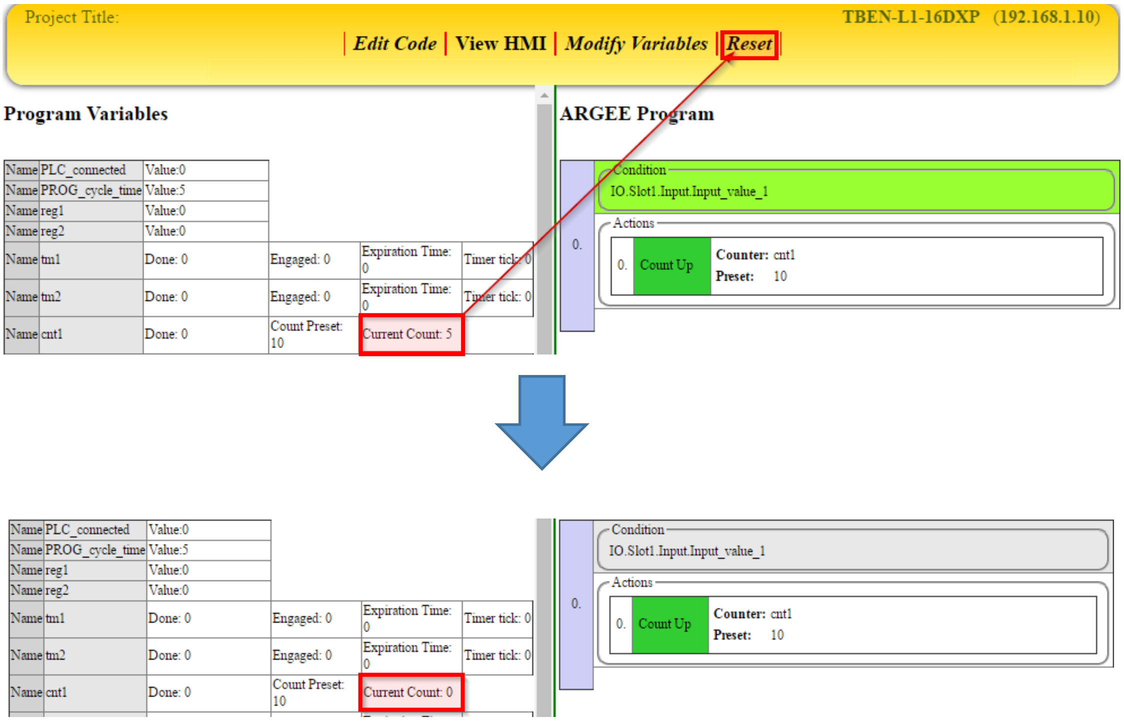


Project Title: Project123 TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 120 bytes (out of 6144 bytes). Total Project size:1411 bytes(out of 262144 bytes). Project Checksum:189BB

5.2.3 Reset

When the user clicks *Reset* while in the ARGEE PRO, the user can visually observe timers and counters resetting to zero from a more advanced screen than in ARGEE Flow.



Project Title: TBEN-L1-16DXP (192.168.1.10)

Program Variables

Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Count Preset: 10	Current Count: 5	

ARGEE Program

0.

Condition

IO.Slot1.Input.Input_value_1

Actions

0. Count Up **Counter:** cnt1 **Preset:** 10

Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Count Preset: 10	Current Count: 0	

0.

Condition

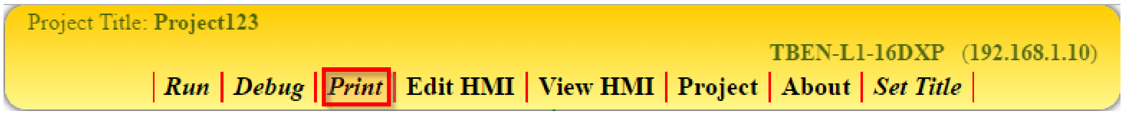
IO.Slot1.Input.Input_value_1

Actions

0. Count Up **Counter:** cnt1 **Preset:** 10

5.3 Print

The user can click *Print* if they want to print out a copy of their project.

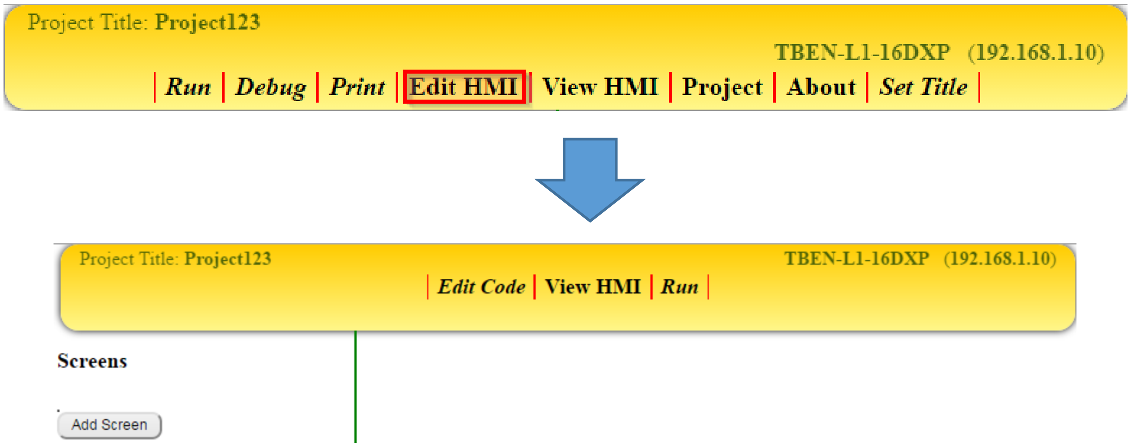


Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

5.4 Edit HMI

The *Edit HMI* tab brings the user to the edit HMI screen.



Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

Project Title: Project123

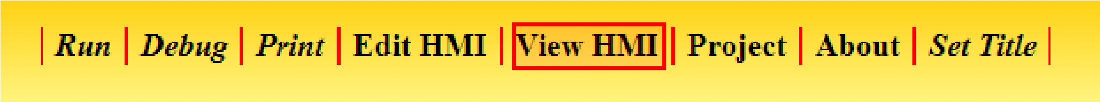
TBEN-L1-16DXP (192.168.1.10)

NOTE

Edit Code is discussed in section 5.2.1 Edit Code. *View HMI* is discussed in section 5.5 View HMI. *Run* is discussed in section 5.1 Run. Instructions for how to build an HMI are located in chapter 13 ARGEE HMI.

5.5 View HMI

The *View HMI* tab allows the user to view their HMI screen. This tab becomes active after the user has already built an HMI.

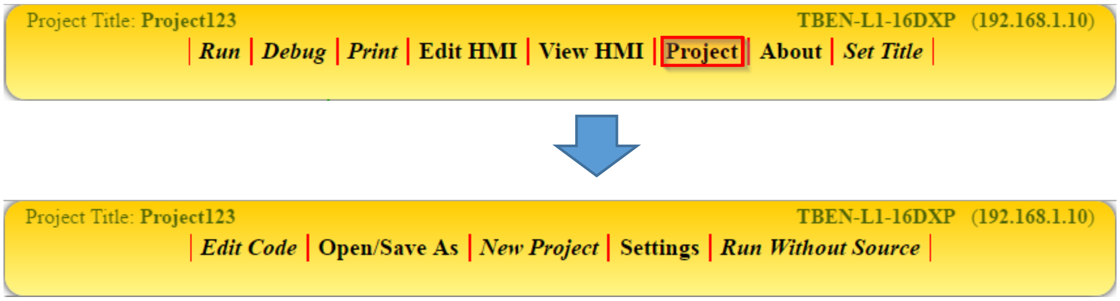


NOTE

Instructions for how to build an HMI are located in chapter 13 ARGEE HMI.

5.6 Project

When the user clicks on the *Project* tab, they will have access to a second ARGEE Menu Bar.



Project Title: Project123 TBEN-L1-16DXP (192.168.1.10)

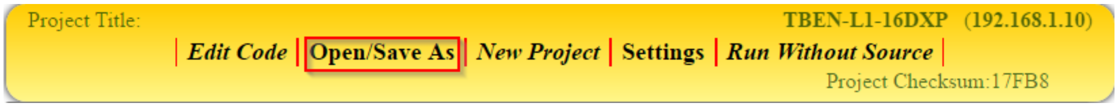
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Edit Code | Open/Save As | New Project | Settings | Run Without Source

NOTE

Edit Code is discussed in section 5.2.1 Edit Code.

5.6.1 Open/Save As

The Open/Save feature allows the user to save a current project or load a previous project.



Project Title: TBEN-L1-16DXP (192.168.1.10)

Project Checksum:17FB8

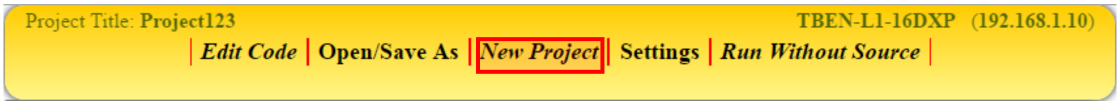
Edit Code | Open/Save As | New Project | Settings | Run Without Source

NOTE

The Open/Save As tab in ARGEE PRO has the same function as the Open/Save As tab in ARGEE Flow and is discussed in section 4.3 Open/Save As.

5.6.2 New Project

The user clicks on New Project to start a new project. They will be brought to the ARGEE Flow programming page if a new project is started.



Project Title: Project123 TBEN-L1-16DXP (192.168.1.10)

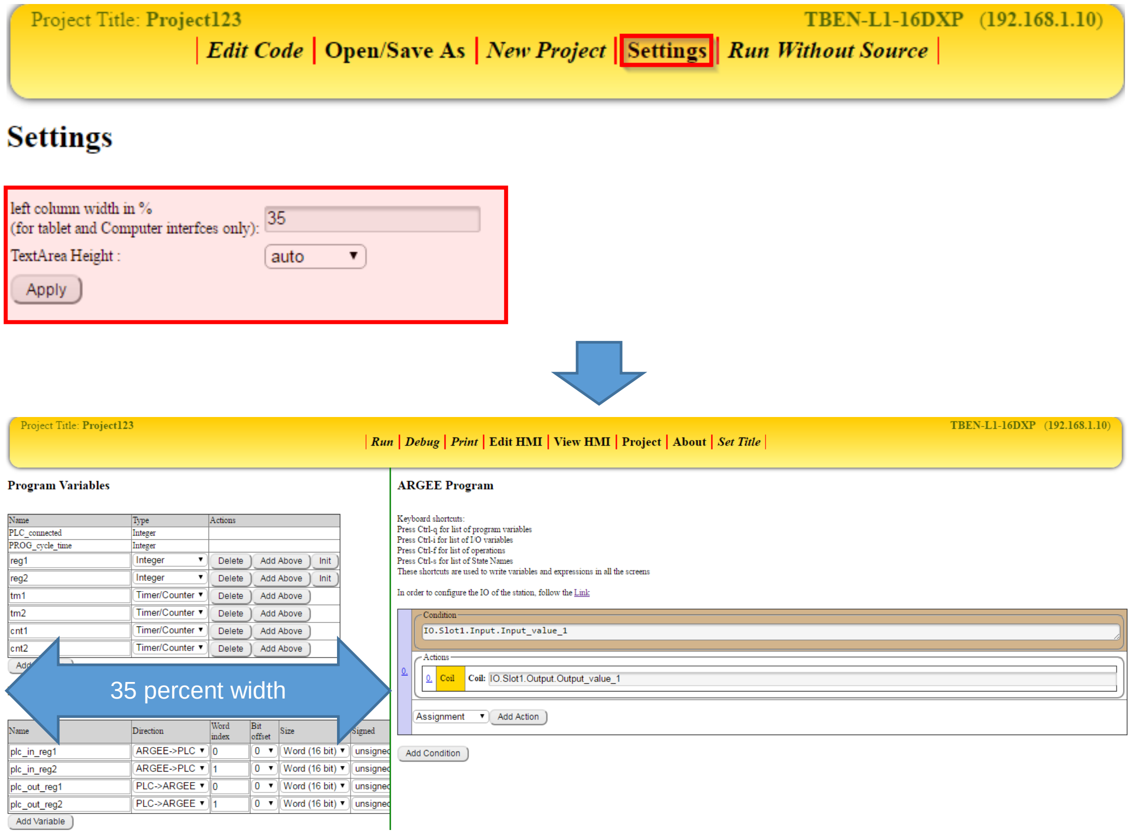
Edit Code | Open/Save As | New Project | Settings | Run Without Source

NOTE

Starting a new project does not erase the code on the user’s block. If the user wants to erase the code on the block, the user has two ways of doing it. These are discussed in section 4.4.1 How to Erase a Project from a Device.

5.6.3 Settings

From the *Settings* screen, the user can set the “percentage” of their screen that displays their Variables and State Names. The remaining “percentage” of the screen displays the ARGEE Program.


Project Title: Project123
TBEN-L1-16DXP (192.168.1.10)
Edit Code | Open/Save As | New Project | Settings | Run Without Source
Apply
35 percent width
Project Title: Project123
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Delete
Add Above
Init
Add Variable
Assignment
Add Action
Add Condition

Settings

left column width in %
(for tablet and Computer interfces only): 35

TextArea Height : auto

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

Name	Direction	Word index	Bit offset	Size	Signed
plc_in_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned
plc_in_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned
plc_out_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned
plc_out_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

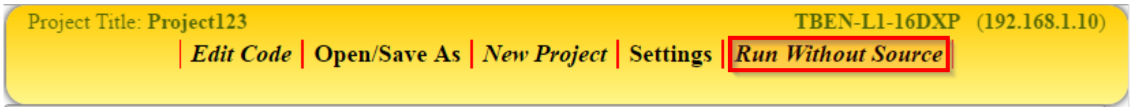
In order to configure the IO of the station, follow the Link

0. Condition: IO.Slot1.Input.Input_value_1

Actions: 0. Coil Coil: IO.Slot1.Output.Output_value_1

5.6.4 Run Without Source

Selecting *Run Without Source* will allow the user to run a program without displaying the actual code. The end user will not be able to access source code by loading the ARGEE environment. *Run Without Source* is one of ARGEE’s security protocols.


Project Title: Project123
TBEN-L1-16DXP (192.168.1.10)
Edit Code | Open/Save As | New Project | Settings | Run Without Source

NOTE

The user needs to save a Master Copy of the program before the user logs out of the environment if the user wants to view/edit the code in the future. Security protocols are discussed in chapter 11 ARGEE Security.

5.7 About and Set Tittle

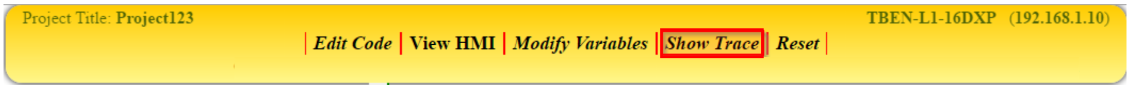
The About and Title tab are same as in ARGEE Flow. Both are discussed in chapter 4. About is discussed in section 4.6 About. Title is discussed in section 4.7 Set Title.

5.8 Trace Mode

The user will use Trace when they want to measure the run-time behavior of the program. Trace allows the user to measure how long each state takes as well as which states were visited in which order. Trace is an Action that must be inserted into the code before Trace will appear in the menu bar.

5.8.1 Show Trace

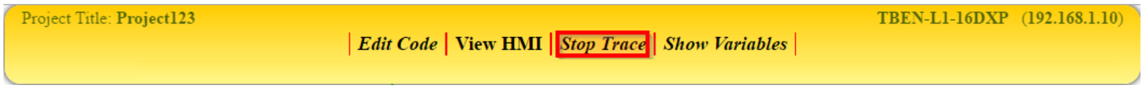
The user can click on Show Trace to view the active Trace data.


Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

5.8.2 Stop Trace

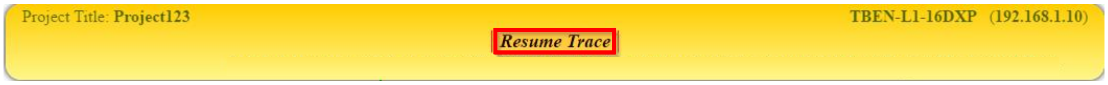
The user can click on Stop Trace to easily view the programs historical run-time data.


Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

5.8.3 Resume Trace

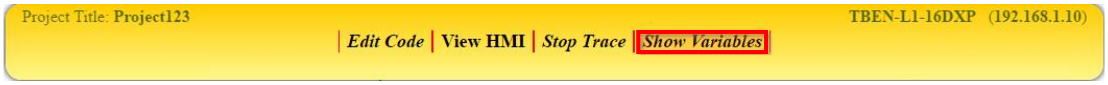
The user can click on Resume Trace to resume tracing the programs run-time.


Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

5.8.4 Show Variables

The user can click on *Show Variables* if the user wants to leave Trace Mode.


Project Title: Project123

TBEN-L1-16DXP (192.168.1.10)

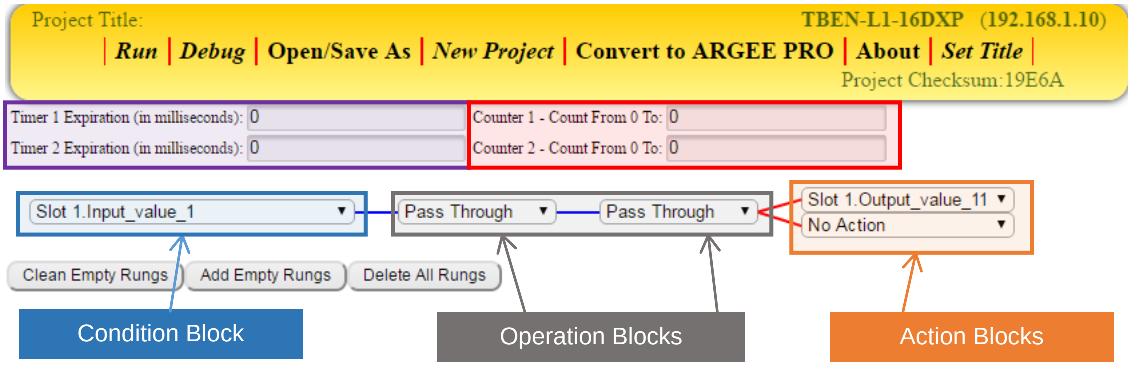
NOTE

More information about Trace can be found in section 8.2.6 Trace.

6 Getting Familiar with ARGEE Flow Chart

6.1 The Basics

The Flow Chart Editor is made up of Condition, Operation, and Action Blocks. Conditions, Operations and Actions are selected by clicking their respective drop down arrows. The Flow Chart Editor also provides the user with two timers (highlighted purple) and two counters (highlighted red).



Project Title:

TBEN-L1-16DXP (192.168.1.10)

Project Checksum:19E6A

Timer 1 Expiration (in milliseconds): 0

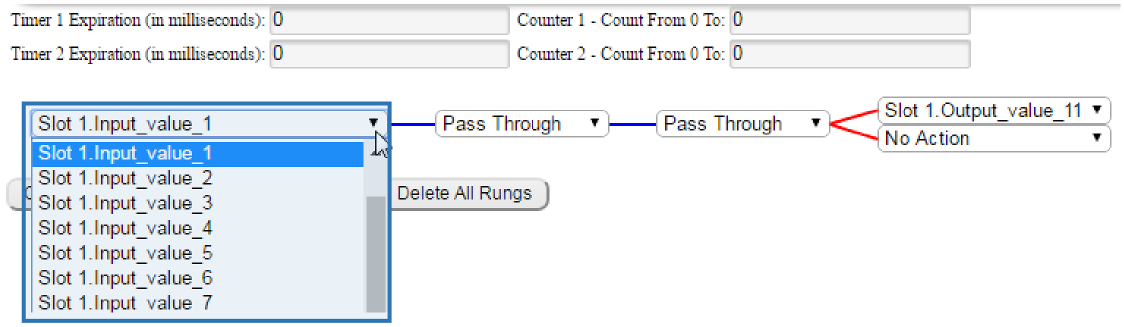
Timer 2 Expiration (in milliseconds): 0

Counter 1 - Count From 0 To: 0

Counter 2 - Count From 0 To: 0

6.2 Conditions

The Condition Block contains Input conditions. The input conditions the user sees corresponds to the block the user is connected to. Other included input conditions are: Timer X expired, Counter X expired, Internal Reg X, and PLC In Reg X.



Timer 1 Expiration (in milliseconds): 0

Timer 2 Expiration (in milliseconds): 0

Counter 1 - Count From 0 To: 0

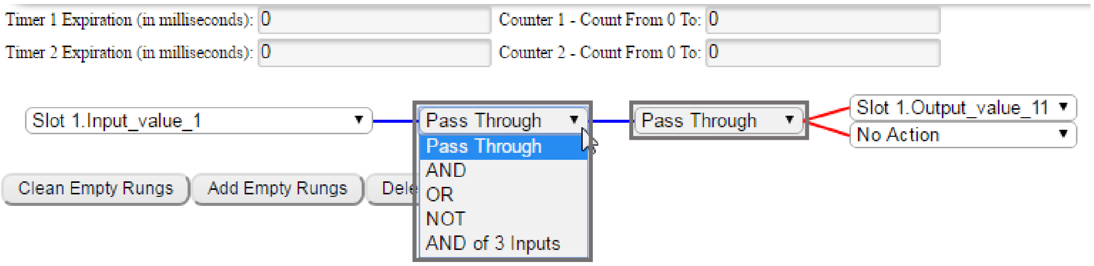
Counter 2 - Count From 0 To: 0

NOTE

Expired functions are discussed in section 9.5.2 Expired. Internal Reg's (Reg = Register) are discussed in section 6.10 Internal Registers. PLC in Reg's are discussed in section 7.6 PLC Variables

6.3 Operations

The Operation Blocks contain various Boolean operations. If no Operations are desired, select Pass Through.



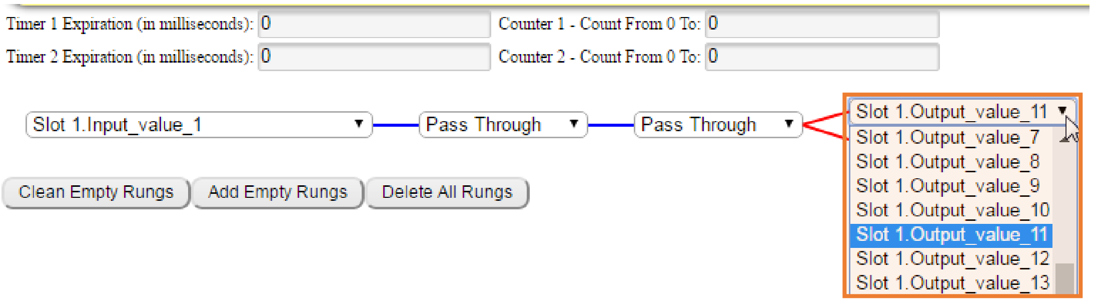
Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

NOTE

Boolean logic is discussed in section 9.3 Boolean Logic.

6.4 Actions

The Action Block contains Output conditions, The Output conditions the user sees corresponds to the block the user is connected to. Other included Output conditions are: TON Timer X (**T**urn **ON** Timer X), CTU Counter X (**C**oun**T** **U**p Counter X), RESET Counter X, Internal Reg X, PLC Out Reg X.



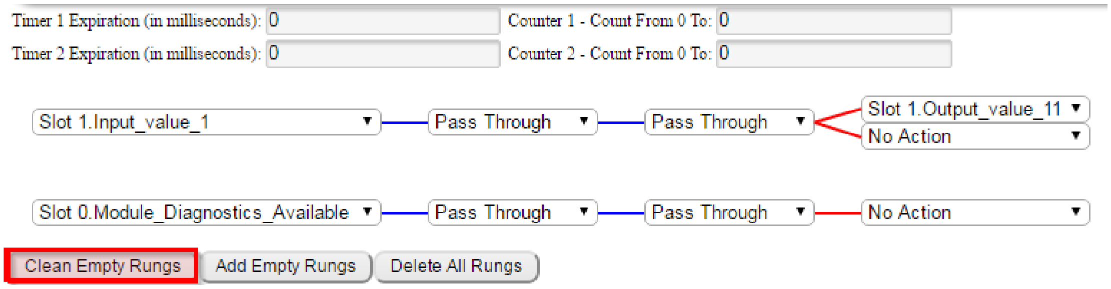
Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

NOTE

TON is discussed in section 8.2.4 Timer On. CTU is discussed in section 8.2.8 Count Up. RESET Counter is discussed in section 8.2.10 Reset Counter. Internal Reg's (Reg = Register) are discussed in section 6.10 Internal Registers. PLC Reg's are discussed in section 7.6 PLC Variables.

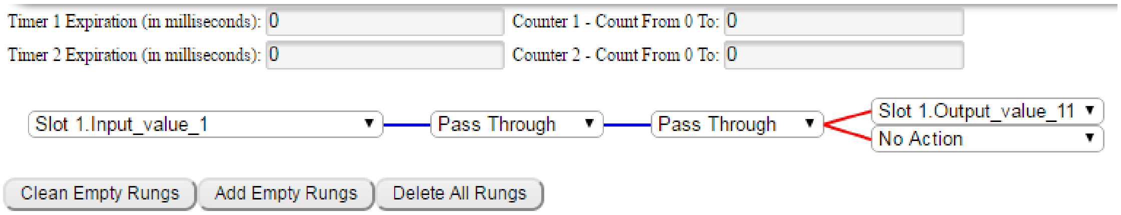
6.5 Clean Empty Runs

The *Clean Empty Rungs* button will remove all unused rungs from the Flow Chart Editor.


Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 1.Output_value_11 ▾ / No Action ▾

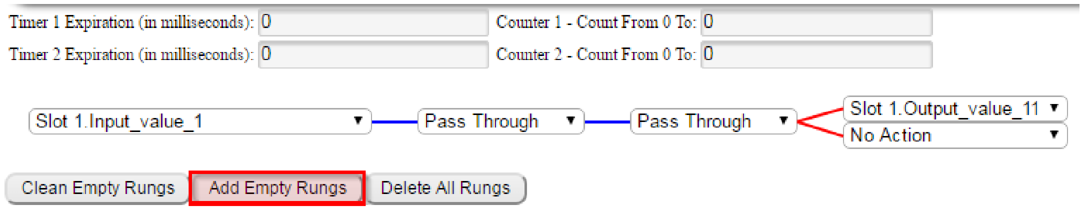
Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾


Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

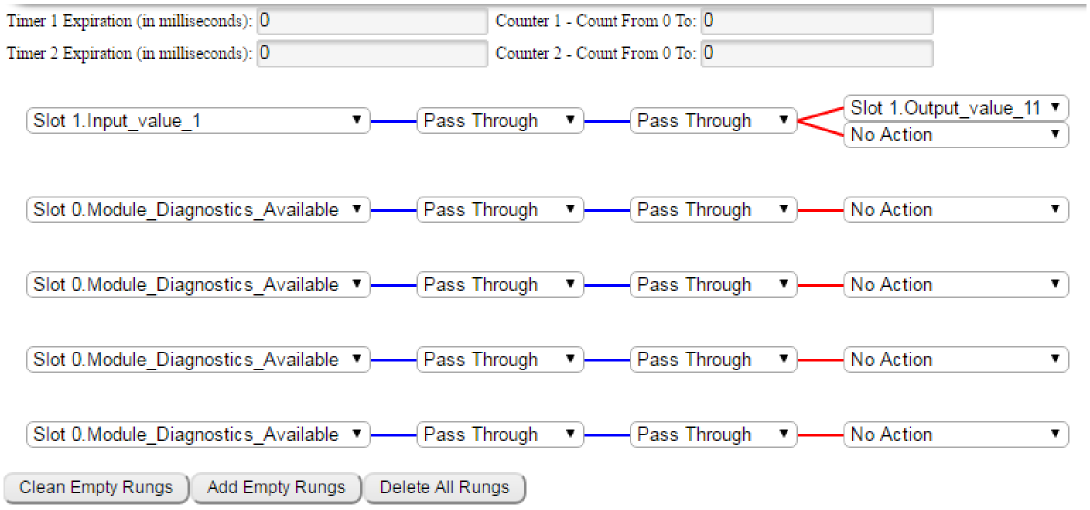
Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 1.Output_value_11 ▾ / No Action ▾

6.6 Add Empty Rungs

The *Add Empty Rungs* button will add four empty rungs to Flow Chart Editor.


Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 1.Output_value_11 ▾ / No Action ▾


Timer 1 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Timer 2 Expiration (in milliseconds): 0
Counter 2 - Count From 0 To: 0

Slot 1.Input_value_1 ▾ — Pass Through ▾ — Pass Through ▾ — Slot 1.Output_value_11 ▾ / No Action ▾

Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

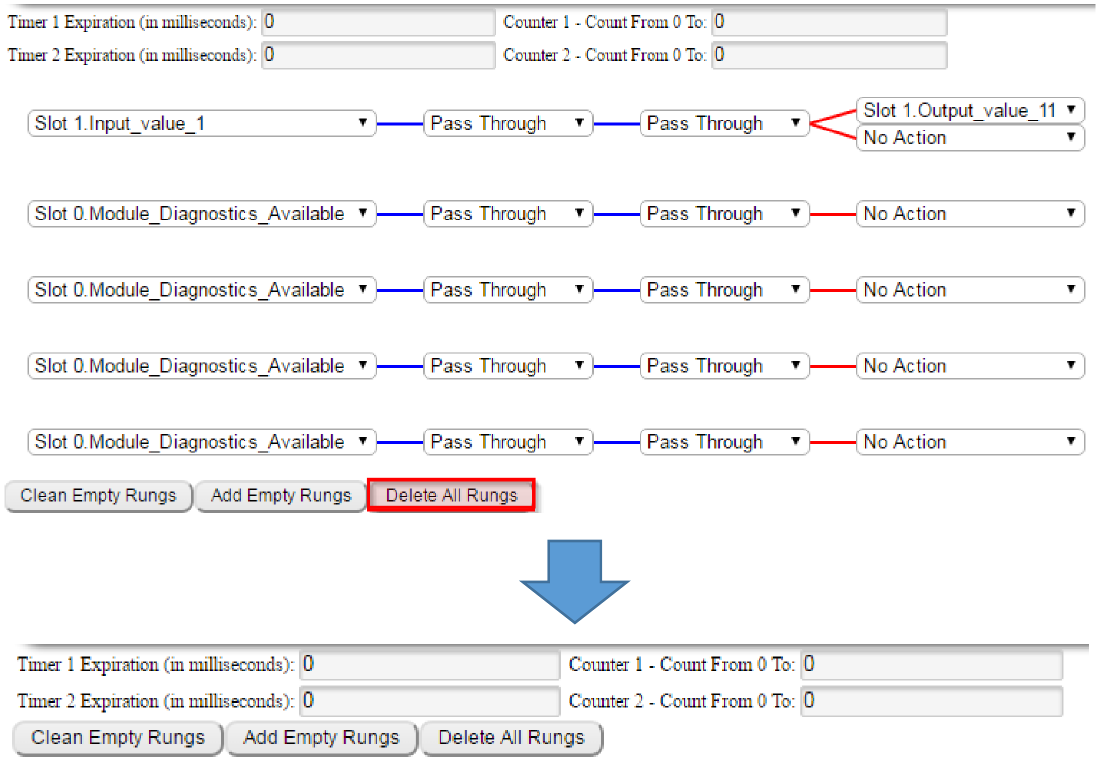
Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

Slot 0.Module_Diagnostics_Available ▾ — Pass Through ▾ — Pass Through ▾ — No Action ▾

6.7 Delete All Rungs

The *Delete All Rungs* button will remove all rungs from Flow Chart Editor.



Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 0

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

Slot 1.Input_value_1 — Pass Through — Pass Through — Slot 1.Output_value_11 / No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — No Action

Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 0

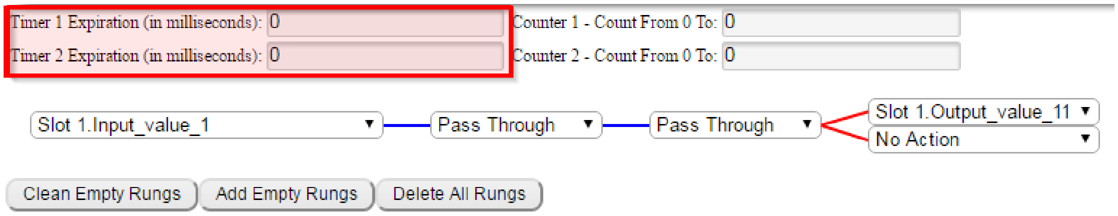
Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

NOTE

Used and unused rungs will both be deleted from the project.

6.8 Timers

Flow Chart Editor contains two *Timers*. The user can set the Timers by typing a value into the Timer text box. Timer values are in milliseconds (1000 Milliseconds = 1 Second).



Timer 1 Expiration (in milliseconds): 0 Counter 1 - Count From 0 To: 0

Timer 2 Expiration (in milliseconds): 0 Counter 2 - Count From 0 To: 0

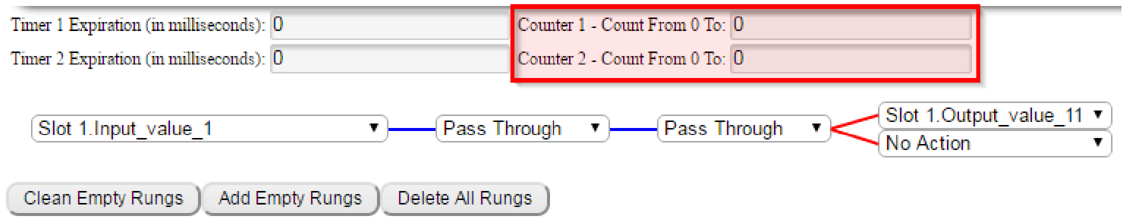
Slot 1.Input_value_1 — Pass Through — Pass Through — Slot 1.Output_value_11 / No Action

NOTE

Timer examples can be seen in sections 8.2.2 Timer Start, 8.2.4 Timer On, 8.2.5 Timer Off, 9.5 Timer/Counter.

6.9 Counters

Flow Chart Editor contains two *Counters*. The user can set the Counters by typing a value into the Counter text box.



Timer 1 Expiration (in milliseconds): 0
Timer 2 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Counter 2 - Count From 0 To: 0

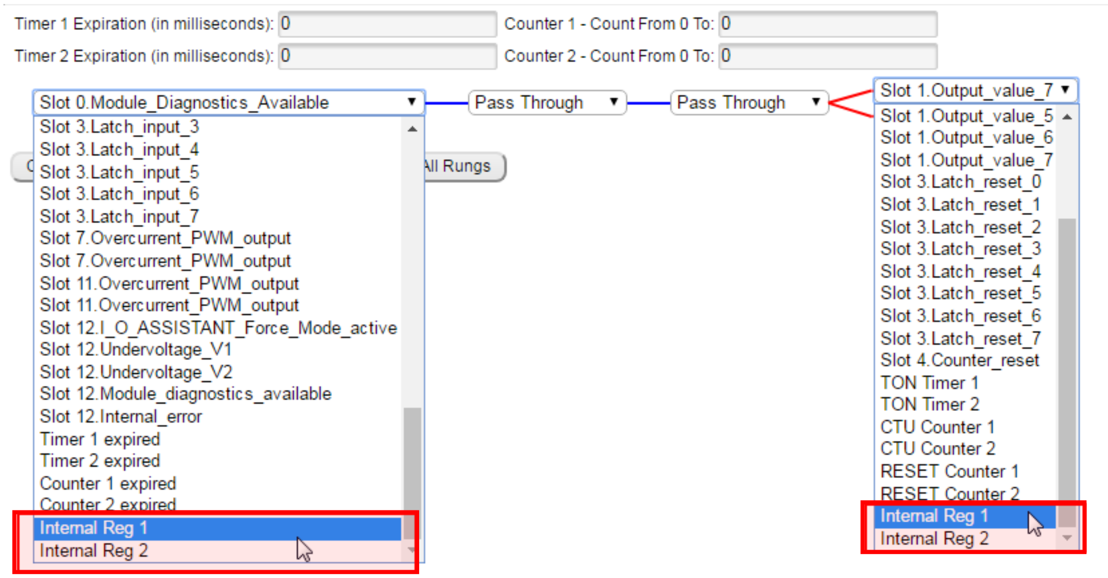
Slot 1.Input_value_1 — Pass Through — Pass Through — Slot 1.Output_value_11 / No Action

NOTE

Counter examples can be seen in sections 8.2.8 Count Up, 8.2.9 Count Down, 8.2.10 Reset Counter, 9.5 Timer/Counter.

6.10 Internal Registers

Flow Chart Editor contains two *Internal Regs (Reg = register)*. The user can use an internal register as a condition to trigger an action or as an action to trigger a condition.



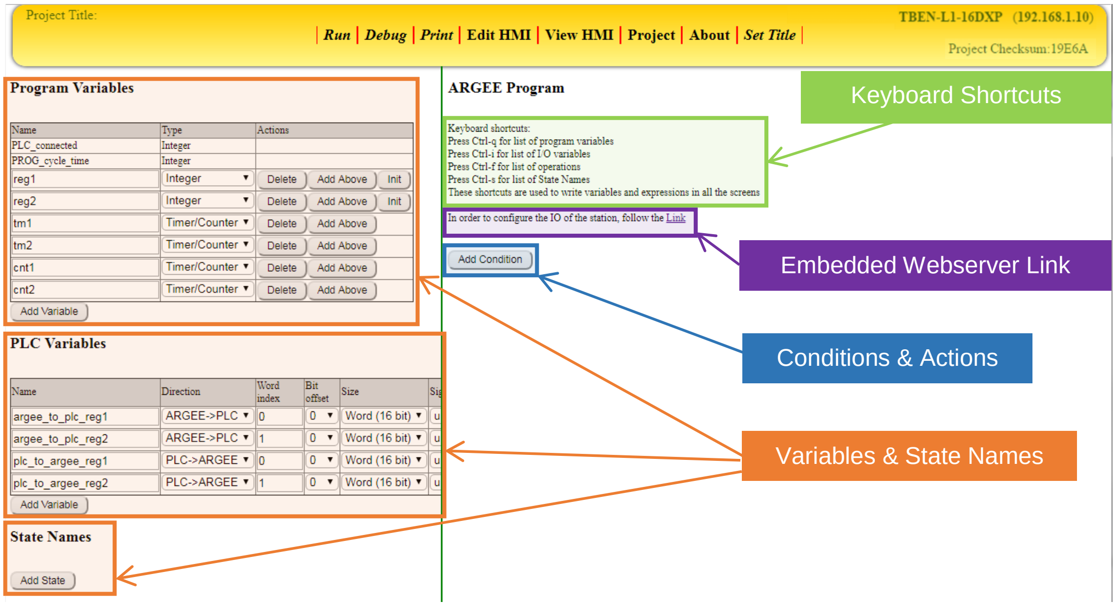
Timer 1 Expiration (in milliseconds): 0
Timer 2 Expiration (in milliseconds): 0
Counter 1 - Count From 0 To: 0
Counter 2 - Count From 0 To: 0

Slot 0.Module_Diagnostics_Available — Pass Through — Pass Through — Slot 1.Output_value_7

7 Getting Familiar with ARGEE PRO

7.1 The Basics

The ARGEE PRO home page is made up of Conditions & Actions, An Embedded Webserver Link, Variables & State Names, and Keyboard Shortcuts.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

PLC Variables

Name	Direction	Word index	Bit offset	Size	Sig
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	u
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	u
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	u
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	u

State Names

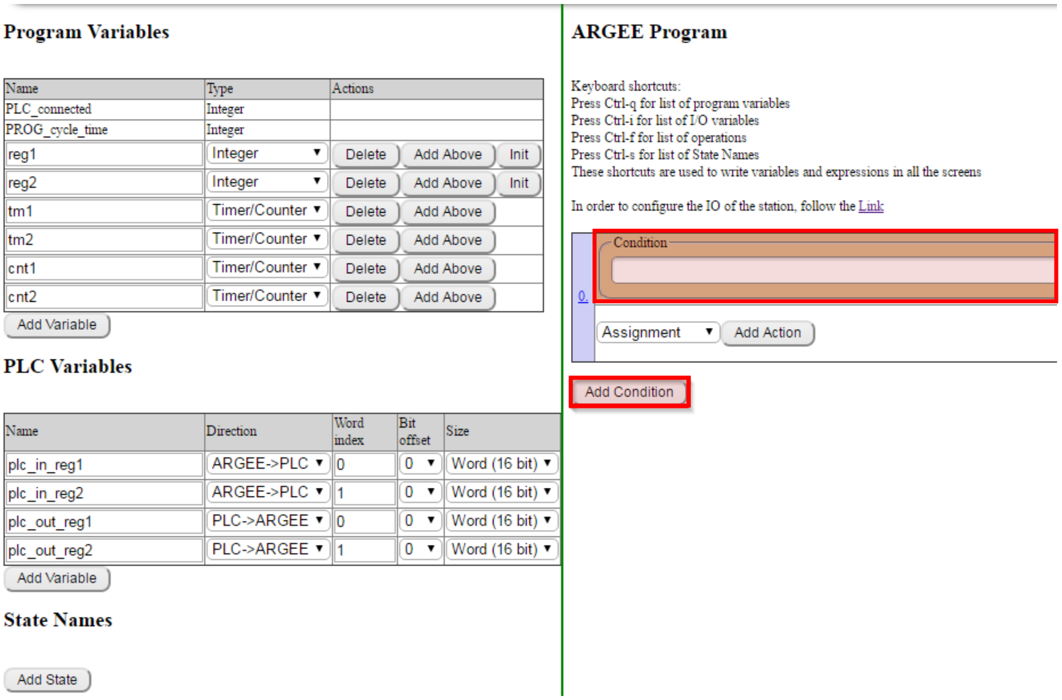
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

7.2 Conditions

The *Add Condition* button will add one blank condition to the ARGEE project. This environment is executed in the manner of IF / THEN statements. ARGEE calls them Conditions (IF) and Actions (THEN).



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

PLC Variables

Name	Direction	Word index	Bit offset	Size
plc_in_reg1	ARGEE->PLC	0	0	Word (16 bit)
plc_in_reg2	ARGEE->PLC	1	0	Word (16 bit)
plc_out_reg1	PLC->ARGEE	0	0	Word (16 bit)
plc_out_reg2	PLC->ARGEE	1	0	Word (16 bit)

State Names

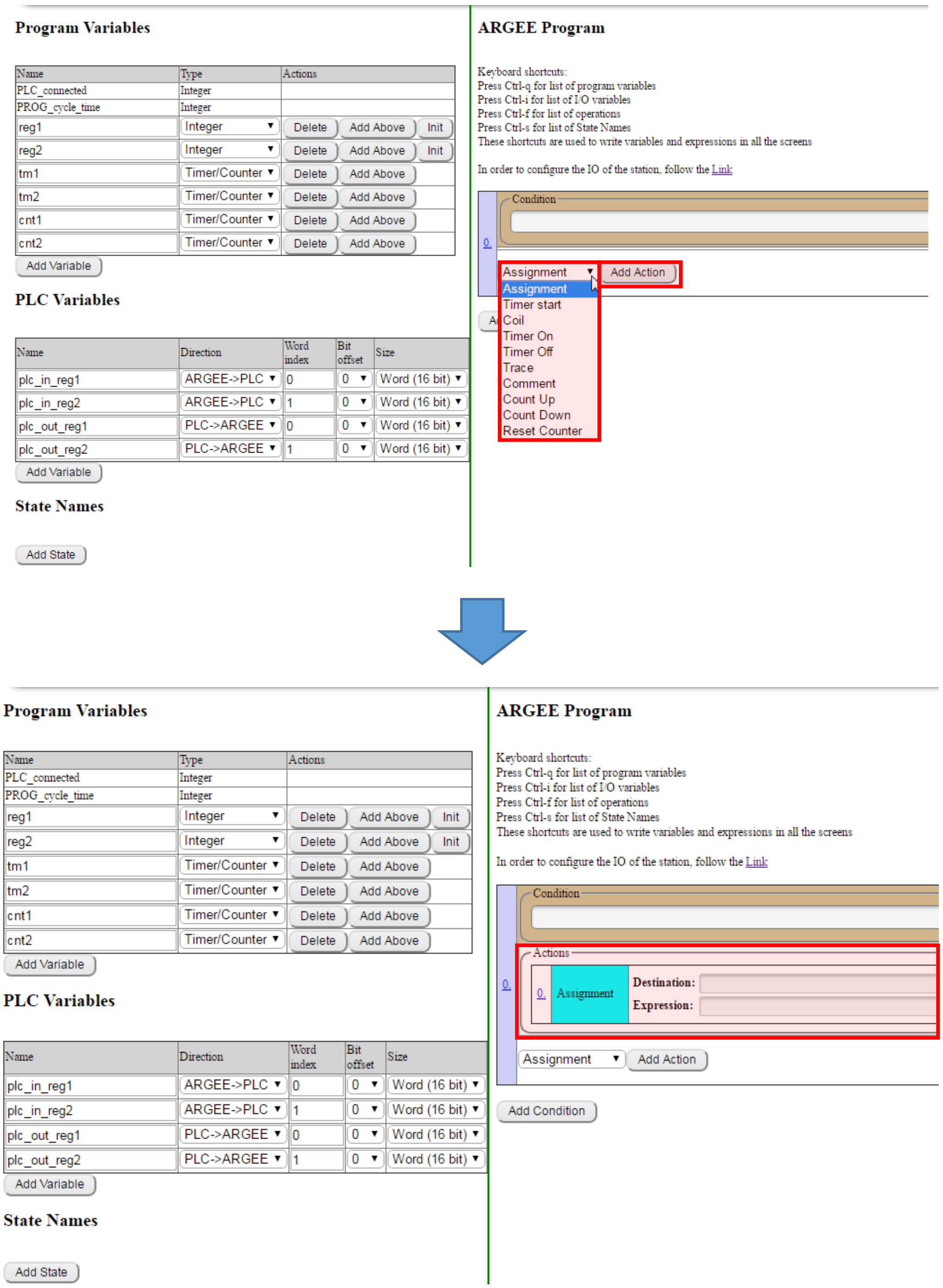
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

7.3 Actions

Actions are selected from a pull down menu. Users select the desired action, and then select the *Add Action* button.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

Add Variable

PLC Variables

Name	Direction	Word index	Bit offset	Size
plc_in_reg1	ARGEE->PLC	0	0	Word (16 bit)
plc_in_reg2	ARGEE->PLC	1	0	Word (16 bit)
plc_out_reg1	PLC->ARGEE	0	0	Word (16 bit)
plc_out_reg2	PLC->ARGEE	1	0	Word (16 bit)

Add Variable

State Names

Add State

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

Add Variable

PLC Variables

Name	Direction	Word index	Bit offset	Size
plc_in_reg1	ARGEE->PLC	0	0	Word (16 bit)
plc_in_reg2	ARGEE->PLC	1	0	Word (16 bit)
plc_out_reg1	PLC->ARGEE	0	0	Word (16 bit)
plc_out_reg2	PLC->ARGEE	1	0	Word (16 bit)

Add Variable

State Names

Add State

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

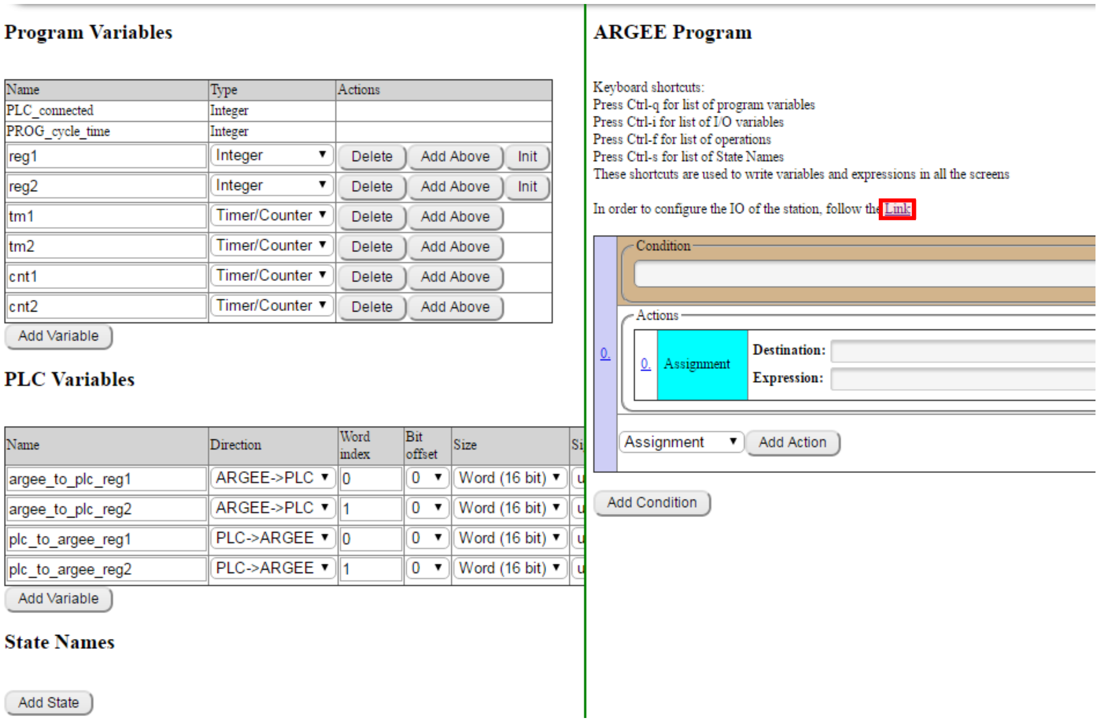
In order to configure the IO of the station, follow the Link

NOTE

Some Actions are executed even if the Condition is false. See chapter 12 System Performance for more information in this topic.

7.4 Embedded Webserver

The user can click *Link* to access the connected blocks webserver. Once the user is in the webserver, they can view the device status and even set device parameters.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete · Add Above · Init
reg2	Integer ▼	Delete · Add Above · Init
tm1	Timer/Counter ▼	Delete · Add Above
tm2	Timer/Counter ▼	Delete · Add Above
cnt1	Timer/Counter ▼	Delete · Add Above
cnt2	Timer/Counter ▼	Delete · Add Above

PLC Variables

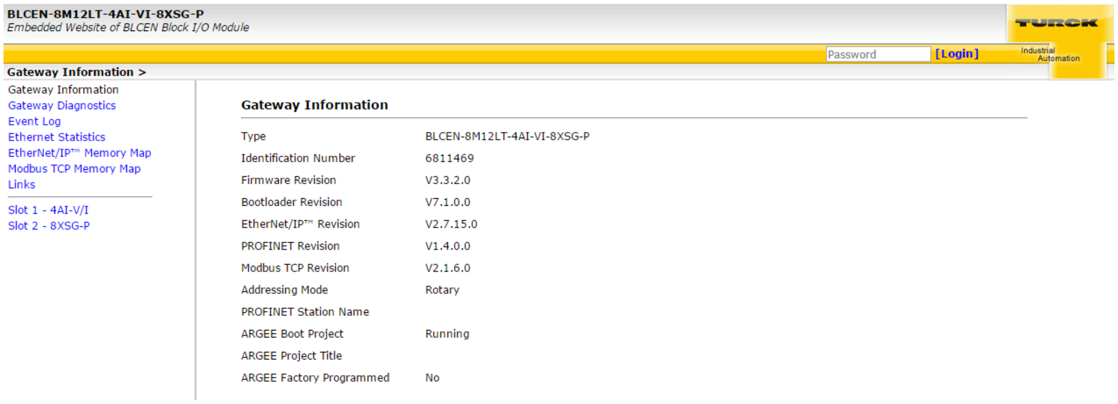
Name	Direction	Word index	Bit offset	Size	Si
argee_to_plc_reg1	ARGEE->PLC ▼	0	0 ▼	Word (16 bit) ▼	u
argee_to_plc_reg2	ARGEE->PLC ▼	1	0 ▼	Word (16 bit) ▼	u
plc_to_argee_reg1	PLC->ARGEE ▼	0	0 ▼	Word (16 bit) ▼	u
plc_to_argee_reg2	PLC->ARGEE ▼	1	0 ▼	Word (16 bit) ▼	u

State Names

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link



BLCEN-8M12LT-4AI-VI-8XSG-P
Embedded Website of BLCEN Block I/O Module

Gateway Information

Type	BLCEN-8M12LT-4AI-VI-8XSG-P
Identification Number	6811469
Firmware Revision	V3.3.2.0
Bootloader Revision	V7.1.0.0
EtherNet/IP™ Revision	V2.7.15.0
PROFINET Revision	V1.4.0.0
Modbus TCP Revision	V2.1.6.0
Addressing Mode	Rotary
PROFINET Station Name	
ARGEE Boot Project	Running
ARGEE Project Title	
ARGEE Factory Programmed	No

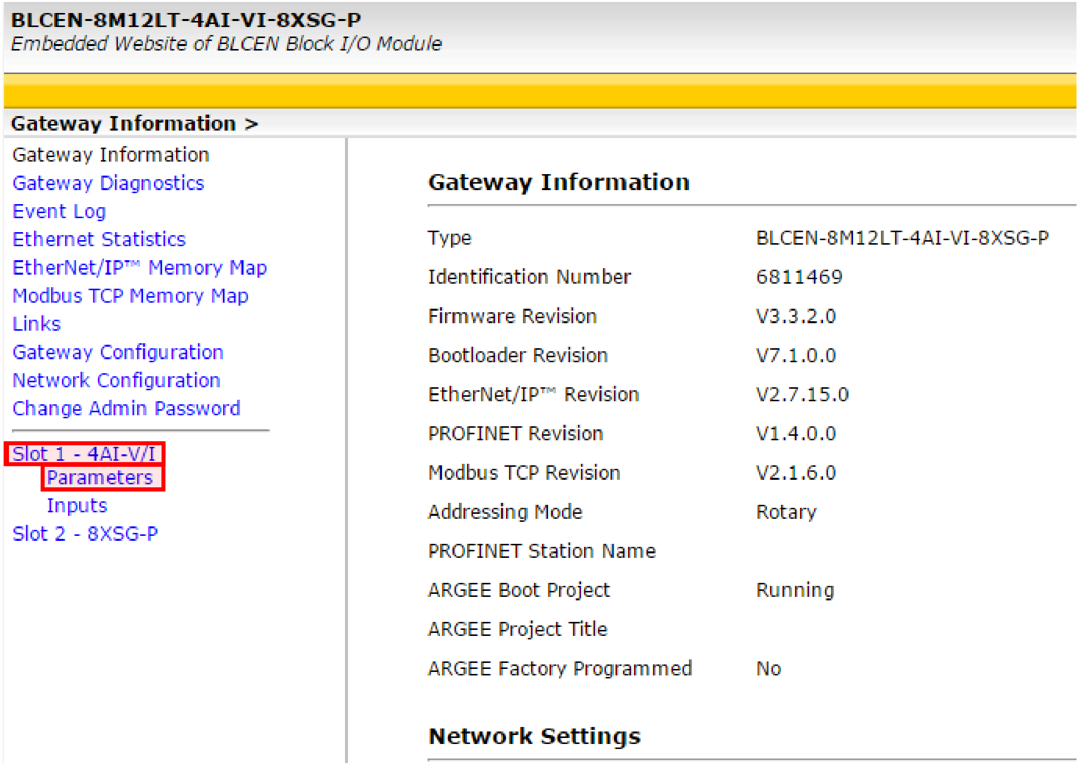
NOTE

The default password for the webserver is “password”

7.4.1 Setting Device Parameters

Once the user has logged in to the webserver they can set the Parameters that ARGEE will use.

➢ Select the Slot to access the parameters you want to change and then select Parameters.



BLCEN-8M12LT-4AI-VI-8XSG-P
Embedded Website of BLCEN Block I/O Module

Gateway Information >

- Gateway Information
- Gateway Diagnostics
- Event Log
- Ethernet Statistics
- EtherNet/IP™ Memory Map
- Modbus TCP Memory Map
- Links
- Gateway Configuration
- Network Configuration
- Change Admin Password
- Slot 1 - 4AI-V/I
  - Parameters
  - Inputs
- Slot 2 - 8XSG-P

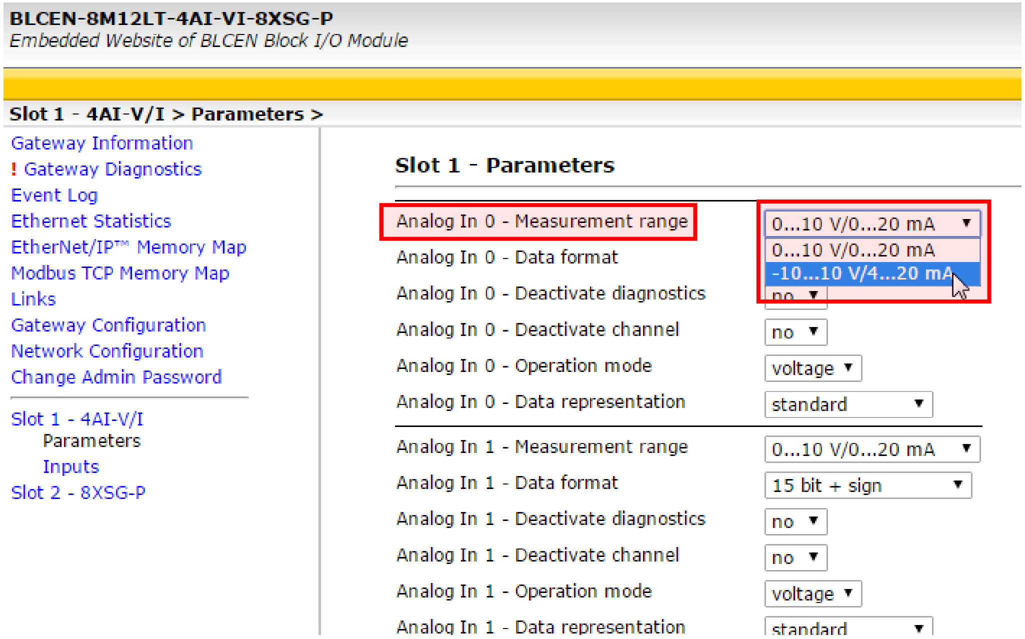
Gateway Information

Type	BLCEN-8M12LT-4AI-VI-8XSG-P
Identification Number	6811469
Firmware Revision	V3.3.2.0
Bootloader Revision	V7.1.0.0
EtherNet/IP™ Revision	V2.7.15.0
PROFINET Revision	V1.4.0.0
Modbus TCP Revision	V2.1.6.0
Addressing Mode	Rotary
PROFINET Station Name	
ARGEE Boot Project	Running
ARGEE Project Title	
ARGEE Factory Programmed	No

Network Settings

➢ Set the parameters to what your setup calls for.

In this instance we are changing the Slot-1 Analog In 0 - Measurement range to -10….10 V/4…..20mA.



BLCEN-8M12LT-4AI-VI-8XSG-P
Embedded Website of BLCEN Block I/O Module

Slot 1 - 4AI-V/I > Parameters >

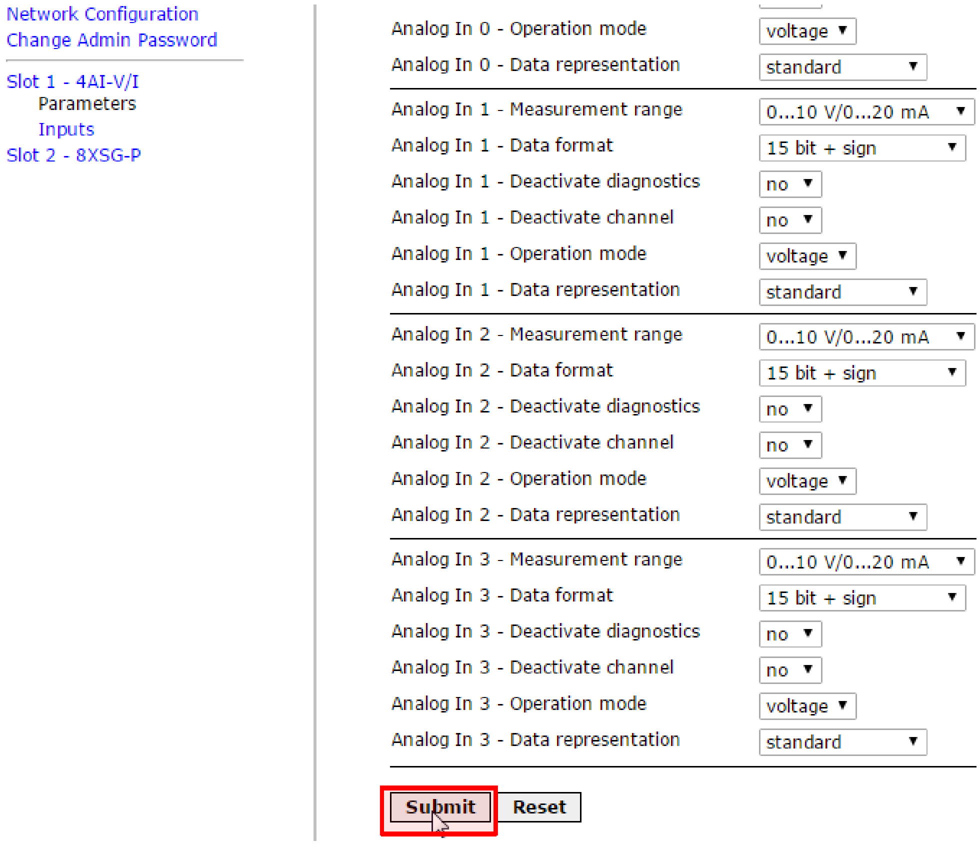
- Gateway Information
- ! Gateway Diagnostics
- Event Log
- Ethernet Statistics
- EtherNet/IP™ Memory Map
- Modbus TCP Memory Map
- Links
- Gateway Configuration
- Network Configuration
- Change Admin Password
- Slot 1 - 4AI-V/I
  - Parameters
  - Inputs
- Slot 2 - 8XSG-P

Slot 1 - Parameters

Analog In 0 - Measurement range	0...10 V/0...20 mA ▼ 0...10 V/0...20 mA -10...10 V/4...20 mA
Analog In 0 - Data format	
Analog In 0 - Deactivate diagnostics	no ▼
Analog In 0 - Deactivate channel	no ▼
Analog In 0 - Operation mode	voltage ▼
Analog In 0 - Data representation	standard ▼
Analog In 1 - Measurement range	0...10 V/0...20 mA ▼
Analog In 1 - Data format	15 bit + sign ▼
Analog In 1 - Deactivate diagnostics	no ▼
Analog In 1 - Deactivate channel	no ▼
Analog In 1 - Operation mode	voltage ▼
Analog In 1 - Data representation	standard ▼

- ➢ Scroll down to the bottom of the page and select *Submit.*

This will save your parameterization and ARGEE will load these parameters before it runs.



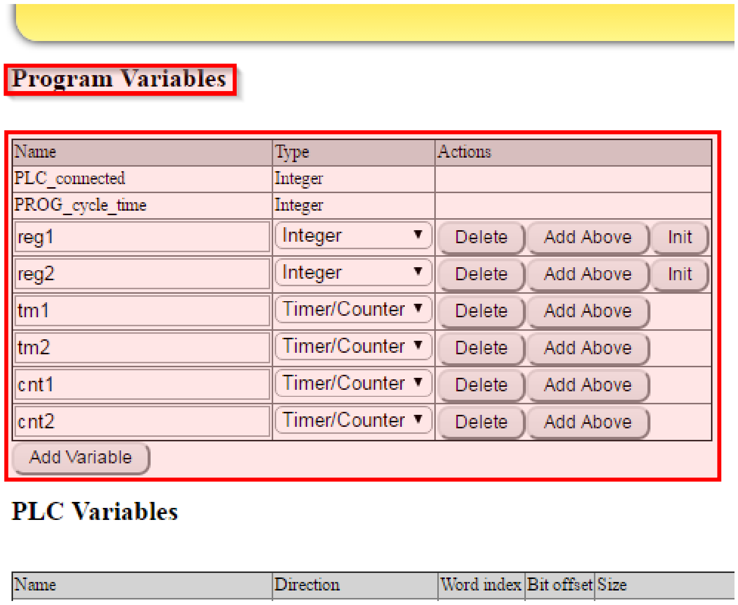
Network Configuration
Change Admin Password

Slot 1 - 4AI-V/I
Parameters
Inputs
Slot 2 - 8XSG-P

Analog In 0 - Operation mode	voltage ▾
Analog In 0 - Data representation	standard ▾
Analog In 1 - Measurement range	0...10 V/0...20 mA ▾
Analog In 1 - Data format	15 bit + sign ▾
Analog In 1 - Deactivate diagnostics	no ▾
Analog In 1 - Deactivate channel	no ▾
Analog In 1 - Operation mode	voltage ▾
Analog In 1 - Data representation	standard ▾
Analog In 2 - Measurement range	0...10 V/0...20 mA ▾
Analog In 2 - Data format	15 bit + sign ▾
Analog In 2 - Deactivate diagnostics	no ▾
Analog In 2 - Deactivate channel	no ▾
Analog In 2 - Operation mode	voltage ▾
Analog In 2 - Data representation	standard ▾
Analog In 3 - Measurement range	0...10 V/0...20 mA ▾
Analog In 3 - Data format	15 bit + sign ▾
Analog In 3 - Deactivate diagnostics	no ▾
Analog In 3 - Deactivate channel	no ▾
Analog In 3 - Operation mode	voltage ▾
Analog In 3 - Data representation	standard ▾

7.5 Program Variables

Program Variables can be added, deleted and renamed. The user can also change the variable type by using the drop down arrow.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▾	Delete Add Above Init
reg2	Integer ▾	Delete Add Above Init
tm1	Timer/Counter ▾	Delete Add Above
tm2	Timer/Counter ▾	Delete Add Above
cnt1	Timer/Counter ▾	Delete Add Above
cnt2	Timer/Counter ▾	Delete Add Above

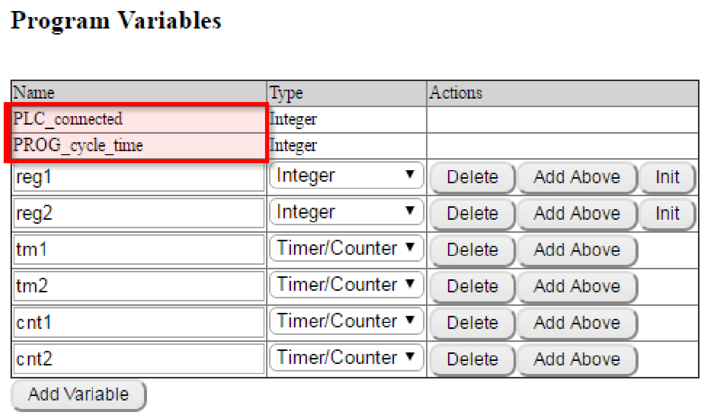
PLC Variables

Name	Direction	Word index	Bit offset	Size

7.5.1 PLC_connected & PROG_cycle_time

The PLC_connected bit is true when a PLC is connected to the device.

The PROG_cycle_time displays the time it takes to execute the entire program.

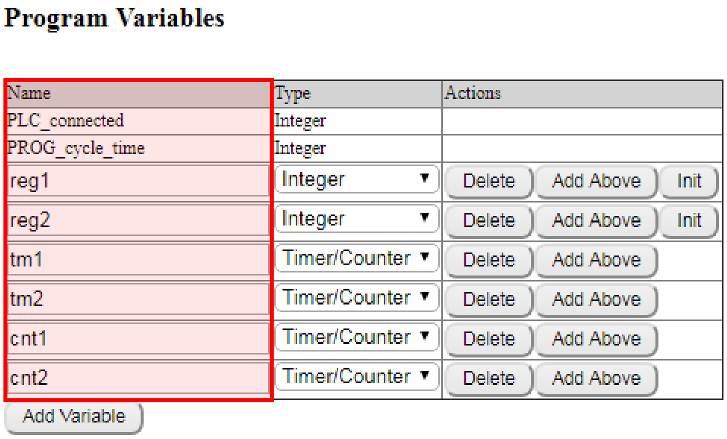


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

7.5.2 Program Variable Names

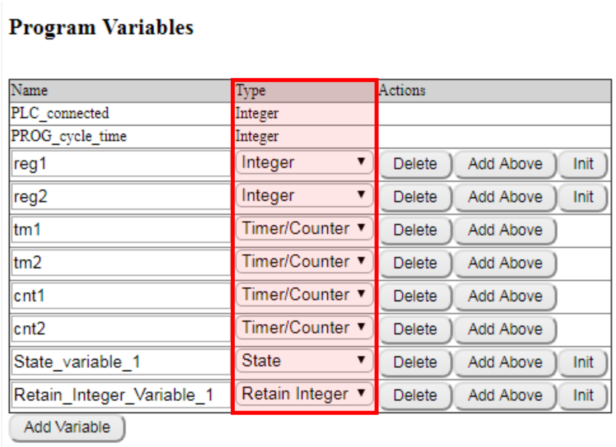
Variable Names are the names of variables in the users program.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above

7.5.3 Program Variable Types

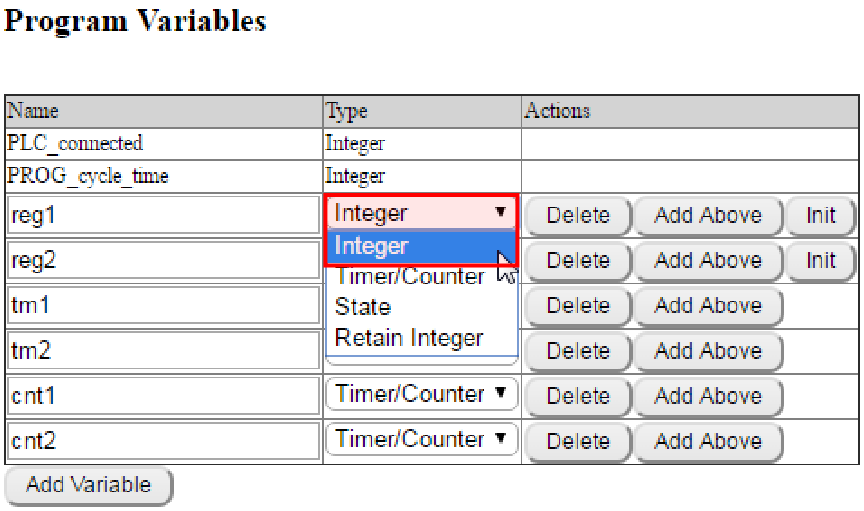


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above
State_variable_1	State	Delete Add Above Init
Retain_Integer_Variable_1	Retain Integer	Delete Add Above Init

7.5.3.1 Integer

If the user selects *Integer*, the Program Variable will be stored in 32 bit signed register. This allows the user to store an Integer value between +2,147,483,647 and -2,147,483,647 in the Program Variable’s register.



Program Variables

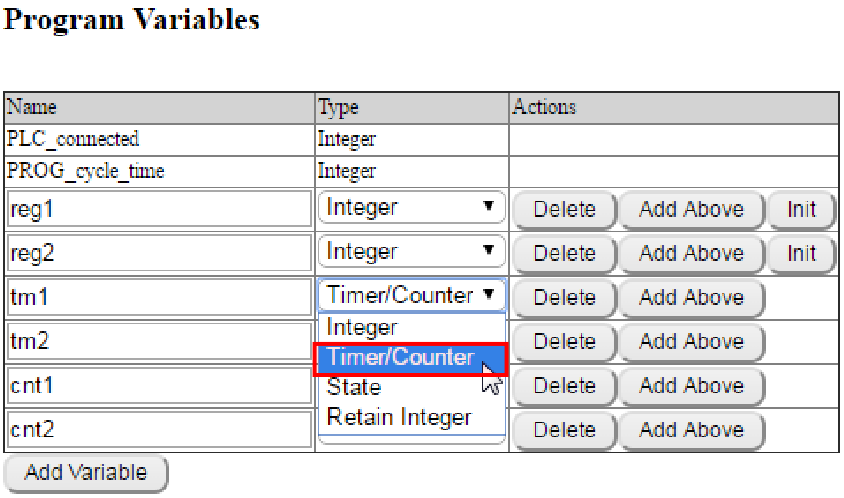
Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete Add Above Init
reg2		Delete Add Above Init
tm1		Delete Add Above
tm2		Delete Add Above
cnt1	Timer/Counter ▼	Delete Add Above
cnt2	Timer/Counter ▼	Delete Add Above

NOTE

Integer values are stored in four 8-bit registers.

7.5.3.2 Timer/Counter

The user can select *Timer/Counter* if they want to add a Timer or Counter variable to their program.

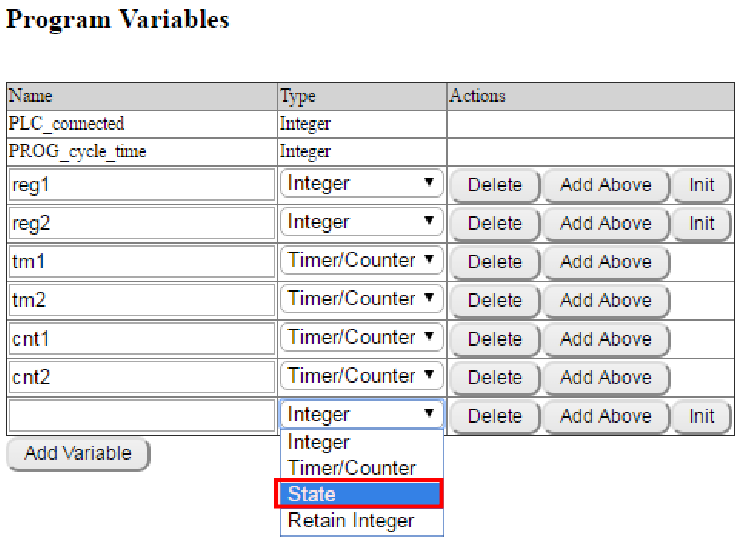


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete Add Above Init
reg2	Integer ▼	Delete Add Above Init
tm1	Timer/Counter ▼	Delete Add Above
tm2		Delete Add Above
cnt1		Delete Add Above
cnt2		Delete Add Above

7.5.3.3 State

The user would select *State* if they wanted to create a State Variable. State Variables are used in State Machines. An example of a State Machine is shown 14.4.1 State Machine.

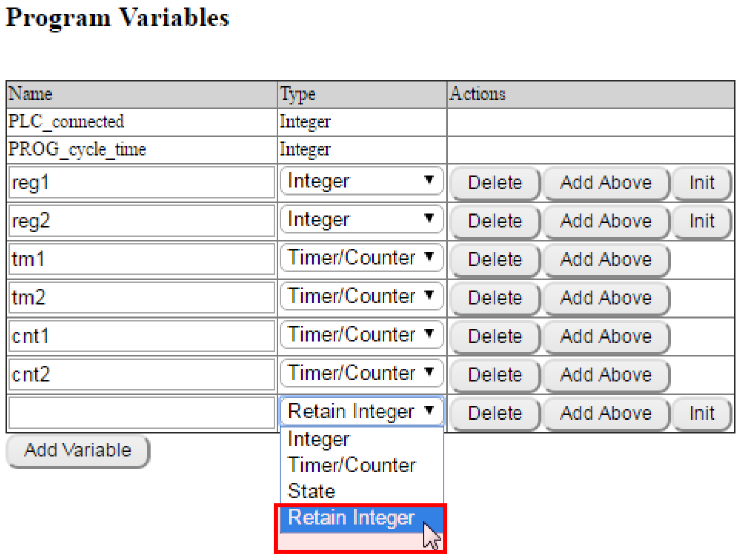


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above
	Integer	Delete Add Above Init

7.5.3.4 Retain Integer

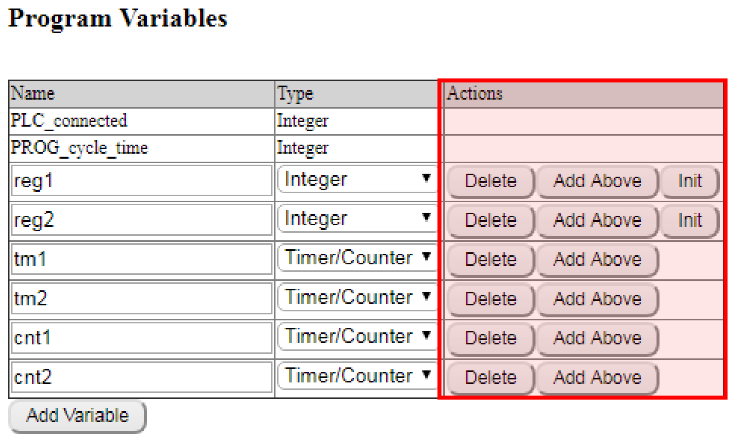
The user would use *Retain Integer* if they wanted to save the value in a Program Variable through a power cycle. The value is saved into flash memory once every three minutes if the value has been changed.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer	Delete Add Above Init
reg2	Integer	Delete Add Above Init
tm1	Timer/Counter	Delete Add Above
tm2	Timer/Counter	Delete Add Above
cnt1	Timer/Counter	Delete Add Above
cnt2	Timer/Counter	Delete Add Above
	Retain Integer	Delete Add Above Init

7.5.4 Program Variable Actions

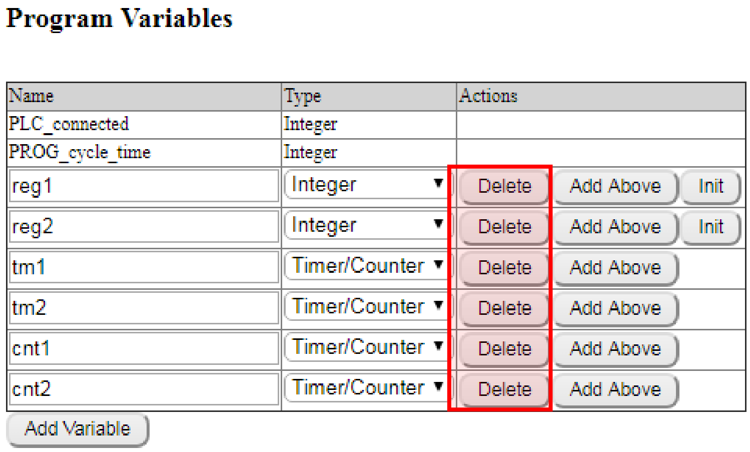


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▾	Delete · Add Above · Init
reg2	Integer ▾	Delete · Add Above · Init
tm1	Timer/Counter ▾	Delete · Add Above
tm2	Timer/Counter ▾	Delete · Add Above
cnt1	Timer/Counter ▾	Delete · Add Above
cnt2	Timer/Counter ▾	Delete · Add Above

7.5.4.1 Delete

The *Delete* button will delete the program variable.

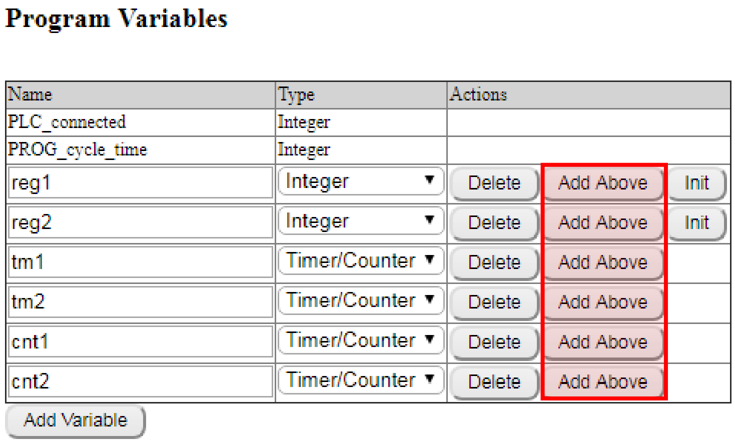


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▾	Delete · Add Above · Init
reg2	Integer ▾	Delete · Add Above · Init
tm1	Timer/Counter ▾	Delete · Add Above
tm2	Timer/Counter ▾	Delete · Add Above
cnt1	Timer/Counter ▾	Delete · Add Above
cnt2	Timer/Counter ▾	Delete · Add Above

7.5.4.2 Add Above

The *Add Above* button will add a Program Variable above the selected variable

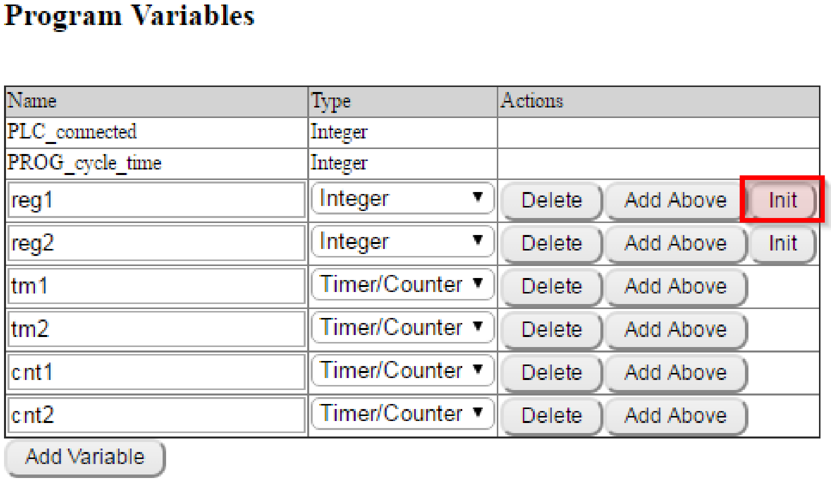


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▾	Delete · Add Above · Init
reg2	Integer ▾	Delete · Add Above · Init
tm1	Timer/Counter ▾	Delete · Add Above
tm2	Timer/Counter ▾	Delete · Add Above
cnt1	Timer/Counter ▾	Delete · Add Above
cnt2	Timer/Counter ▾	Delete · Add Above

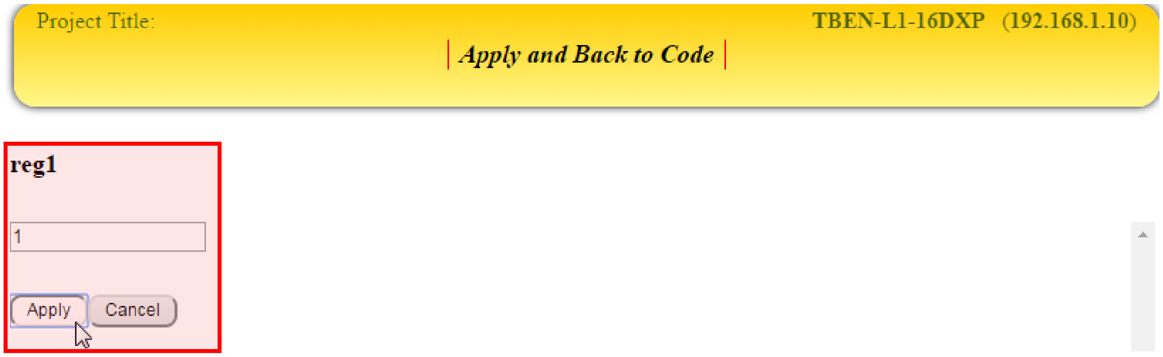
7.5.4.3 Init (Initialize)

The user will use *Initialize* if they want to pre-set the value in a Program Variable's register.



Program Variables

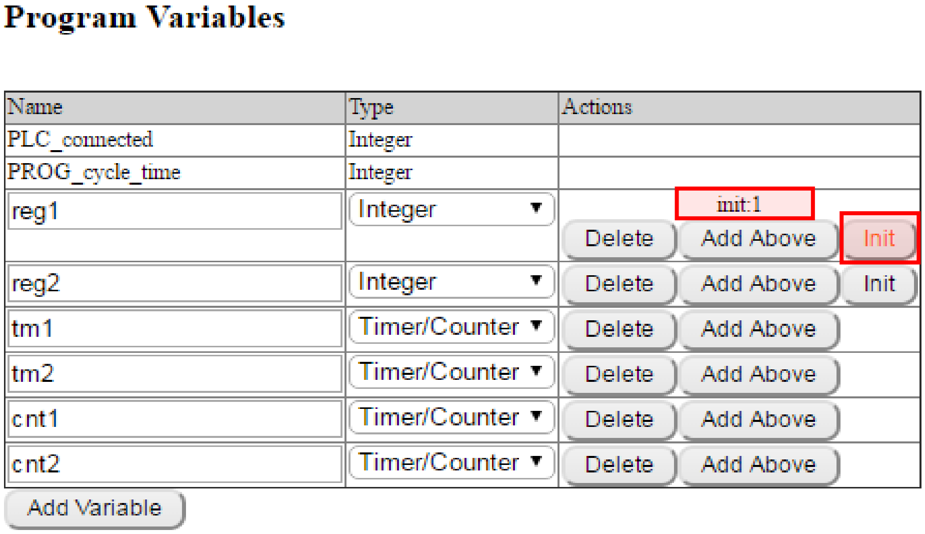
Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete · Add Above · Init
reg2	Integer ▼	Delete · Add Above · Init
tm1	Timer/Counter ▼	Delete · Add Above
tm2	Timer/Counter ▼	Delete · Add Above
cnt1	Timer/Counter ▼	Delete · Add Above
cnt2	Timer/Counter ▼	Delete · Add Above



Project Title: TBEN-L1-16DXP (192.168.1.10)

reg1

1

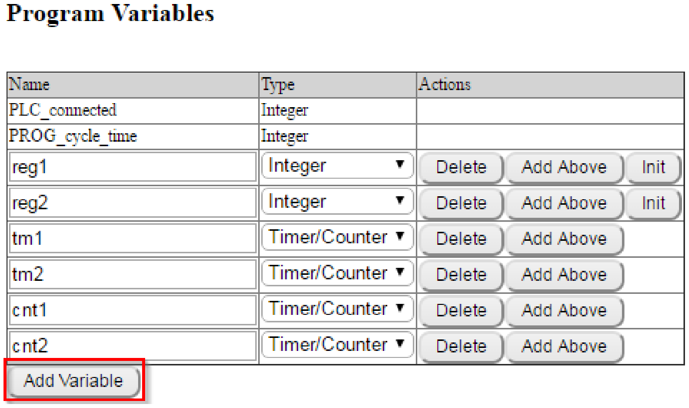


Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	init:1 Delete · Add Above · Init
reg2	Integer ▼	Delete · Add Above · Init
tm1	Timer/Counter ▼	Delete · Add Above
tm2	Timer/Counter ▼	Delete · Add Above
cnt1	Timer/Counter ▼	Delete · Add Above
cnt2	Timer/Counter ▼	Delete · Add Above

7.5.5 Add Variable

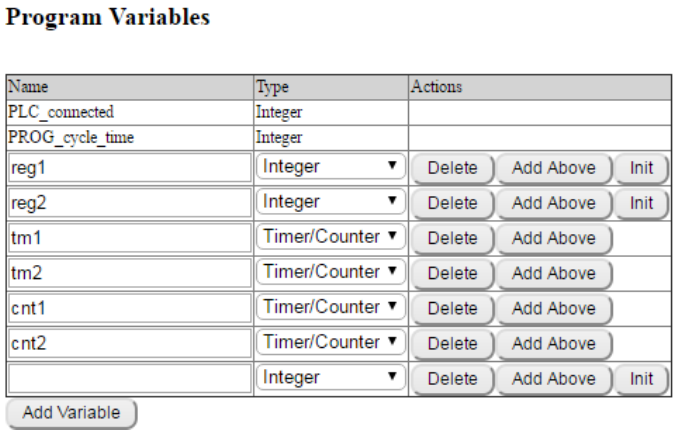
The *Add Variable* button will add a Program Variable to the program.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete · Add Above · Init
reg2	Integer ▼	Delete · Add Above · Init
tm1	Timer/Counter ▼	Delete · Add Above
tm2	Timer/Counter ▼	Delete · Add Above
cnt1	Timer/Counter ▼	Delete · Add Above
cnt2	Timer/Counter ▼	Delete · Add Above

Add Variable



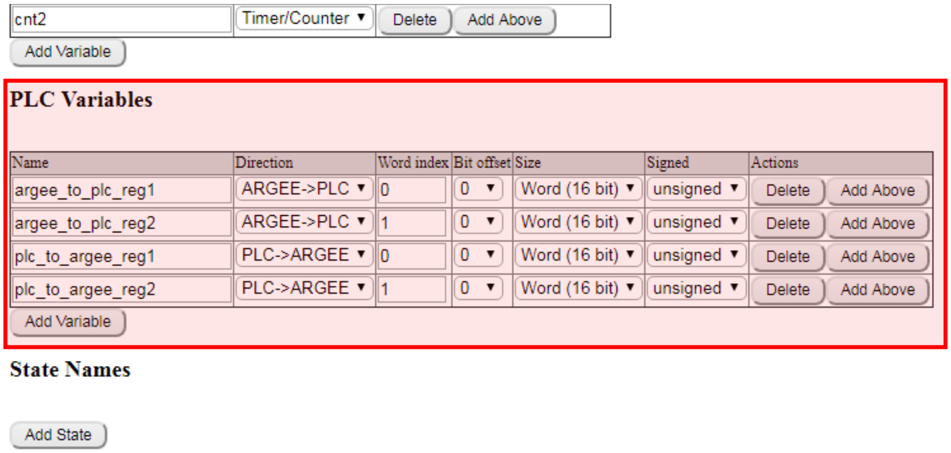
Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete · Add Above · Init
reg2	Integer ▼	Delete · Add Above · Init
tm1	Timer/Counter ▼	Delete · Add Above
tm2	Timer/Counter ▼	Delete · Add Above
cnt1	Timer/Counter ▼	Delete · Add Above
cnt2	Timer/Counter ▼	Delete · Add Above
	Integer ▼	Delete · Add Above · Init

Add Variable

7.6 PLC Variables

PLC Variables are used to define communication between the ARGEE block and the PLC. PLC examples are shown in chapter 14 Common Applications.



cnt2	Timer/Counter ▼	Delete · Add Above

Add Variable

PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete · Add Above
argee_to_plc_reg2	ARGEE->PLC ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete · Add Above
plc_to_argee_reg1	PLC->ARGEE ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete · Add Above
plc_to_argee_reg2	PLC->ARGEE ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete · Add Above

Add Variable

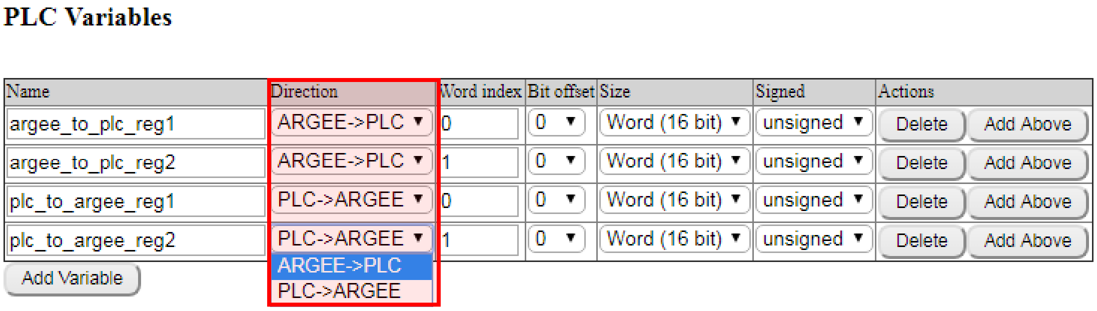
State Names

Add State

7.6.1 Direction

The user will use the *Direction* dropdown arrow to assign which direction the data is traveling.

- *ARGEE->PLC* means the data is traveling from the ARGEE block to the PLC.
- *PLC->ARGEE* means the data is traveling from the PLC to the ARGEE block.

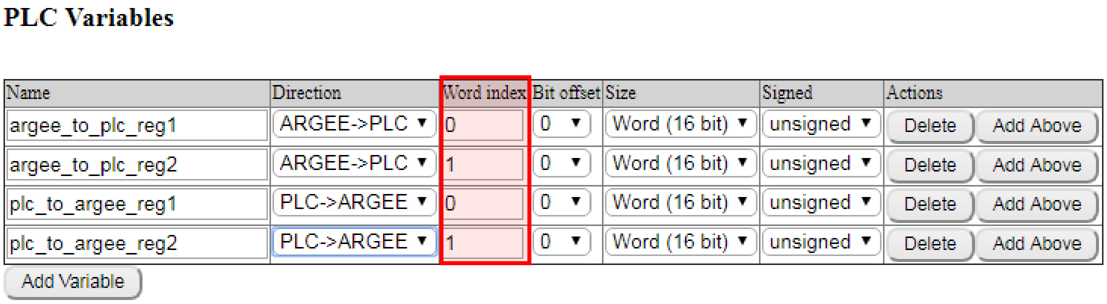


PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

7.6.2 Word Index

The user will use the *Word index* to assign the data to a specific register in the PLC or the ARGEE block.

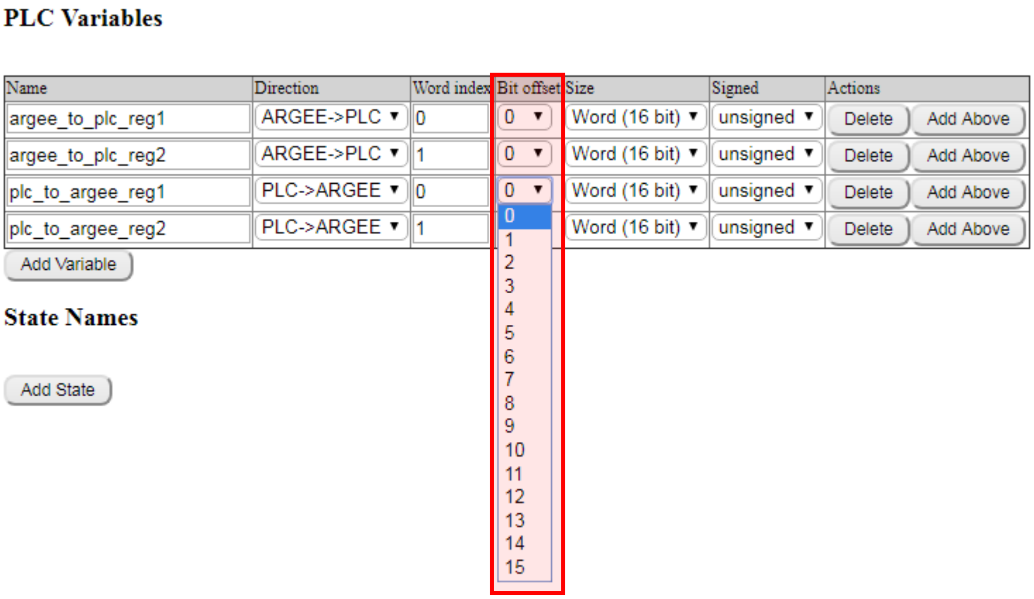


PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

7.6.3 Bit Offset

The user will use the *Bit offset* to assign the data to a specific bit in a register.



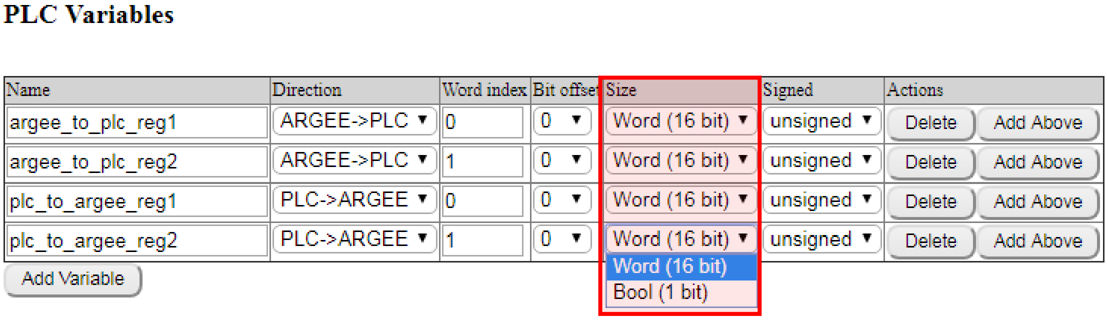
PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1		Word (16 bit)	unsigned	Delete Add Above

State Names

7.6.4 Size

The user will use the *Size* drop down to set the size of the data being transferred.

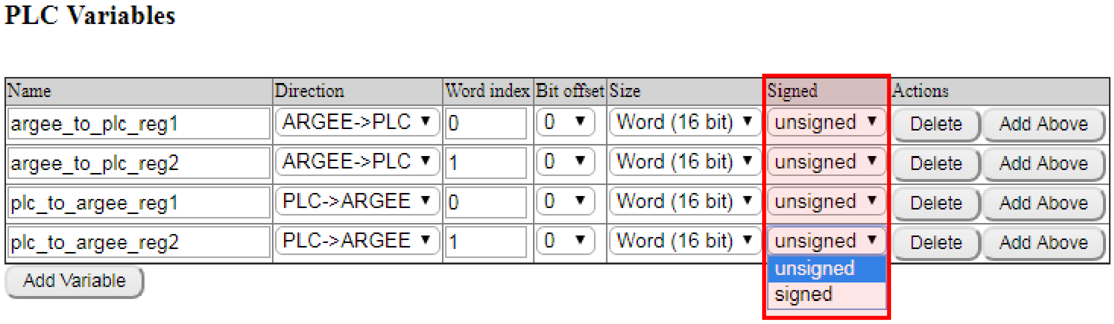


PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

7.6.5 Signed

The user will use the *Signed* drop down to indicate whether the data being transferred is a signed or unsigned integer.

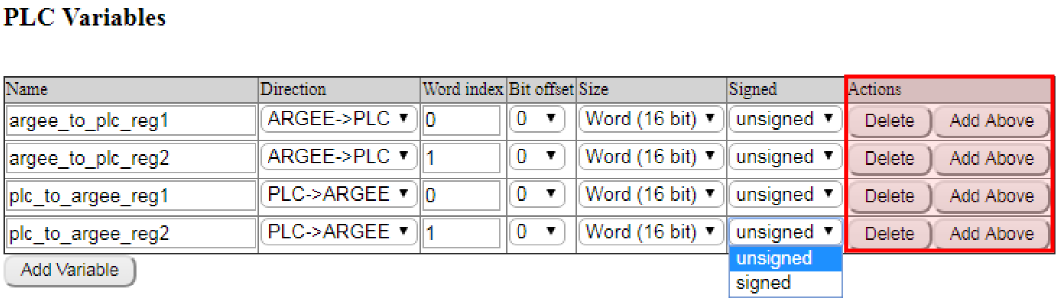


PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

7.6.6 Actions

The Delete and Add Above actions are discussed in section 7.5.4 Program Variable Actions.

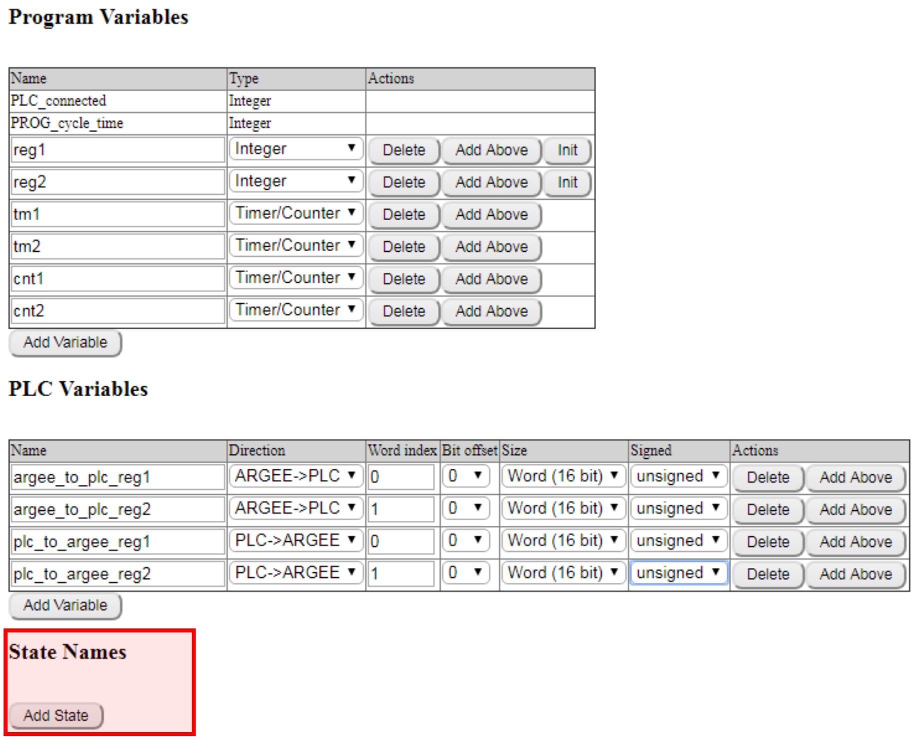


PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC	1	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE	1	0	Word (16 bit)	unsigned	Delete Add Above

7.7 State Names

State Names are used to make it easier to identify which State the users program is in. “State” is the term used to identify a specific program operation at a specific moment.



Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
reg1	Integer ▼	Delete Add Above Init
reg2	Integer ▼	Delete Add Above Init
tm1	Timer/Counter ▼	Delete Add Above
tm2	Timer/Counter ▼	Delete Add Above
cnt1	Timer/Counter ▼	Delete Add Above
cnt2	Timer/Counter ▼	Delete Add Above

Add Variable

PLC Variables

Name	Direction	Word index	Bit offset	Size	Signed	Actions
argee_to_plc_reg1	ARGEE->PLC ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above
argee_to_plc_reg2	ARGEE->PLC ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above
plc_to_argee_reg1	PLC->ARGEE ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above
plc_to_argee_reg2	PLC->ARGEE ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above

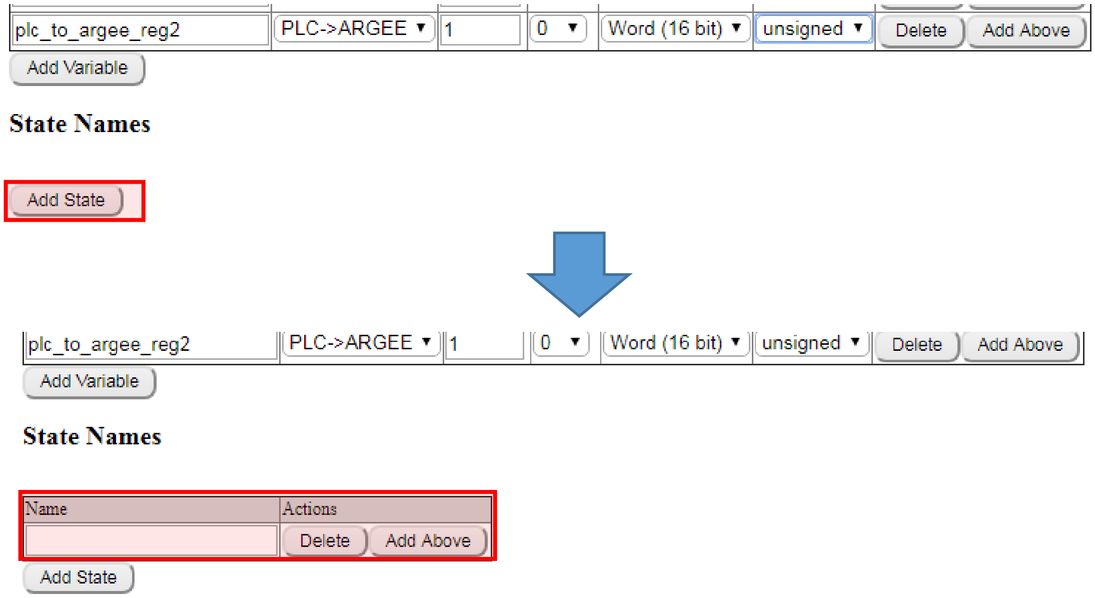
Add Variable

State Names

Add State

7.7.1 Add State

The user would click *Add State* if they wanted to add State Names to their program.



plc_to_argee_reg2	PLC->ARGEE ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above

Add Variable

State Names

Add State

plc_to_argee_reg2	PLC->ARGEE ▼	1	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above

Add Variable

State Names

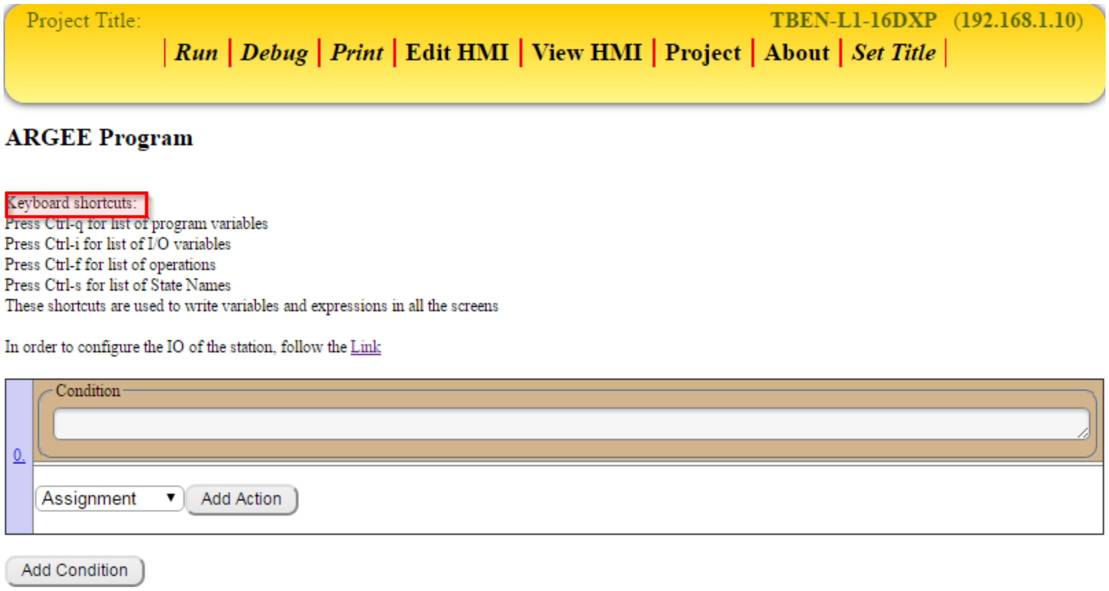
Name	Actions
	Delete Add Above

Add State

7.8 Keyboard Shortcuts

Keyboard shortcuts provide the user with a quick way to access Program Variables, I/O Variables, Operations and State Names.

- Ctrl-q, Program Variables.
- Ctrl-i, I/O Variables.
- Ctrl-f, Operations
- Ctrl-s, State names



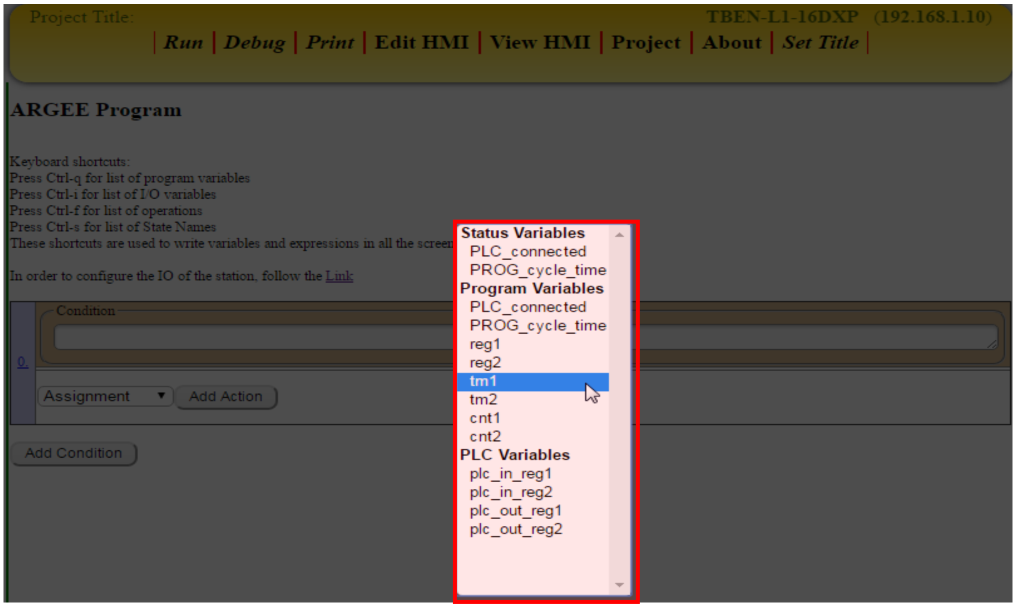
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

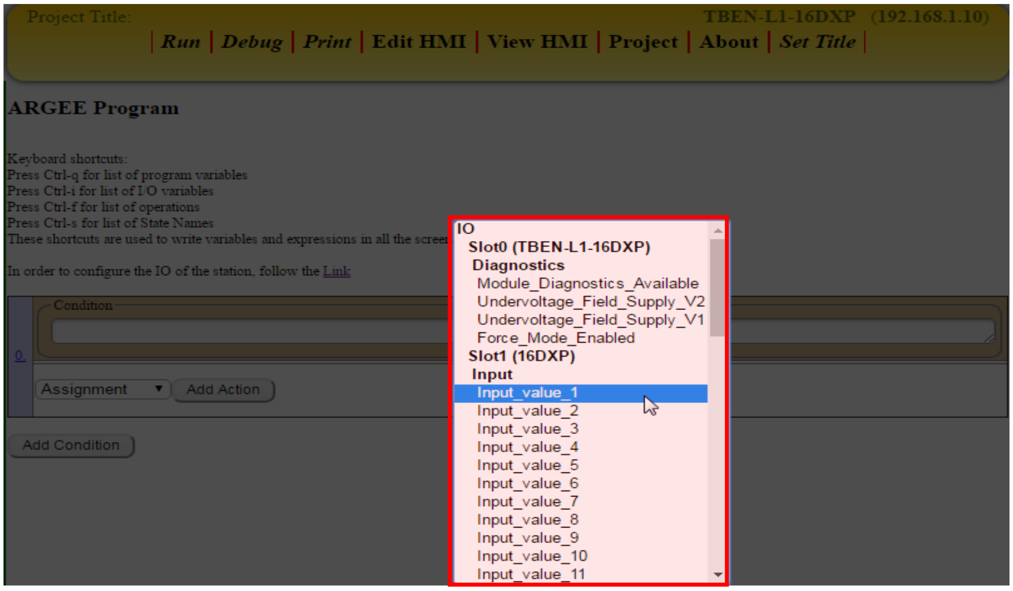
7.8.1 Program Variables (Control + Q)

If the user presses *Ctrl + q* while in a Condition or Action box, the Program Variable List will pop up. The user can select their desired variable and it will be added to their respective Condition or Action box.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run
Debug
Print
Edit HMI
View HMI
Project
About
Set Title
ARGEE Program
Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screen
In order to configure the IO of the station, follow the Link
Condition
0.
Assignment
Add Action
Add Condition
Status Variables
PLC_connected
PROG_cycle_time
Program Variables
PLC_connected
PROG_cycle_time
reg1
reg2
tm1
tm2
cnt1
cnt2
PLC Variables
plc_in_reg1
plc_in_reg2
plc_out_reg1
plc_out_reg2

7.8.2 I/O Variables (Control + I)

If the user presses *Ctrl + i* while in a Condition or Action box, the I/O Variable List will pop up. The user can select their desired variable and it will be added to their respective Condition or Action box.



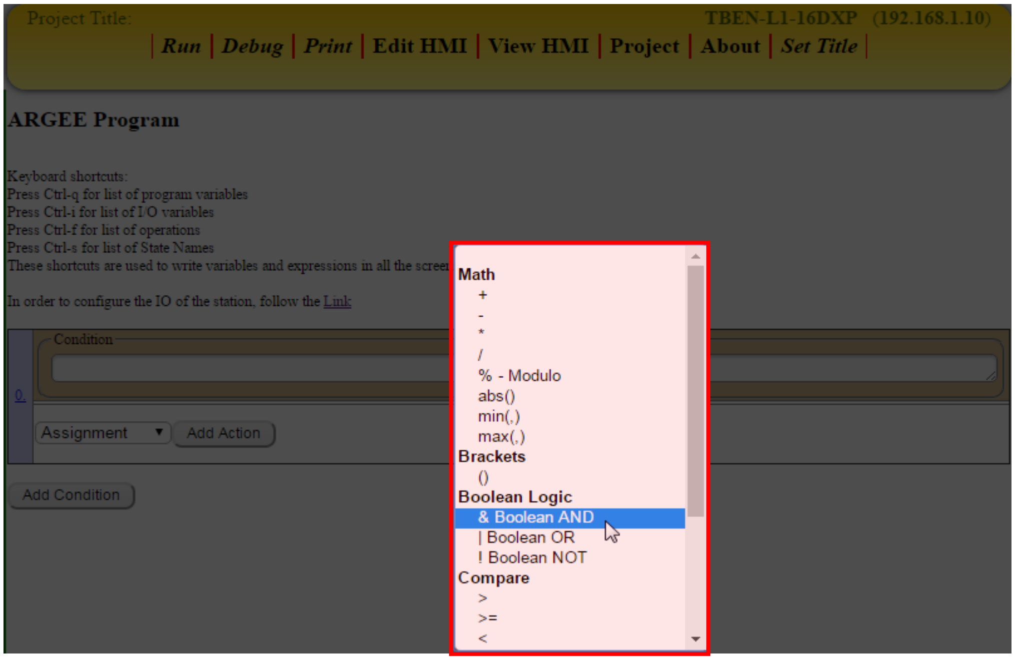
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screer

In order to configure the IO of the station, follow the Link

7.8.3 Operations (Control + F)

If the user presses *Ctrl + f* while in a Condition or Action box, the Operations List will pop up. The user can select their desired operation and it will be added to their respective Condition or Action box.



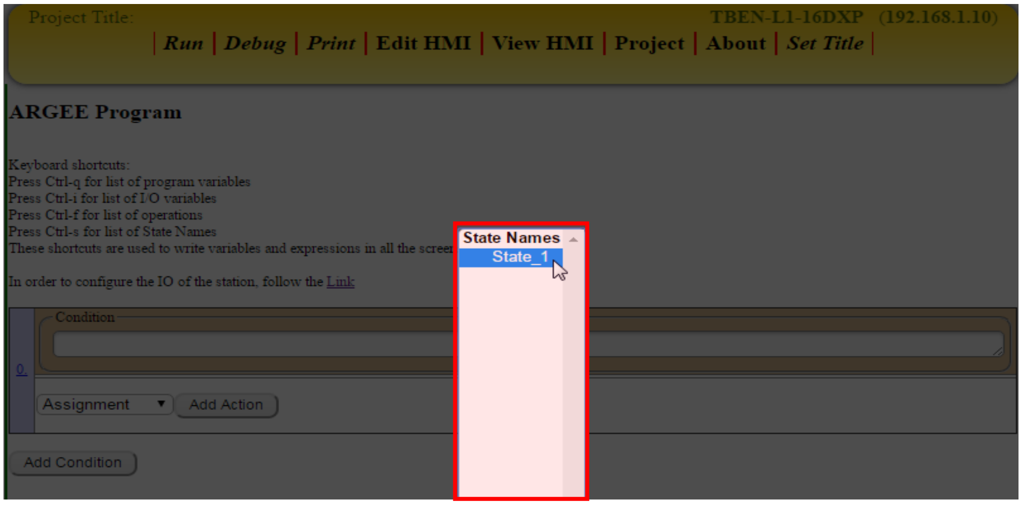
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screer

In order to configure the IO of the station, follow the Link

7.8.4 State Names (Control + S)

If the user presses *Ctrl + s* while in a Condition or Action box, the State Name List will pop up. The user can select their desired State Name and it will be added to their respective Condition or Action box.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
| Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title |
State Names
State_1
Condition
0.
Assignment
Add Action
Add Condition

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screen

In order to configure the IO of the station, follow the Link

NOTE

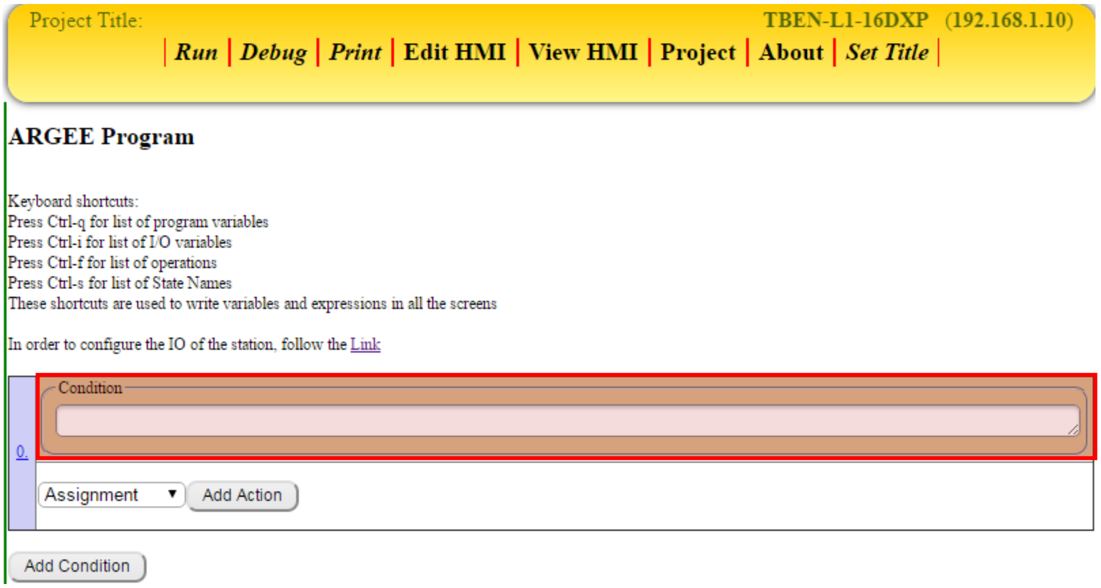
If the user has not added a State Name to their project, the State Name List will be empty.

8 Conditions & Actions

8.1 Conditions

The Condition box is where the user puts their input conditions. An example of an Input condition could be:

- A Timer expiring
- A Counter reaching a specific value
- A Counter expiring
- A register value changing from 0 to 1
- A register value changing from 1 to 0
- An Input from a sensor becoming true
- …many other things can also be used as an input condition


Project Title:
TBEN-L1-16DXP (192.168.1.10)
| Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title |

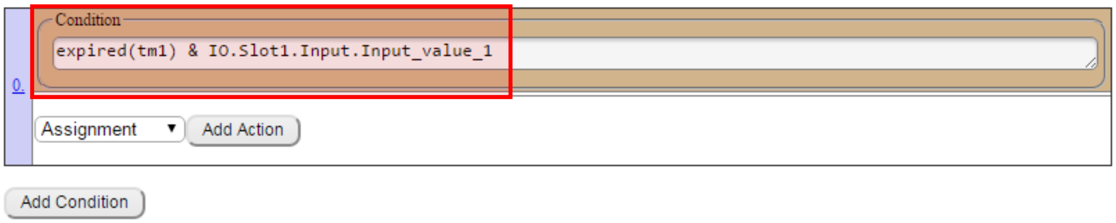
ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

Condition
0.
Assignment ▾
Add Action
Add Condition

The Condition box also allows the user to combine different types of Inputs.



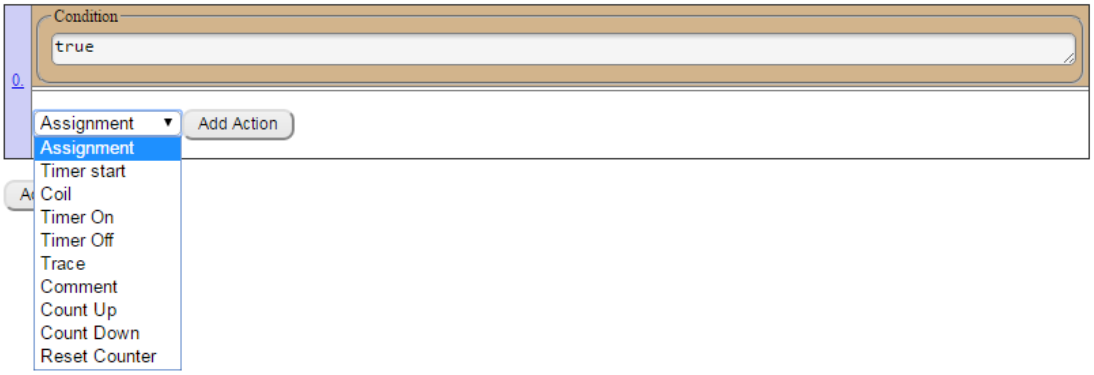
```
expired(tm1) & IO.Slot1.Input.Input_value_1
```

Explaining the Screenshot: The above Condition will only become true when timer 1 expires and Input_value_1 goes true.

8.2 Actions

The Actions box is where the user puts their Output conditions. The user can execute several Actions under a single Condition statement. An Action could be:

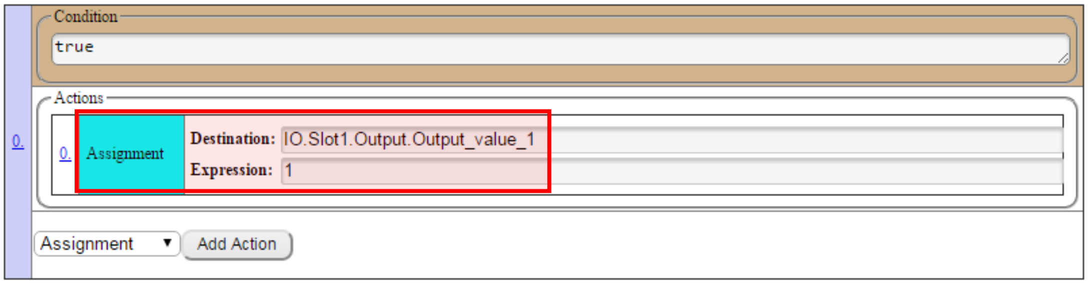
- Loading a value into a register
- Stating a Timer
- Stopping a Timer
- Signal Tracing
- Incrementing a Counter
- Decrementing a Counter
- Resetting a Counter



Condition: true

8.2.1 Assignment

The user would use the *Assignment* action if they want to load a value into a register.



Condition: true

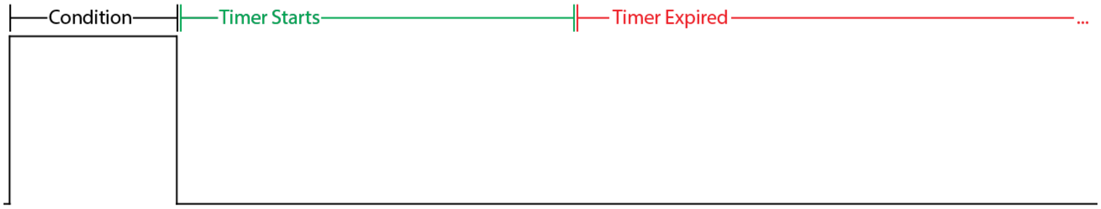
Destination: IO.Slot1.Output.Output_value_1

Expression: 1

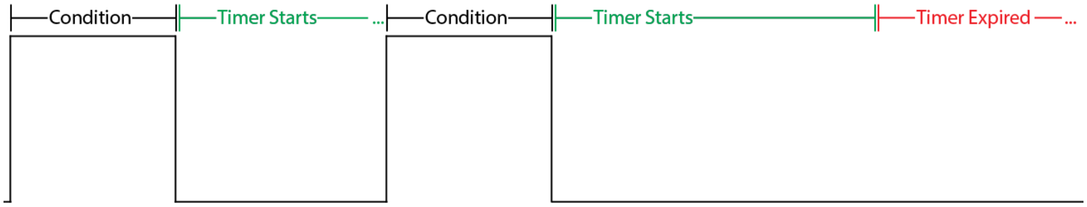
Explaining the Example: The Condition in the above statement is always “true”. The value “1” is loaded into register Output_value_1. In other words, this means that the user’s Output 1 will always be on.

8.2.2 Timer Start

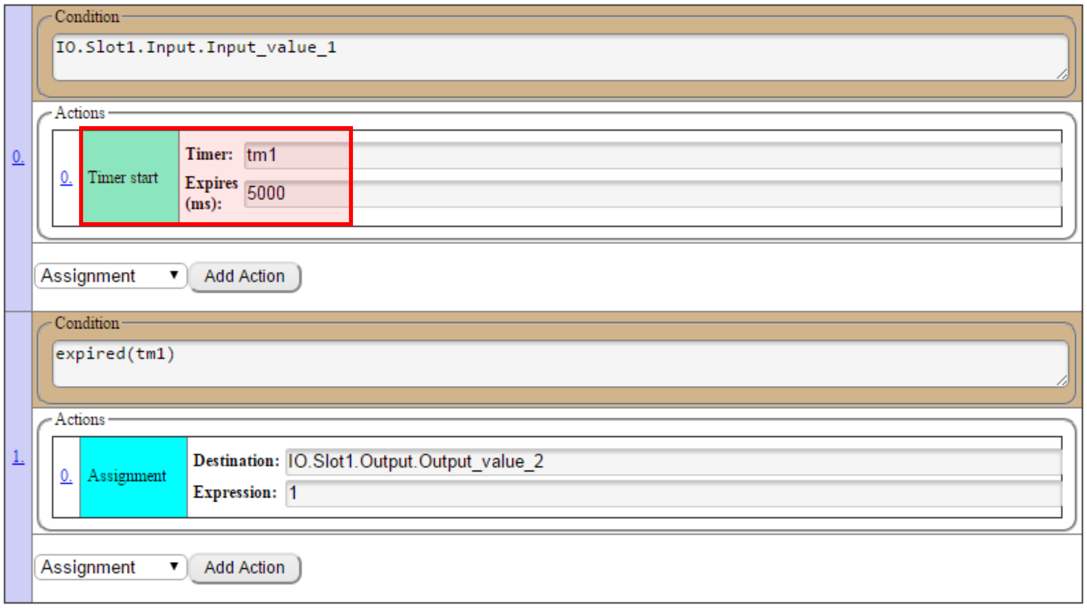
The user will use the *Timer Start* action if they want to start a timer after the Condition has occurred.



If the Condition occurs again before the timer expires, the timer will restart.



Example of Timer Start:



0\.

Condition

IO.Slot1.Input.Input_value_1

Actions

0. Timer start | Timer: tm1 | Expires (ms): 5000

1\.

Condition

expired(tm1)

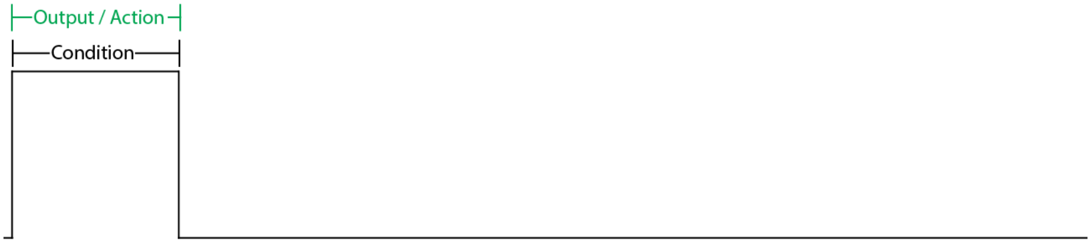
Actions

0. Assignment | Destination: IO.Slot1.Output.Output_value_2 | Expression: 1

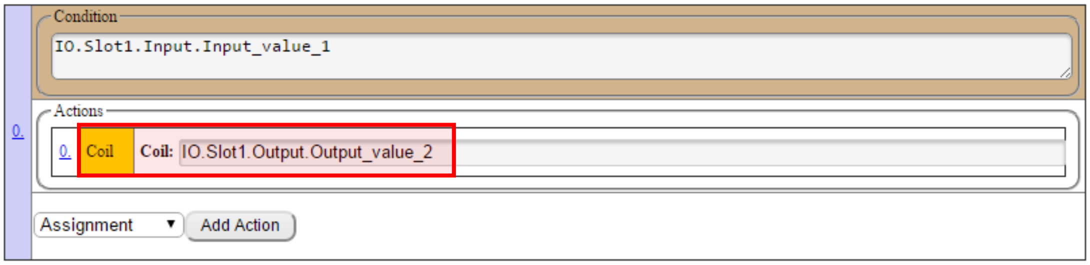
Explaining the Example: When Input_value_1 goes true and then false, start timer 1. When timer 1 expires, load the value “1” into register Output_value_2 (or turn on Output 2).

8.2.3 Coil

The user will use the *Coil* action if they want an Output to be “set” if the Condition is true and “cleared” when the Condition is false.



Example of Coil:


Condition

```
IO.Slot1.Input.Input_value_1
```

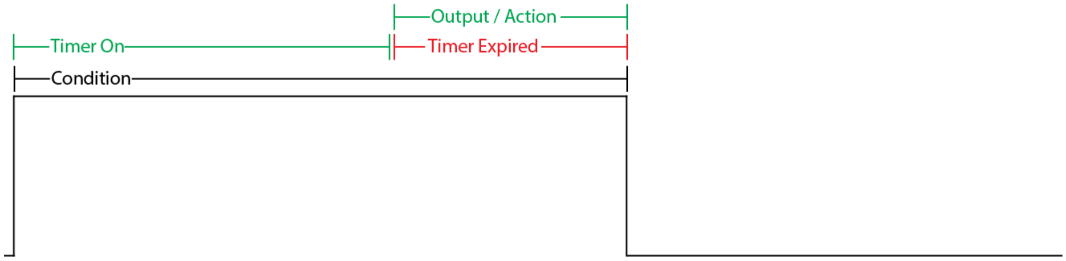
Actions

0. Coil **Coil:** IO.Slot1.Output.Output_value_2

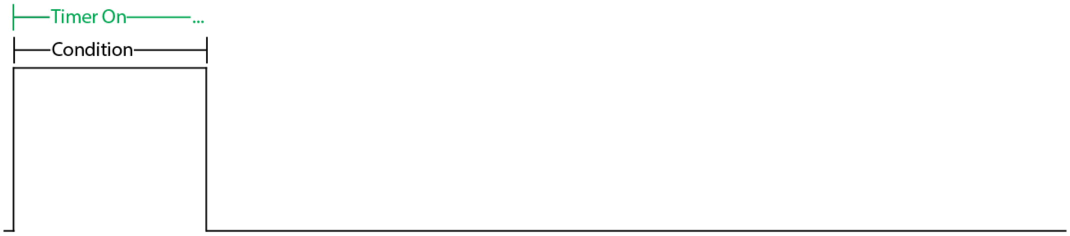
Explaining the Example: When Input_value_1 is true, Output_value_2 is true. When Input_value_1 is false, Output_value_2 is false.

8.2.4 Timer On

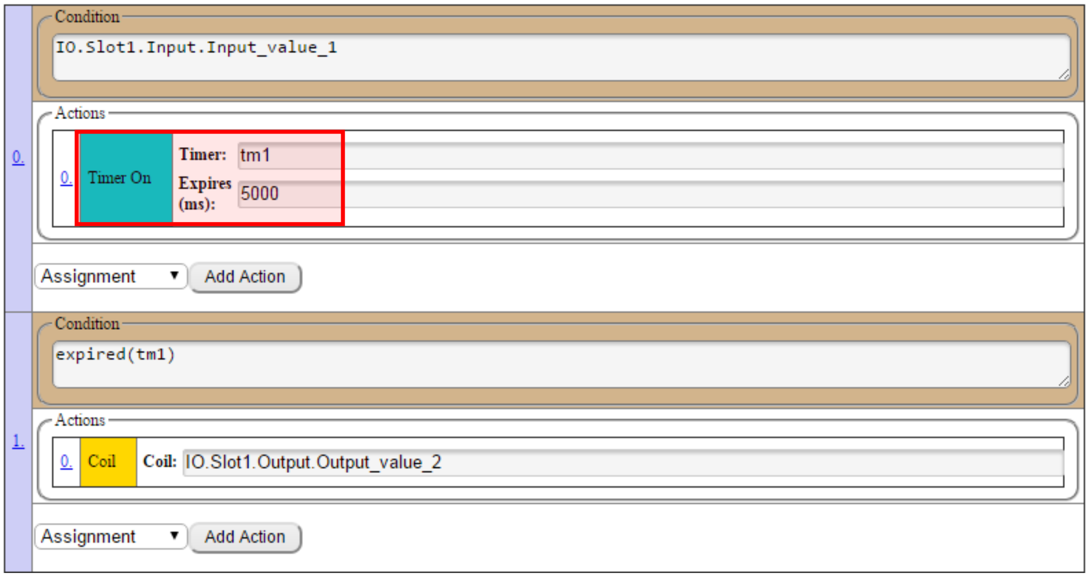
The user will use the Timer On action if they want a timer to run while a Condition is true. The user will normally tie an additional Action or Output to the timer expired Condition.



If the Condition ends before the timer expires, the Action tied to the expired timer will not occur.



Example of Timer On:



0.

Condition

```
IO.Slot1.Input.Input_value_1
```

Actions

0. Timer On — Timer: tm1 — Expires (ms): 5000

1.

Condition

```
expired(tm1)
```

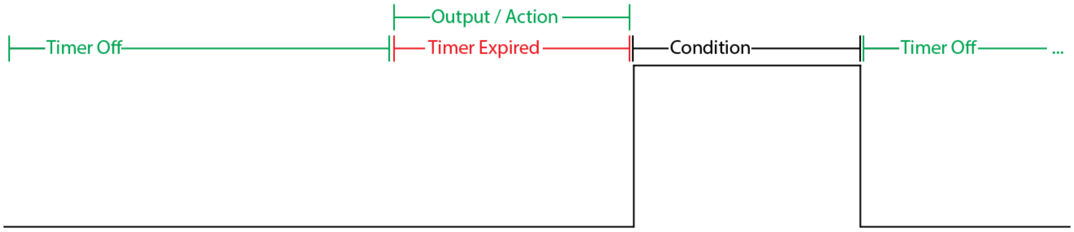
Actions

0. Coil — Coil: IO.Slot1.Output.Output_value_2

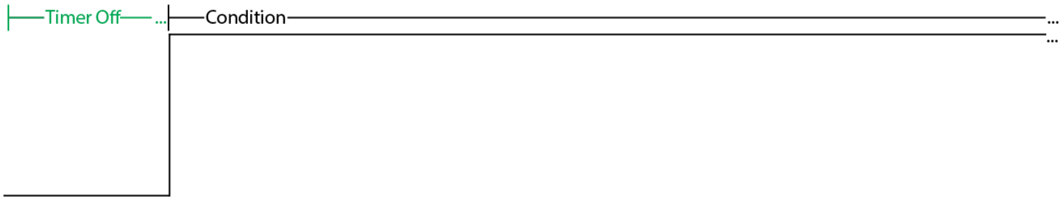
Explaining the Example: When Input_value_1 is true, start timer 1. When timer 1 expires, coil Output_value_2. When Input_value_1 is false, Output_value_2 will be false.

8.2.5 Timer Off

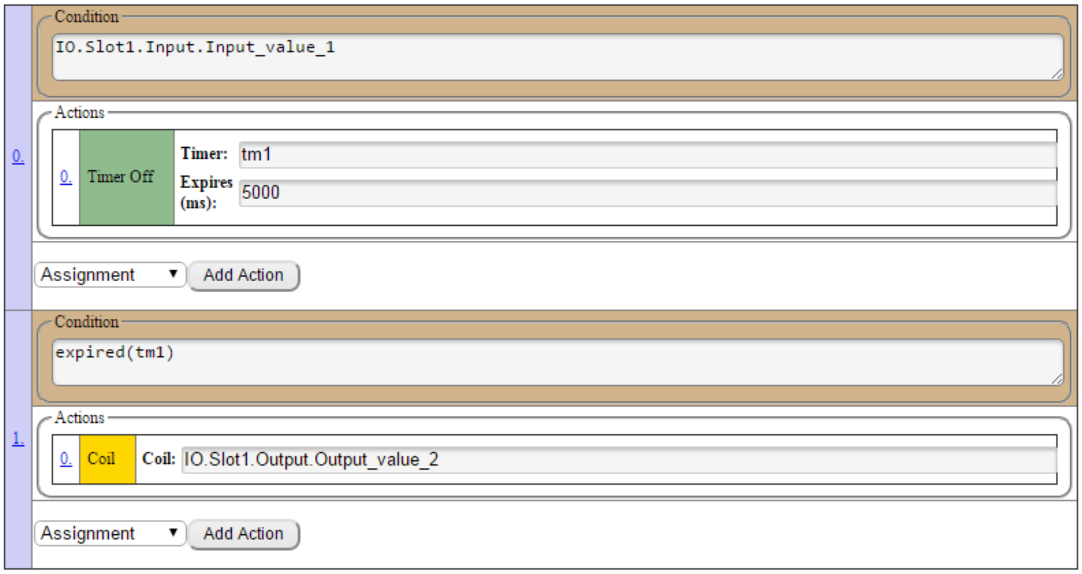
The user will use the Timer Off action if they want a timer to run while a Condition is false. The user will normally tie an additional Action or Output to the timer expired Condition.



If the Condition starts before the timer expires, the Action tied to the expired timer will not occur.



Example of Timer Off:



```
IO.Slot1.Input.Input_value_1
```

Timer: tm1
Expires (ms): 5000

```
expired(tm1)
```

Coil: IO.Slot1.Output.Output_value_2

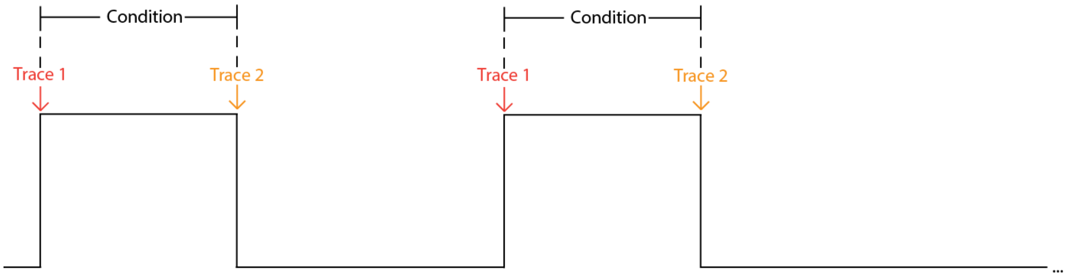
Explaining the Example: Timer 1 starts counting as soon as the program starts. When timer 1 expires, Output_value_2 is coiled on. When Input_value_1 is true, timer 1 is reset to zero and Output_value_2 goes false. When Input_value_1 is false, timer 1 starts counting again.

8.2.6 Trace

The user will use the Trace function if they want to time stamp exactly when an event occurred. Trace can be used to measure a programs run-time behavior, how long each state takes and even which states were visited in which order.

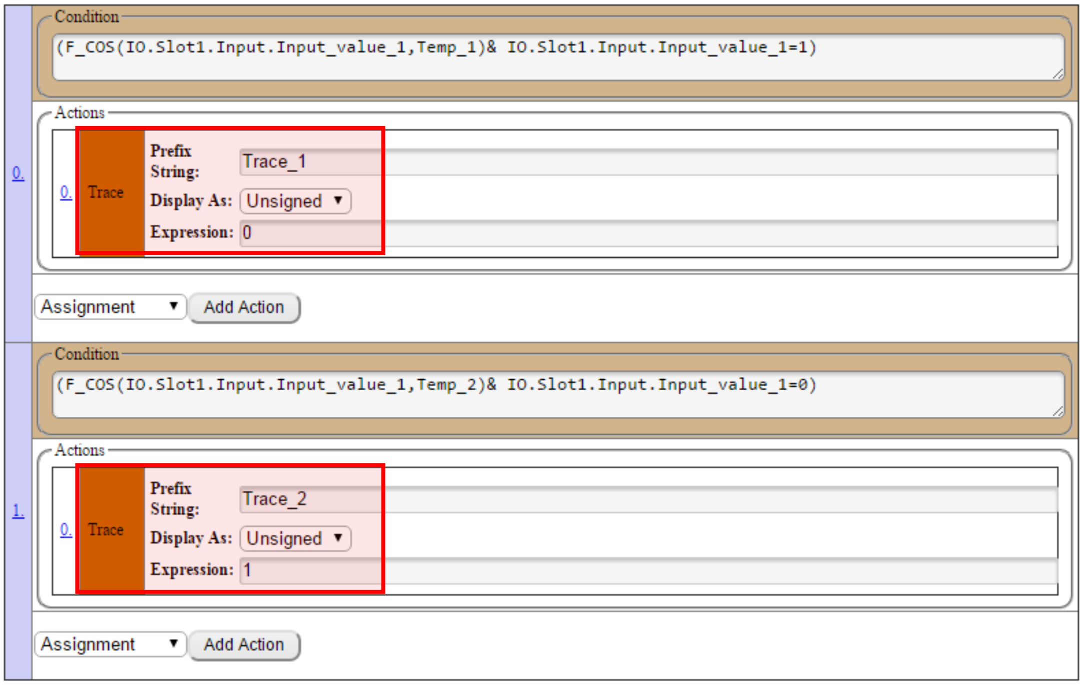
Example of Trace:

The user wants to use Trace to measure how long the condition is true.



NOTE

The below example uses the Change of State operation (F_COS) in the Condition block. The Change of State Operation is discussed in section 9.4.7 Change of State.



Condition

```
(F_COS(IO.Slot1.Input.Input_value_1,Temp_1)& IO.Slot1.Input.Input_value_1=1)
```

Actions

0. 0. Trace

Prefix String: Trace_1

Display As: Unsigned

Expression: 0

Condition

```
(F_COS(IO.Slot1.Input.Input_value_1,Temp_2)& IO.Slot1.Input.Input_value_1=0)
```

Actions

1. 0. Trace

Prefix String: Trace_2

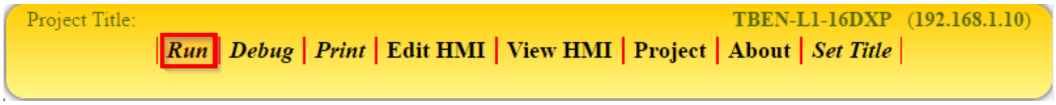
Display As: Unsigned

Expression: 1

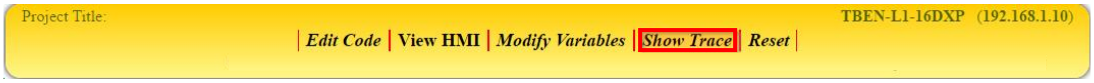
Explaining the Example: When Input_value_1 is true, Trace_1 time stamps that event. When Input_value_1 goes false, Trace_2 time stamps that event. The Prefix String is a name that makes sense to the user. The Expression can be any value or even another variable name that makes sense to the user.

The Trace example is continued on the next page.

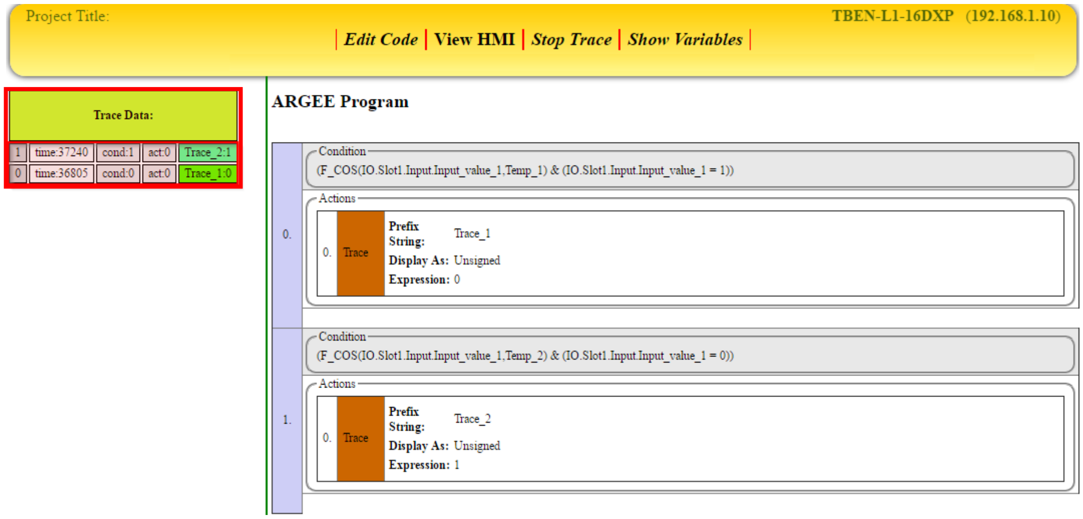
Trace Example (Continued): Once the user has written the code the user will click *Run*.



To view the Trace, the user will click *Show Trace*.



The user will trigger their Condition true then false to show a transition in the Trace data.



Trace Data:

1	time:37240	cond:1	act:0	Trace_2:1
0	time:36805	cond:0	act:0	Trace_1:0

ARGEE Program

0. Condition: (F_COS(IO.Slot1.Input.Input_value_1,Temp_1) & (IO.Slot1.Input.Input_value_1 = 1))

Actions: 0. Trace — Prefix String: Trace_1; Display As: Unsigned; Expression: 0

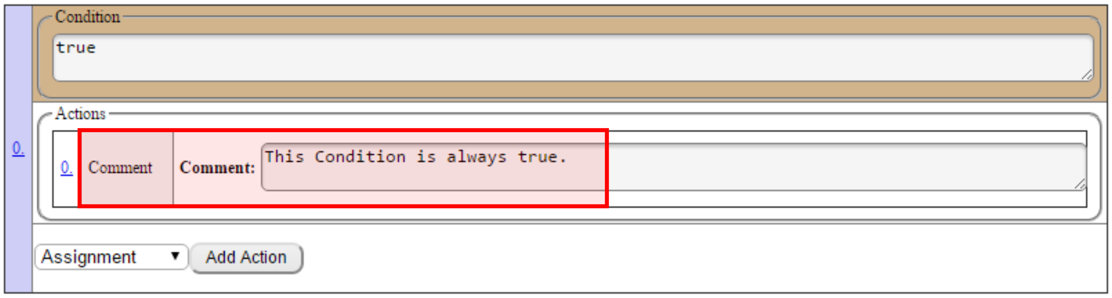
1. Condition: (F_COS(IO.Slot1.Input.Input_value_1,Temp_2) & (IO.Slot1.Input.Input_value_1 = 0))

Actions: 0. Trace — Prefix String: Trace_2; Display As: Unsigned; Expression: 1

To calculate how long the users Condition is true, the user must subtract the two time stamps from one another: 37240 - 36805 = 435ms.

8.2.7 Comment

The user can use a Comment to explain the Condition and Action statements.



Condition: true

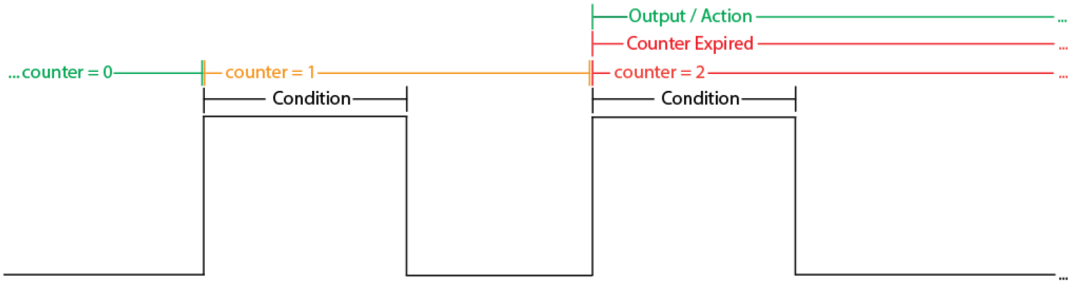
Actions: 0. Comment — Comment: This Condition is always true.

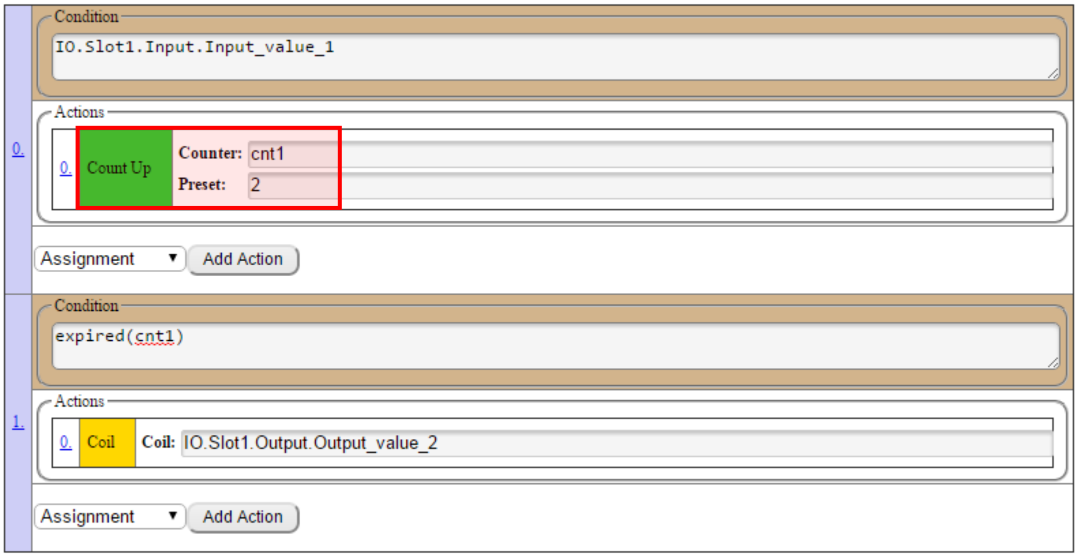
8.2.8 Count Up

The user will use *Count Up* if they want to count the number of times their condition is true. The user will normally tie an additional Action or Output to the counter expired Condition.

Example of Count Up:

The user wants to do an Action after the same Condition has occurred two times.



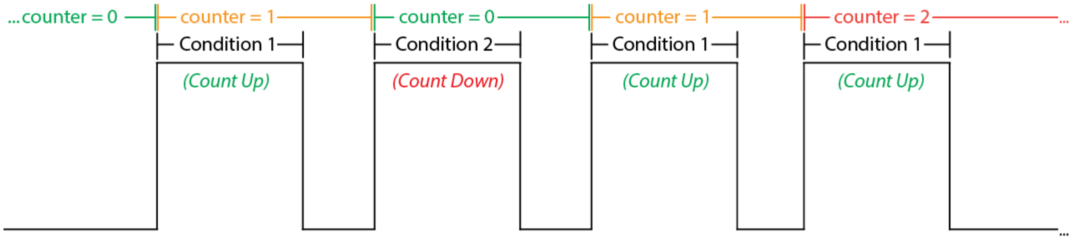

Condition
IO.Slot1.Input.Input_value_1
Actions
0. Count Up Counter: cnt1 Preset: 2

Condition
expired(cnt1)
Actions
0. Coil Coil: IO.Slot1.Output.Output_value_2

Explaining the Example: Each time Input_value_1 is true, counter 1 counts up one time. Counter 1 expires after two counts. When counter 1 expires, Output_value_2 is coiled on.

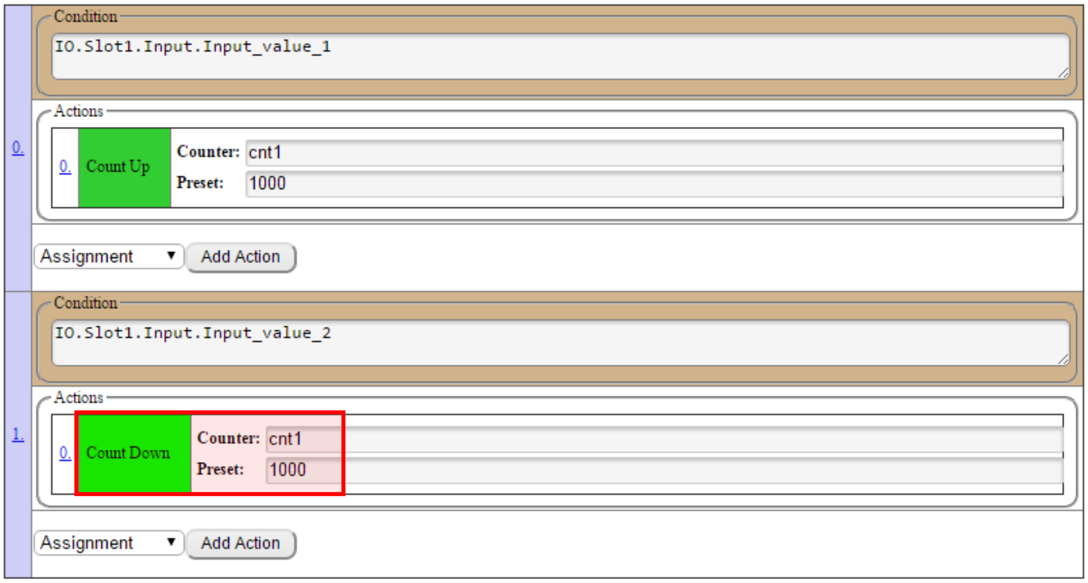
8.2.9 Count Down

The user will use *Count Down* if they want to count down when a condition is true. Count Down is normally used to counter the *Count Up* Action.



Example of Count Down:

The user wants to keep track of the number of guests in the store. When a guest walks in the store the counter goes up, but when a guest walks out of the store the counter goes down.


0.

Condition

```
IO.Slot1.Input.Input_value_1
```

Actions

0. Count Up — Counter: cnt1 — Preset: 1000

1.

Condition

```
IO.Slot1.Input.Input_value_2
```

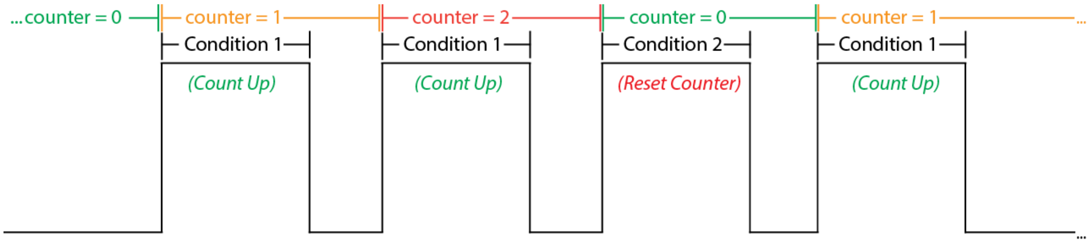
Actions

0. Count Down — Counter: cnt1 — Preset: 1000

Explaining the Example: Each time Input_value_1 is true (or a guest walks in the store), counter 1 counts up one time. Each time Input_value_2 is true (or a guest walks out of the store), counter 1 counts down one time.

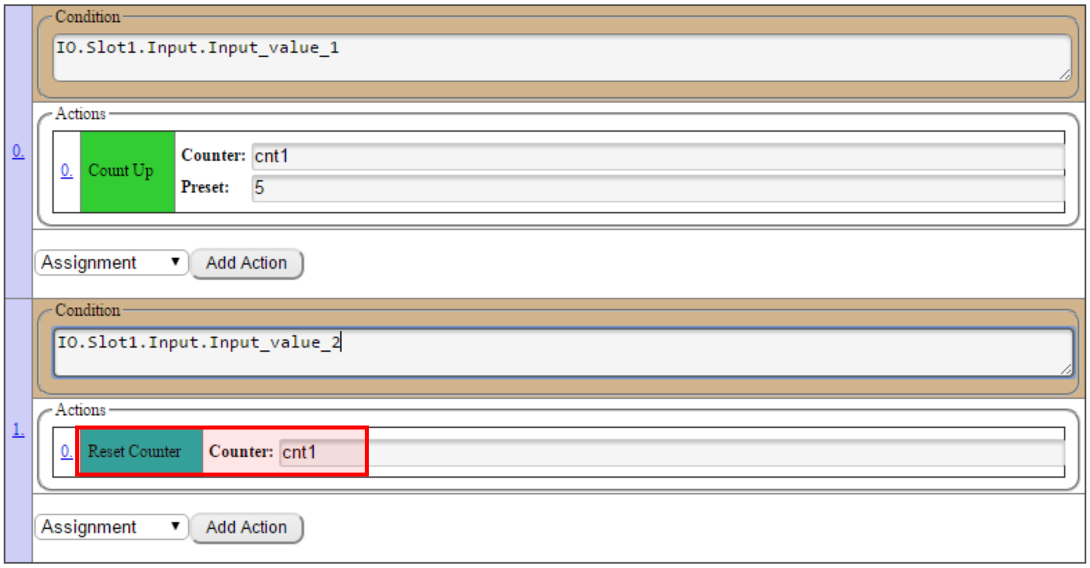
8.2.10 Reset Counter

The user will use *Reset Counter* if they want to reset a counter to zero.



Example of Reset Counter:

The user wants the ability to reset the counter at any time.



```
IO.Slot1.Input.Input_value_1
```

Counter: cnt1
Preset: 5

```
IO.Slot1.Input.Input_value_2
```

Counter: cnt1

Explaining the Example: Each time Input_value_1 is true, counter 1 counts up one time. Each time Input_value_2 is true, counter 1 resets to zero.

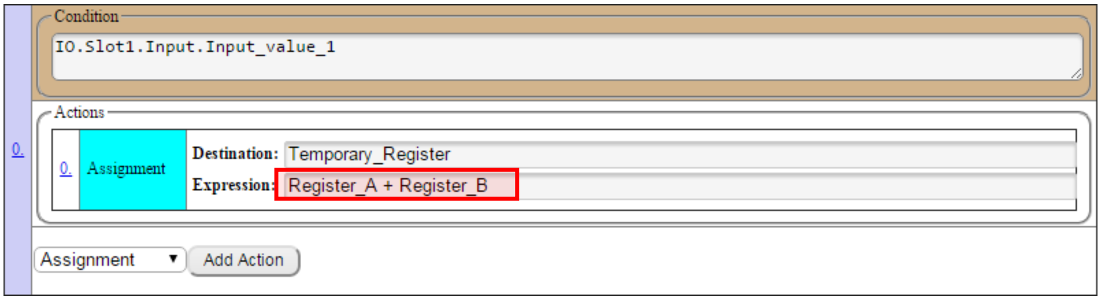
9 Operations

9.1 Math

9.1.1 Addition

The user will use the Add Operation (+) to add one value to another value.

Example of Add Operation:



Condition

```
IO.Slot1.Input.Input_value_1
```

Actions

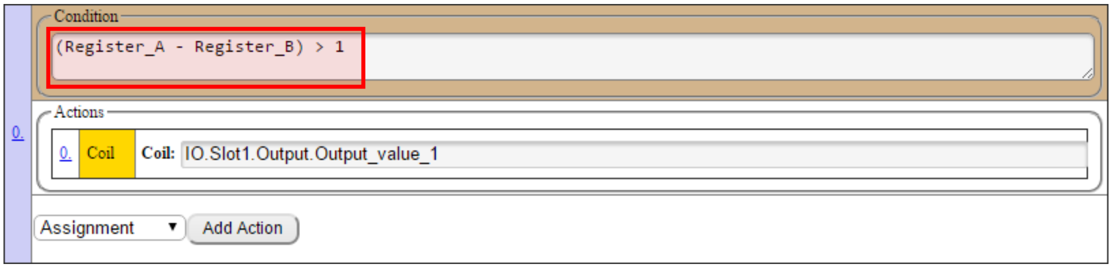
0. Assignment — Destination: Temporary_Register — Expression: Register_A + Register_B

Explaining the Example: When Input_value_1 is true, the value in Register_A will be added to the value in Register_B. The result is placed in Temporary_Register.

9.1.2 Subtraction

The user will use the Subtraction Operation (-) to subtract one value from another value.

Example of Subtraction Operation:



Condition

```
(Register_A - Register_B) > 1
```

Actions

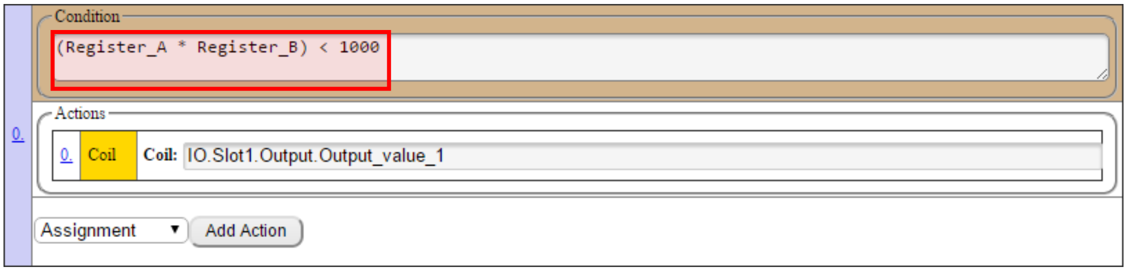
0. Coil — Coil: IO.Slot1.Output.Output_value_1

Explaining the Example: The user is subtracting the value in Register_A from the value in Register_B. When Register A minus Register B is greater than 1, the user Coils on Output_value_1.

9.1.3 Multiplication

The user will use the Multiplication Operation (*) to multiply one value with another value.

Example of Multiplication Operation:



Condition

```
(Register_A * Register_B) < 1000
```

Actions

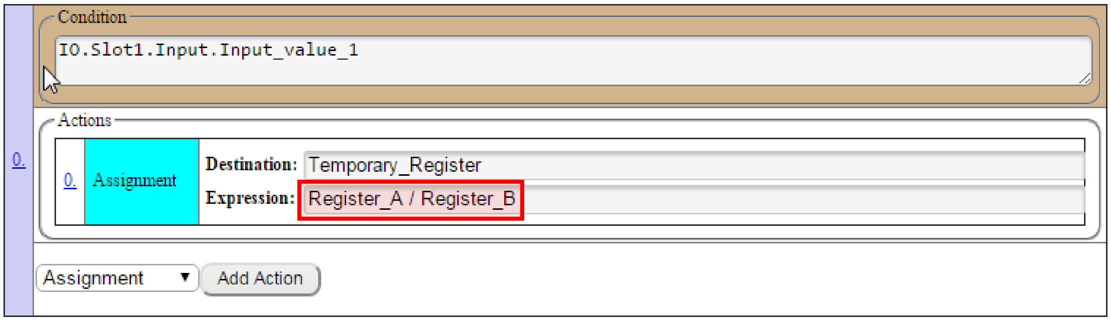
0. Coil — **Coil:** IO.Slot1.Output.Output_value_1

Explaining the Example: The user is multiplying the value in Register_A with the value in Register_B. If Register A times Register B is less than 1000, the user Coils on Output_value_1.

9.1.4 Division

The user will use the Division Operation (/) to divide one value into another value.

Example of Division Operation:



Condition

```
IO.Slot1.Input.Input_value_1
```

Actions

0. Assignment — **Destination:** Temporary_Register
Expression: Register_A / Register_B

Explaining the Example: When Input_value_1 is true, the value in Register_A will be divided by the value in Register_B. The result is placed in Temporary_Register.

ℹ NOTE

ARGEE does not currently support floating point math (or fractions). If the result has a fraction in it, ARGEE will drop the fraction and just display the whole number.

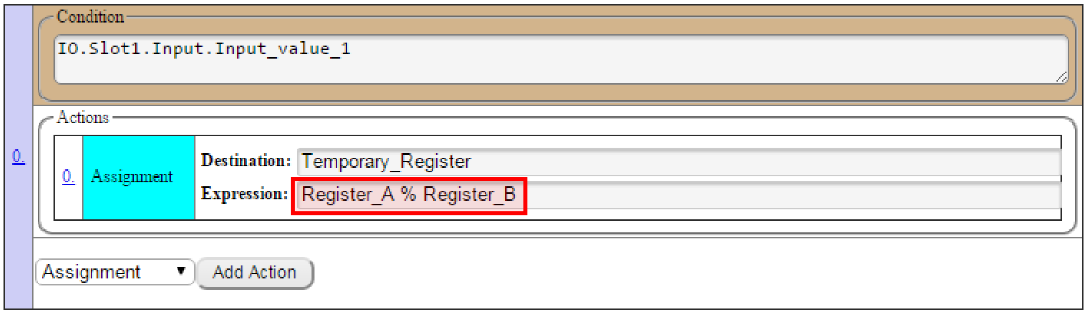
For example:

36 / 6 = 6	**→**	**ARGEE displays "6"**
34 / 6 = $5\frac{4}{6}$	**→**	**ARGEE displays "5"**
6 / 36 = $\frac{1}{6}$	**→**	**ARGEE displays "0"**

9.1.5 Modulo

The user will use the Modulo Operation (%) if they want to capture the “remainder” after a Division Operation has occurred.

Example of Modulo Operation:



IO.Slot1.Input.Input_value_1

Destination: Temporary_Register
Expression: Register_A % Register_B

Explaining the Example: When Input_value_1 is true, the value in Register_A will be divided by the value in Register_B. The “remainder” from the Division Operation is placed in Temporary_Register.

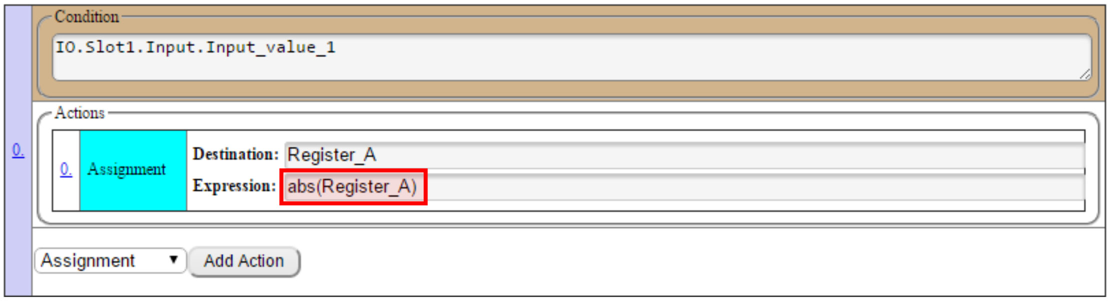
For example:

36 / 6 = “6” with a remainder of “0”	**→**	**ARGEE displays “0”**
34 / 6 = “5” with a remainder of “4”	**→**	**ARGEE displays “4”**
6 / 36 = “0” with a remainder of “6”	**→**	**ARGEE displays “6”**

9.1.6 Absolute Value

The user will use the Absolute Value Operation (abs) to capture the magnitude of a real number without regard to its sign.

Example of Absolute Value Operation:



IO.Slot1.Input.Input_value_1

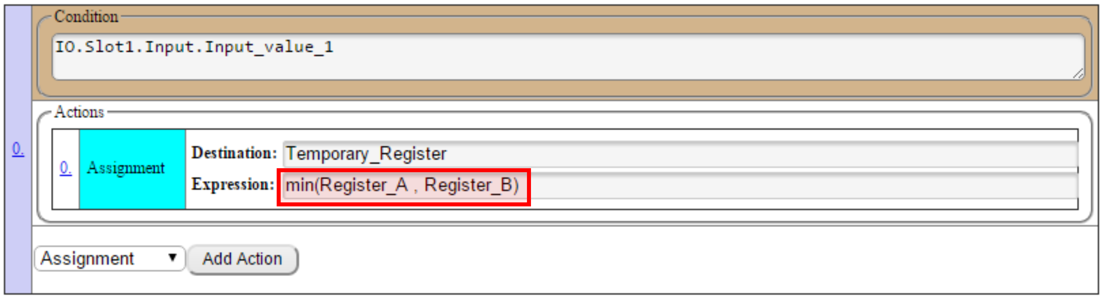
Destination: Register_A
Expression: abs(Register_A)

Explaining the Example: When Input_value_1 is true, ARGEE will take the value in Register_A, find its Absolute Value, and place that value back in Register_A.

9.1.7 Minimum Value

The user will use the Minimum Value Operation (min) to compare multiple registers and place the smallest value in to the Destination Register. The user can also use the Minimum Value Operation (min) to compare multiple registers and use the smallest value in a Math Operation.

Example of Minimum Value Operation:



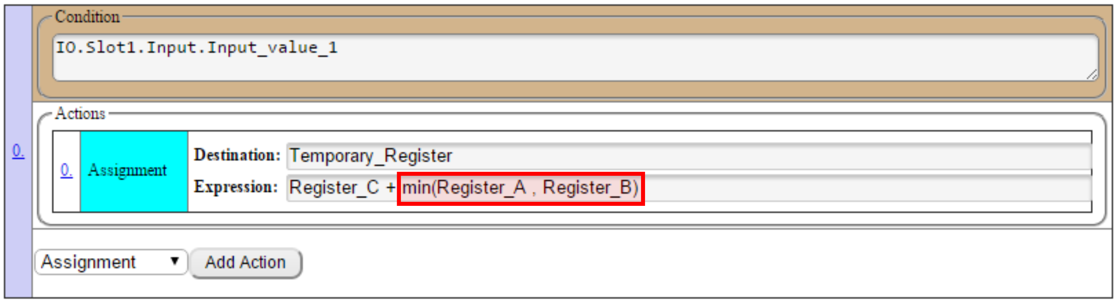
IO.Slot1.Input.Input_value_1

Destination: Temporary_Register

Expression: min(Register_A , Register_B)

Explaining the Example: When Input_value_1 is true, ARGEE will take the smallest value between Register_A and Register_B and place that value into Temporary_Register

OR



IO.Slot1.Input.Input_value_1

Destination: Temporary_Register

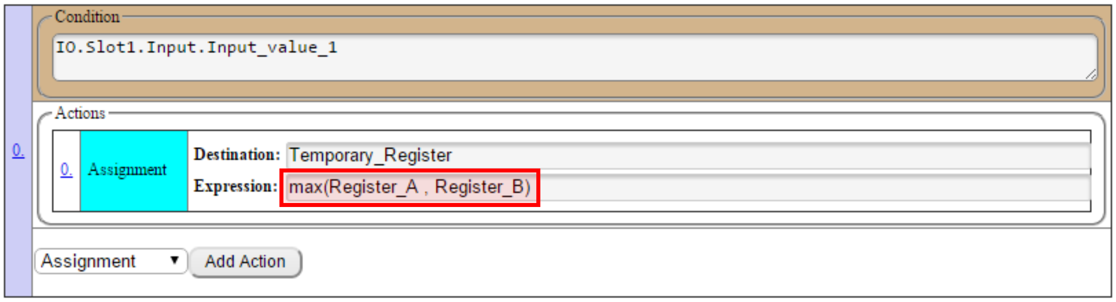
Expression: Register_C + min(Register_A , Register_B)

Explaining the Example: When Input_value_1 is true, ARGEE will take the smallest value between Register_A and Register_B and place that value into the Math Operation. The result will be stored in Temporary_Register.

9.1.8 Maximum Value

The user will use the Maximum Value Operation (max) to compare multiple registers and place the largest value in to the Destination Register. The user can also use the Maximum Value Operation (max) to compare multiple registers and use the largest value in a Math Operation.

Example of Maximum Value Operation:

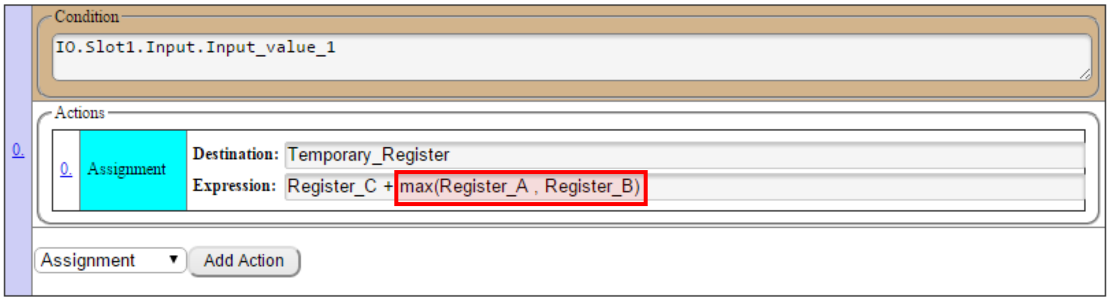


Condition: IO.Slot1.Input.Input_value_1

Actions: 0. Assignment — Destination: Temporary_Register — Expression: max(Register_A , Register_B)

Explaining the Example: When Input_value_1 is true, ARGEE will take the largest value between Register_A and Register_B and place that value into Temporary_Register.

OR



Condition: IO.Slot1.Input.Input_value_1

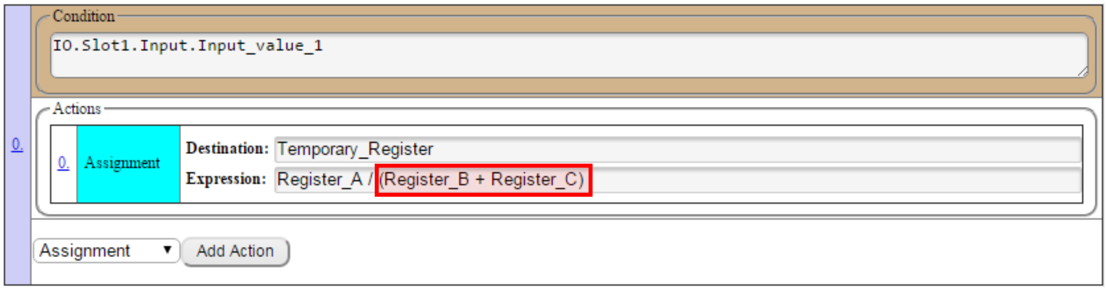
Actions: 0. Assignment — Destination: Temporary_Register — Expression: Register_C + max(Register_A , Register_B)

Explaining the Example: When Input_value_1 is true, ARGEE will take the largest value between Register_A and Register_B and place that value into the Math Operation. The result will be stored in Temporary_Register.

9.2 Brackets

The user will use Brackets ( ) to show the order of operations while performing Math.

Example of Brackets:



Condition: IO.Slot1.Input.Input_value_1

Destination: Temporary_Register

Expression: Register_A / (Register_B + Register_C)

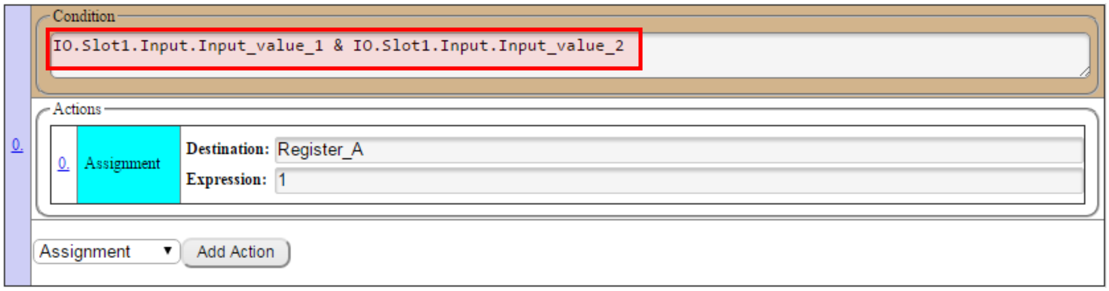
Explaining the Example: When Input_value_1 is true, ARGEE will examine the "(Register_B + Register_C)" first and then divide the answer into the value in Register_A. The result will be stored in Temporary_Register.

9.3 Boolean Logic

9.3.1 Boolean AND

The user will use the Boolean AND Operation (&) if the user wants to combine several *Conditions* together before allowing a specific Action to occur.

Example of Boolean AND:



Condition: IO.Slot1.Input.Input_value_1 & IO.Slot1.Input.Input_value_2

Destination: Register_A

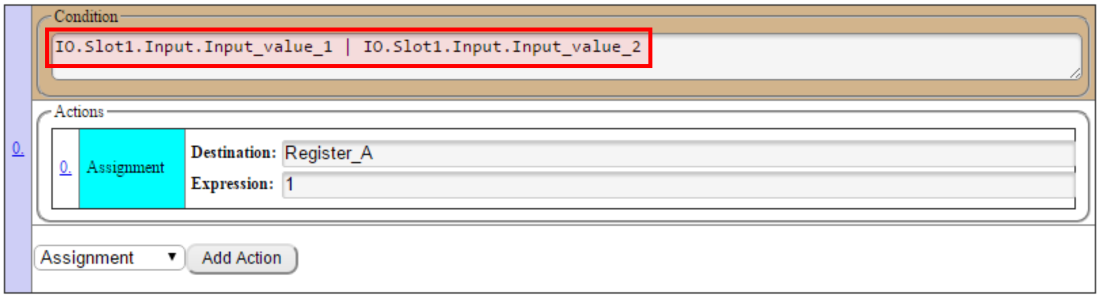
Expression: 1

Explaining the Example: When both Input_value_1 AND input_value_2 are true, load the value "1" into Register_A.

9.3.2 Boolean OR

The user will use the Boolean OR Operation (|) if the user wants one of several *Conditions* to cause an Action to occur.

Example of Boolean OR:



Condition: `IO.Slot1.Input.Input_value_1 | IO.Slot1.Input.Input_value_2`

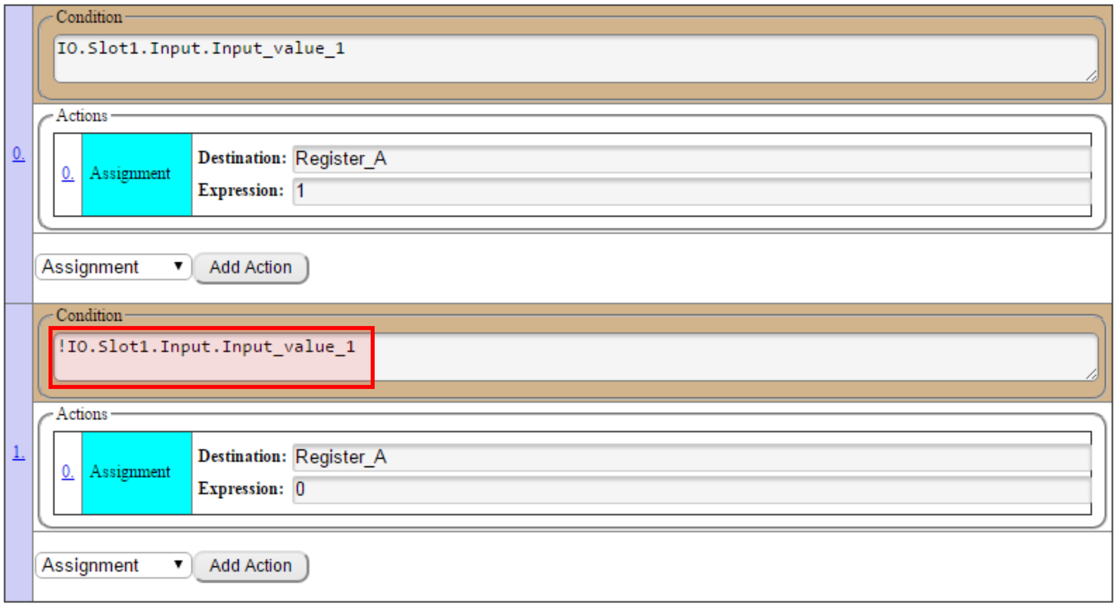
Actions: 0. Assignment — Destination: Register_A; Expression: 1

Explaining the Example: When either Input_value_1 OR input_value_2 are true, load the value “1” into Register_A.

9.3.3 Boolean NOT

The user will use the Boolean NOT Operation (!) if the user wants an Action to occur while a *Condition* is false.

Example of Boolean NOT:



0. Condition: `IO.Slot1.Input.Input_value_1`

Actions: 0. Assignment — Destination: Register_A; Expression: 1

1. Condition: `!IO.Slot1.Input.Input_value_1`

Actions: 0. Assignment — Destination: Register_A; Expression: 0

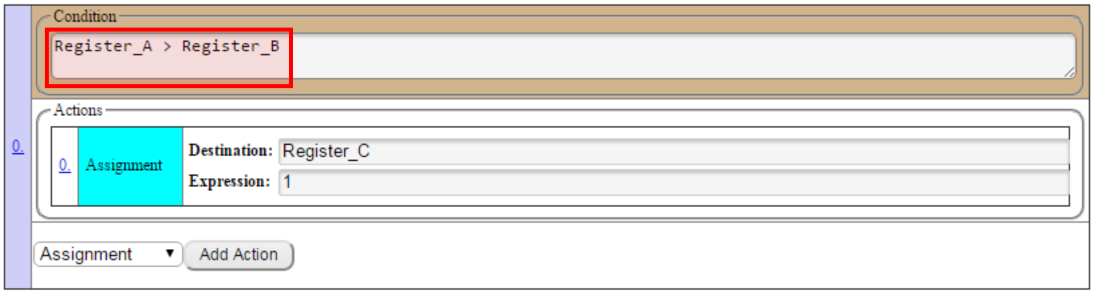
Explaining the Example: When Input_value_1 is true, load the value “1” into Register_A. When Input_value_1 is false, load the value “0” into Register_A.

9.4 Compare

9.4.1 Greater Than

The user will use the Greater Than Operation (>) if the user wants a *Condition* to occur when one register value is Greater Than another register value.

Example of Greater Than:



Condition

```
Register_A > Register_B
```

Actions

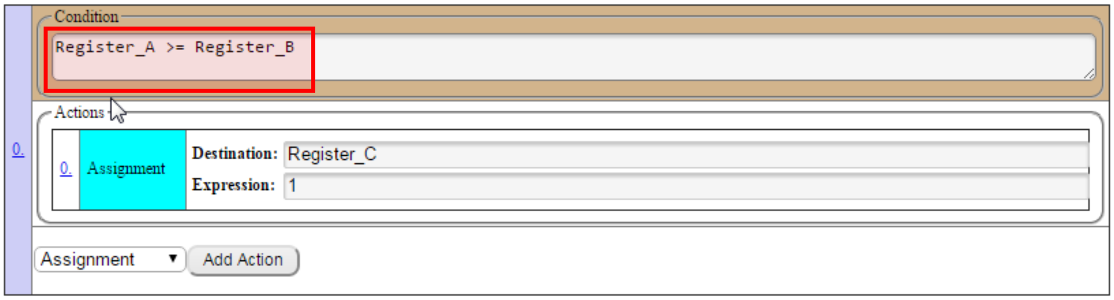
0. Assignment — Destination: Register_C; Expression: 1

Explaining the Example: When the value in Register_A is Greater Than the value in Register_B, the value “1” will be loaded into Register_C.

9.4.2 Greater Than or Equal to

The user will use the Greater Than or Equal to Operation (>=) if the user wants a *Condition* to occur when one register value is Greater Than or Equal to another register value.

Example of Greater Than:



Condition

```
Register_A >= Register_B
```

Actions

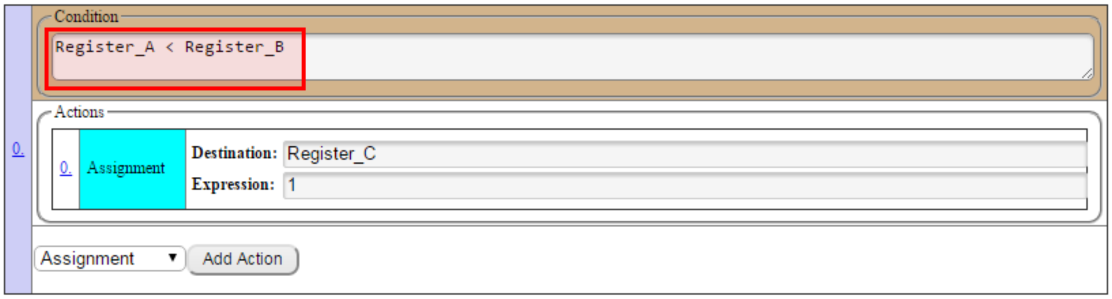
0. Assignment — Destination: Register_C; Expression: 1

Explaining the Example: When the value in Register_A is Greater Than or Equal to the value in Register_B, the value “1” will be loaded into Register_C.

9.4.3 Less Than

The user will use the Less Than Operation (<) if the user wants a *Condition* to occur when one register value is Less Than another register value.

Example of Greater Than:



Condition
```
Register_A < Register_B
```
Actions

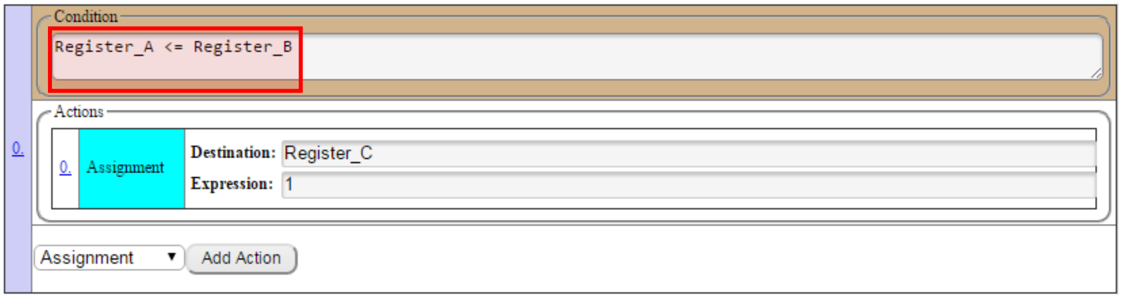
0. Assignment — Destination: Register_C — Expression: 1

Explaining the Example: When the value in Register_A is Less Than the value in Register_B, the value “1” will be loaded into Register_C.

9.4.4 Less Than or Equal to

The user will use the Less Than or Equal to Operation (<=) if the user wants a *Condition* to occur when one register value is Less Than or Equal to another register value.

Example of Greater Than:



Condition
```
Register_A <= Register_B
```
Actions

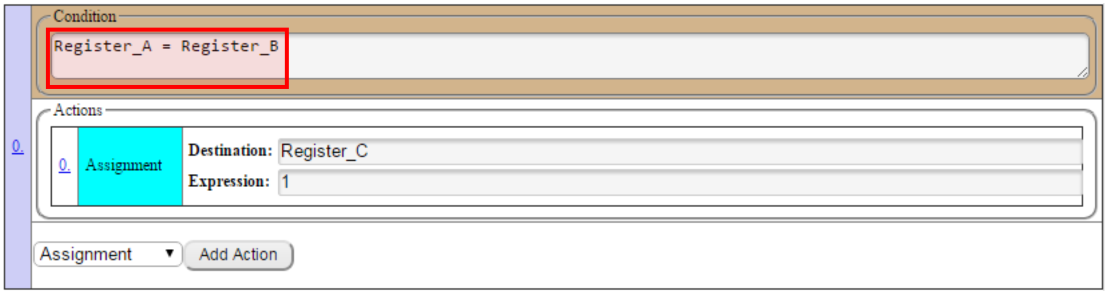
0. Assignment — Destination: Register_C — Expression: 1

Explaining the Example: When the value in Register_A is Less Than or Equal to the value in Register_B, the value “1” will be loaded into Register_C.

9.4.5 Equal

The user will use the Equal Operation (=) if the user wants a *Condition* to occur when one register value is Equal to another register value.

Example of Equal:



Condition

```
Register_A = Register_B
```

Actions

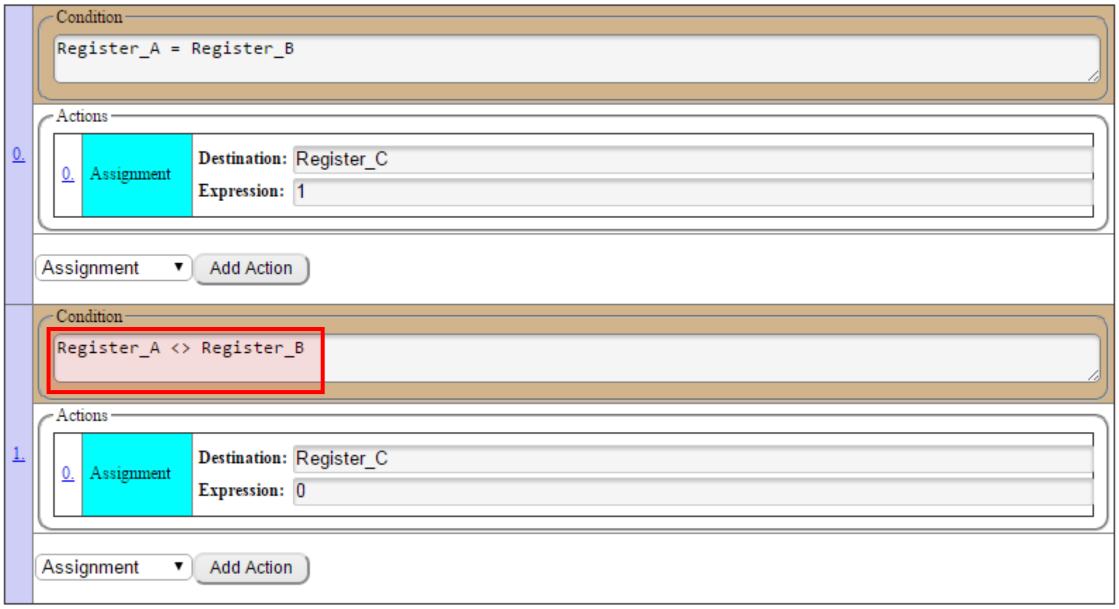
0. Assignment — Destination: Register_C — Expression: 1

Explaining the Example: When the value in Register_A is Equal to the value in Register_B, the value “1” will be loaded into Register_C.

9.4.6 Not Equal

The user will use the Not Equal Operation (<>) if the user wants a *Condition* to occur when one register value is Not Equal to another register value.

Example of Not Equal:



0.

Condition

```
Register_A = Register_B
```

Actions

0. Assignment — Destination: Register_C — Expression: 1

1.

Condition

```
Register_A <> Register_B
```

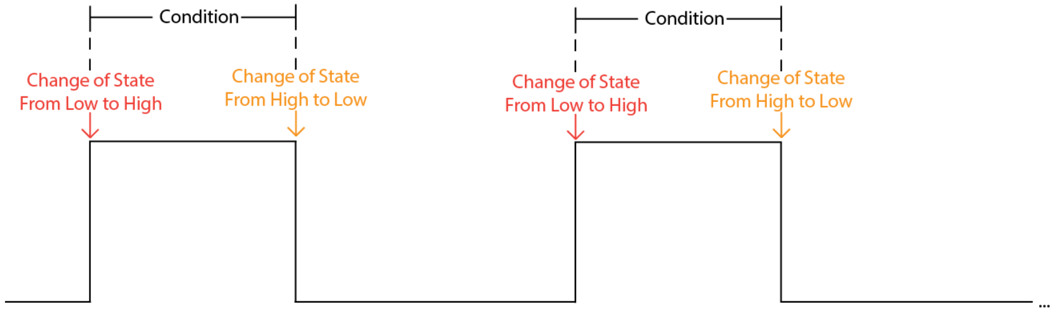
Actions

0. Assignment — Destination: Register_C — Expression: 0

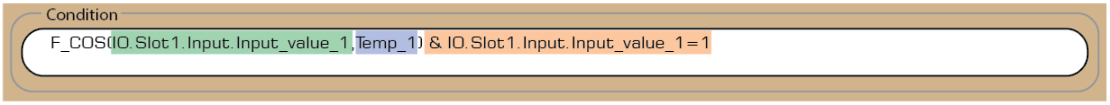
Explaining the Example: When the value in Register_A is Equal to the value in Register_B, the value “1” will be loaded into Register_C. When the value in Register_A is Not Equal to the value in Register_B, the value “0” will be loaded into Register_C.

9.4.7 Change of State

The user will use the Change of State Operation (F_COS) if the user wants an Action only to occur when a *Condition* changes state. A *Condition* can change state from either “low to high” or “high to low”.



Change of State Command Structure:



Condition

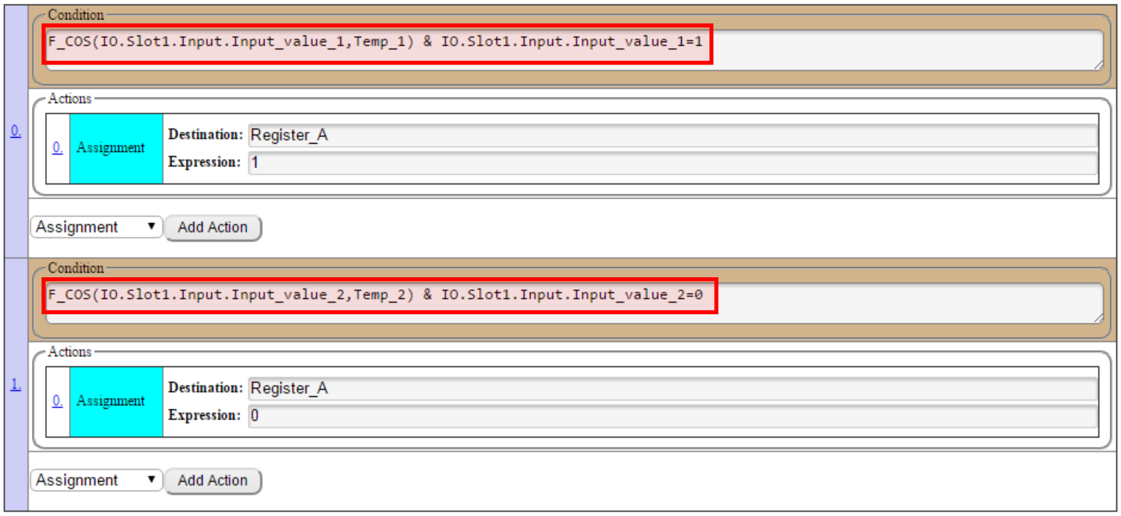
F_COS(IO.Slot1.Input.Input_value_1,Temp_1) & IO.Slot1.Input.Input_value_1=1

🟩 = The *Condition* that is being monitored for a *Change of State.*

🟦 = The register that stores the monitored *Condition’s* current state.

🟧 = The part of the *Condition* that tells *ARGEE* to monitor the “low to high” *Change of State or* the “high to low” *Change of State*.

Example of Change of State:



0.

Condition

```
F_COS(IO.Slot1.Input.Input_value_1,Temp_1) & IO.Slot1.Input.Input_value_1=1
```

Actions

0. Assignment — Destination: Register_A; Expression: 1

1.

Condition

```
F_COS(IO.Slot1.Input.Input_value_2,Temp_2) & IO.Slot1.Input.Input_value_2=0
```

Actions

0. Assignment — Destination: Register_A; Expression: 0

Explaining the Example: When Input_value_1 does a Change of State from low (zero) to high (one), the value “1” is loaded into Register_A. When Input_value_2 does a Change of State from high (one) to low (zero), the value “0” is loaded into Register_A.

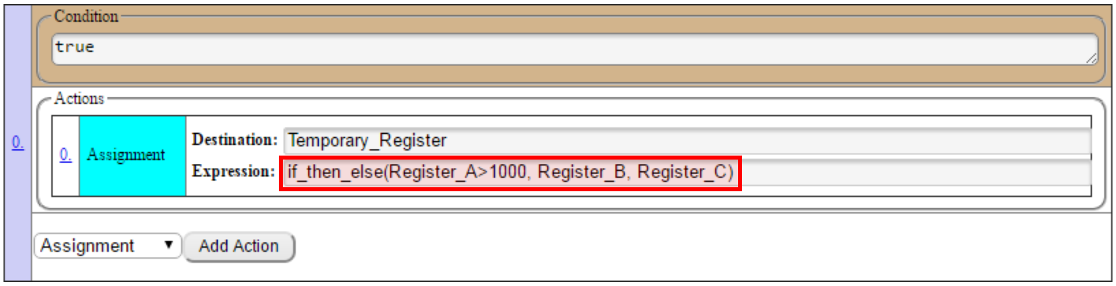
NOTE

Each monitored Condition requires its own Current State register. Notice in the example, Temp_1 was used to monitor the Change of State of Input_value_1 and Temp_2 was used to monitor the Change of State of Input_value_2.

9.4.8 If_Then_Else

The user will use the If_Then_Else operation if they want an expression to be set *only* if a particular test evaluates as true. If it evaluates as false, a secondary expression is chosen.

Example of if_then_else:



Condition: true

Destination: Temporary_Register

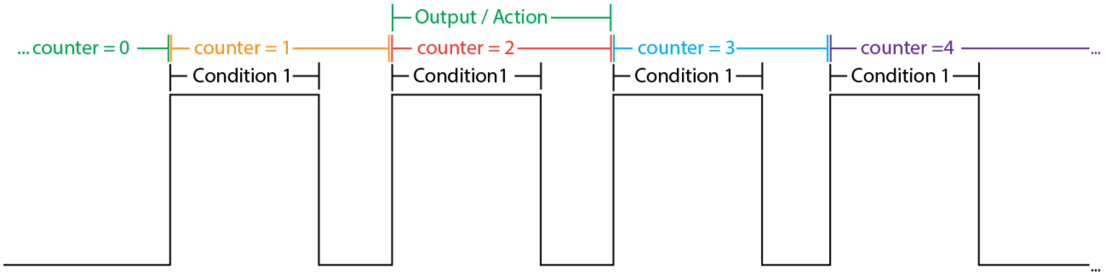
Expression: if_then_else(Register_A>1000, Register_B, Register_C)

Explaining the Example: If the value in Register_A is above 1000, then the value in Register_B is loaded into the Temporary_Register. If the value in Register_A is below 1000, then the value in Register_C is loaded into the Temporary_Register.

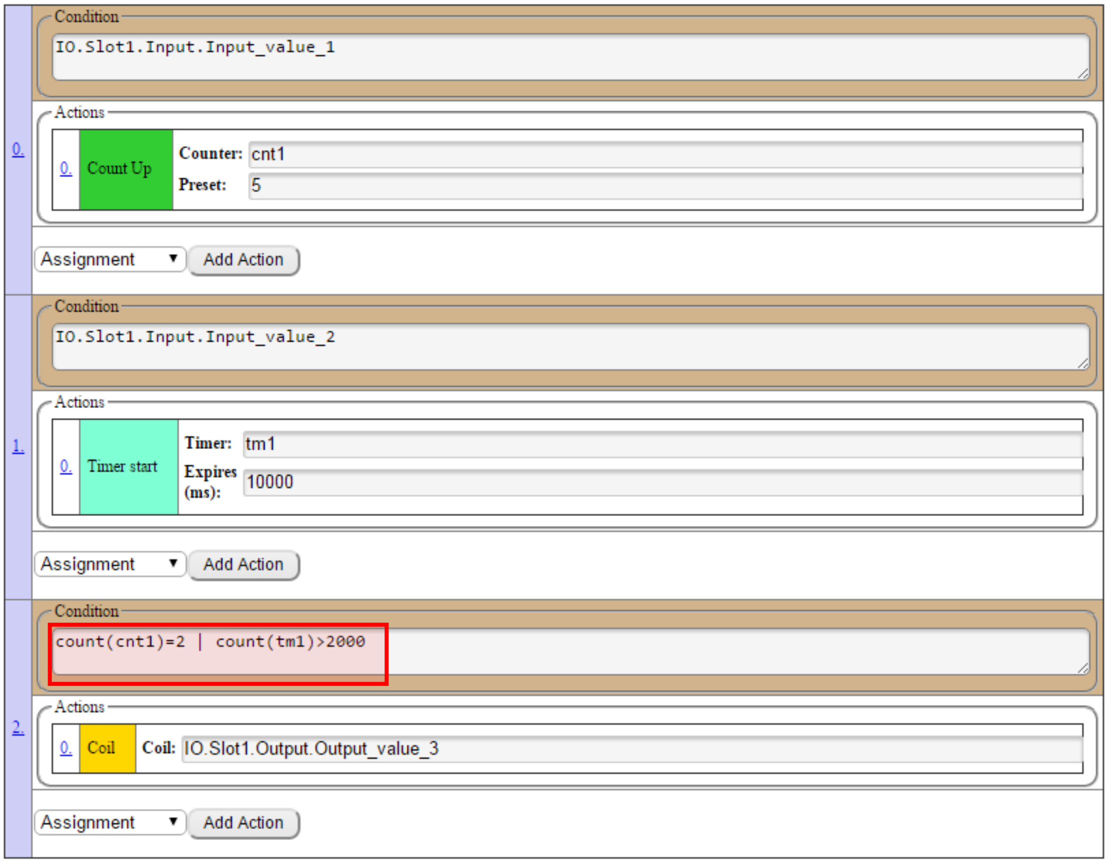
9.5 Timer/Counter

9.5.1 Count

The user will use the Count Operation (count) if the user wants to perform an Action when a counter or timer is at a specific value but has not yet expired.



Example of Count:



0.
Condition
```
IO.Slot1.Input.Input_value_1
```
Actions

0.	Count Up	Counter:	cnt1
		Preset:	5

1.
Condition
```
IO.Slot1.Input.Input_value_2
```
Actions

0.	Timer start	Timer:	tm1
		Expires (ms):	10000

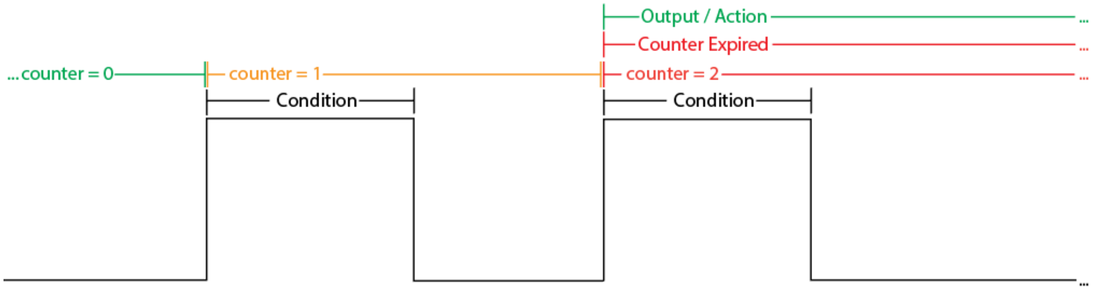
2.
Condition
```
count(cnt1)=2 | count(tm1)>2000
```
Actions

0.	Coil	Coil:	IO.Slot1.Output.Output_value_3

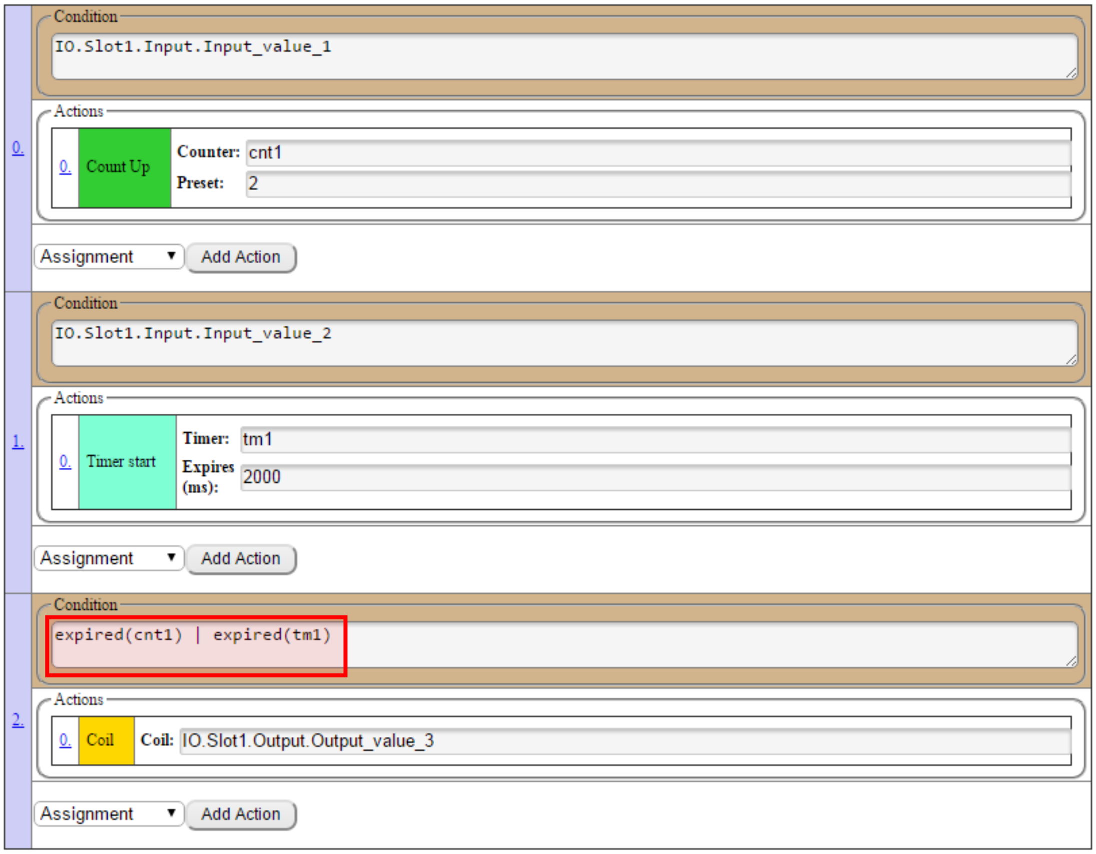
Explaining the Example: Each time Input_value_1 goes true, counter 1 counts up one time. Counter 1 expires after five counts. After input_value_2 goes true, timer 1 starts. Timer 1 will expire after ten seconds (or 10,000ms). When counter 1 counts to “2” OR timer 1 is Greater Than two seconds (or 2000ms), Output_value_3 is coiled on.

9.5.2 Expired

The user will use the Expired Operation (expired) if they want to perform an Action when a counter or timer has expired.



Example of Expired:



0.

Condition

```
IO.Slot1.Input.Input_value_1
```

Actions

0. Count Up — Counter: cnt1 — Preset: 2

Assignment ▼ Add Action

1.

Condition

```
IO.Slot1.Input.Input_value_2
```

Actions

0. Timer start — Timer: tm1 — Expires (ms): 2000

Assignment ▼ Add Action

2.

Condition

```
expired(cnt1) | expired(tm1)
```

Actions

0. Coil — Coil: IO.Slot1.Output.Output_value_3

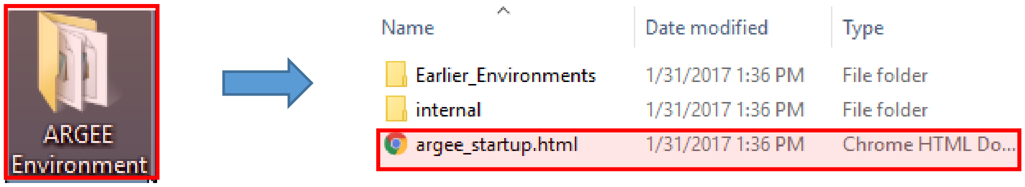
Assignment ▼ Add Action

Explaining the Example: Each time Input_value_1 goes true, counter 1 counts up one time. Counter 1 expires after two counts. After input_value_2 it goes true, timer 1 starts. Timer 1 will expire after two seconds (or 2,000ms). When counter 1 OR timer 1 expires, Output_value_3 is coiled on.

10 ARGEE Simulation Mode

10.1 Opening the Environment

➢ Open the ARGEE Environment and double click on argee_startup.html.



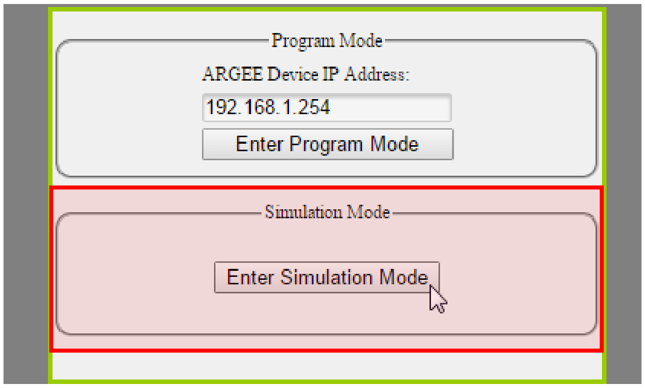
Name	Date modified	Type
Earlier_Environments	1/31/2017 1:36 PM	File folder
internal	1/31/2017 1:36 PM	File folder
argee_startup.html	1/31/2017 1:36 PM	Chrome HTML Do...

NOTE

ARGEE only opens up in HTML 5 compliant web browsers such as Google Chrome or Firefox.

10.2 Logging into Simulation Mode

➢ Click Enter Simulation Mode.

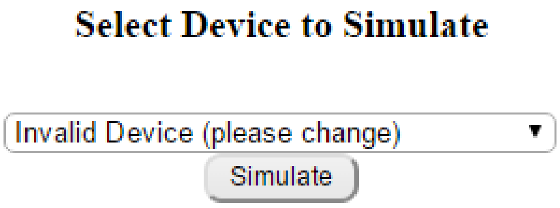


Program Mode

ARGEE Device IP Address: 192.168.1.254

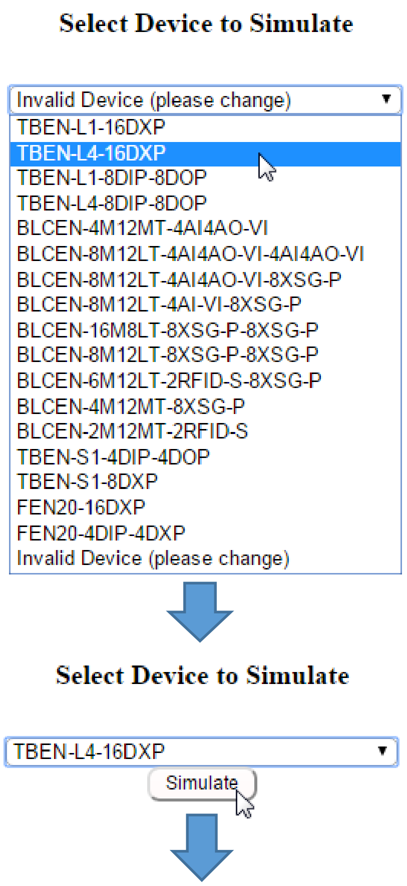
10.3 Select Device to Simulate

If the user has never used Simulation Mode before, the first thing they will have to do is select a device to simulate from the drop down arrow.



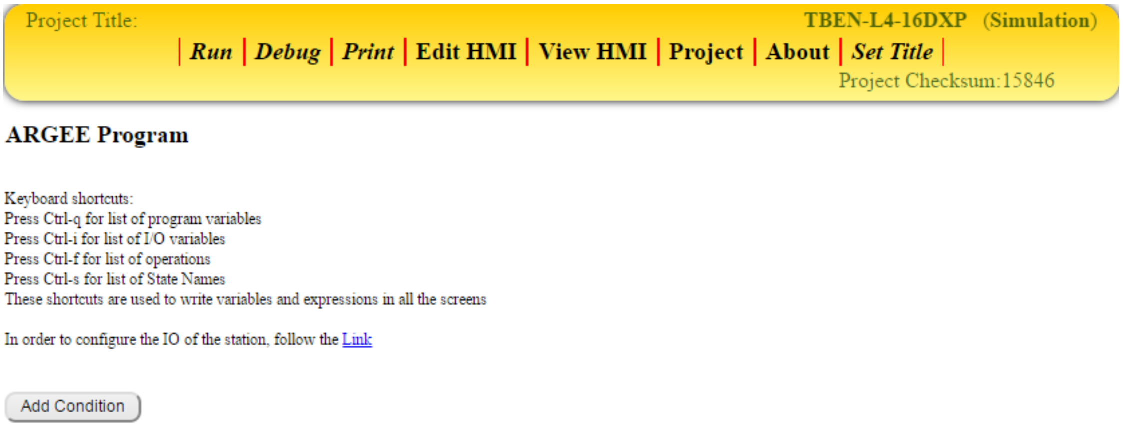
Select Device to Simulate

Invalid Device (please change) ▼



Select Device to Simulate
Invalid Device (please change)
TBEN-L1-16DXP
TBEN-L4-16DXP
TBEN-L1-8DIP-8DOP
TBEN-L4-8DIP-8DOP
BLCEN-4M12MT-4AI4AO-VI
BLCEN-8M12LT-4AI4AO-VI-4AI4AO-VI
BLCEN-8M12LT-4AI4AO-VI-8XSG-P
BLCEN-8M12LT-4AI-VI-8XSG-P
BLCEN-16M8LT-8XSG-P-8XSG-P
BLCEN-8M12LT-8XSG-P-8XSG-P
BLCEN-6M12LT-2RFID-S-8XSG-P
BLCEN-4M12MT-8XSG-P
BLCEN-2M12MT-2RFID-S
TBEN-S1-4DIP-4DOP
TBEN-S1-8DXP
FEN20-16DXP
FEN20-4DIP-4DXP
Invalid Device (please change)
Select Device to Simulate
TBEN-L4-16DXP
Simulate

10.4 Welcome to ARGEE Simulation Mode



Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

11 ARGEE Security

11.1 General Security

Security is a concern to some users. ARGEE provides several security features, the first of which is General Security. General Security is the term used to explain a block's behavior with ARGEE programming versus a block's behavior without ARGEE programming.

11.1.1 Visual Behavior

If there is an ARGEE program running on the block:

- The BUS LED will flash green three times and then stay off for 1 second.

If there is not an ARGEE program running on the block:

- The block's LED's will behave in accordance with that block's data sheet.

11.1.2 Connection Behavior

11.1.2.1 Ethernet IP Master (Allen Bradley)

If there is an ARGEE program running on the block before a PLC connection is established:

- The PLC connection point combinations 101,102 or 103,104 will not be allowed
- ARGEE will block any attempt by the PLC to upload parameters from the block
- The PLC will only be able to make connection to the block via the ARGEE connection pair 101, 110

If the PLC makes a connection to the block before an ARGEE program is loaded:

- The PLC connection point combinations 101,102 or 103,104 will be allowed
- The AGREE connection pair 101, 110 will not be allowed
- The ARGEE environment will not allow upload of new code

11.1.2.2 Modbus TCP Master (VT500 or Red Lion HMI)

If there is an ARGEE program running on the block before a Modbus connection is established:

- Regular Modbus/TCP registers will not be accessible
- Access to Regular Modbus/TCP registers results in “exception”
- Only ARGEE Modbus/TCP registers can be read/written from:
- 0x4000 - 0x407F (Registers 16384 - 16512 in decimal) Read only Input Data (ARGEE -> PLC)
- 0x4400 – 0x447F (Register 17408 - 17536 in decimal) Read/Write Output Data (PLC -> ARGEE)

If a Modbus/TCP connection is established before an ARGEE program is loaded:

- Regular Modbus/TCP registers are accessible
- Access to ARGEE specific registers results in “exception”

11.1.2.3 PROFINET Master

If there is an ARGEE program running on the block before a PROFINET connection is established:

- Standard IO PROFINET connection is not allowed. ARGEE PROFINET connection is allowed
- Access to the block can be established by installing the ARGEE GSD file to the project

If a PROFINET connection is established before an ARGEE program is loaded:

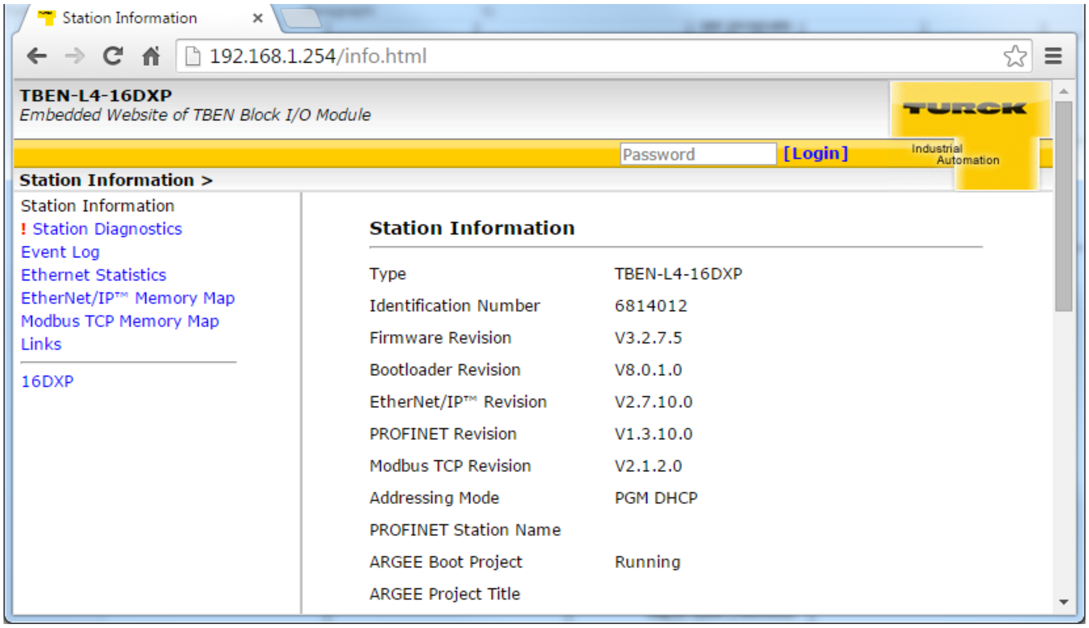
- The regular PROFINET module ID is accessible. ARGEE PROFINET connection is not allowed. If the ARGEE environment attempts to load an ARGEE code when a standard PROFINET connection is establish, the ARGEE environment will block the upload.

NOTE

PLC Connection examples can be found in chapter 14 Common Applications.

11.2 Password Protection – ARGEE Environment

All Turck block devices support a password protected webserver. To access the block's webserver, the user needs to type the blocks IP address into any web browser.



TBEN-L4-16DXP
Embedded Website of TBEN Block I/O Module

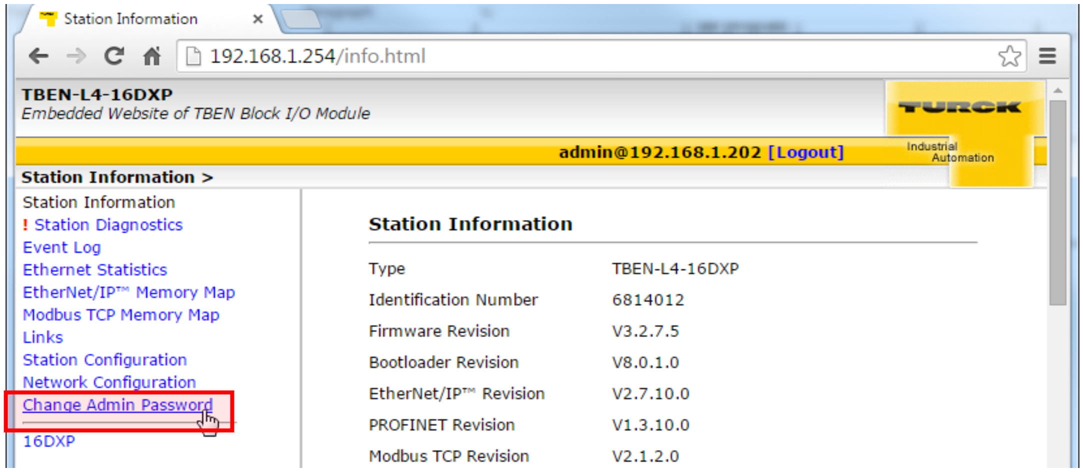
Station Information

Type	TBEN-L4-16DXP
Identification Number	6814012
Firmware Revision	V3.2.7.5
Bootloader Revision	V8.0.1.0
EtherNet/IP™ Revision	V2.7.10.0
PROFINET Revision	V1.3.10.0
Modbus TCP Revision	V2.1.2.0
Addressing Mode	PGM DHCP
PROFINET Station Name	
ARGEE Boot Project	Running
ARGEE Project Title	

NOTE

The default password to log into the blocks webserver is "password".

- To password protect the users ARGEE environment, the user must change the Admin password on their webserver.



TBEN-L4-16DXP
Embedded Website of TBEN Block I/O Module

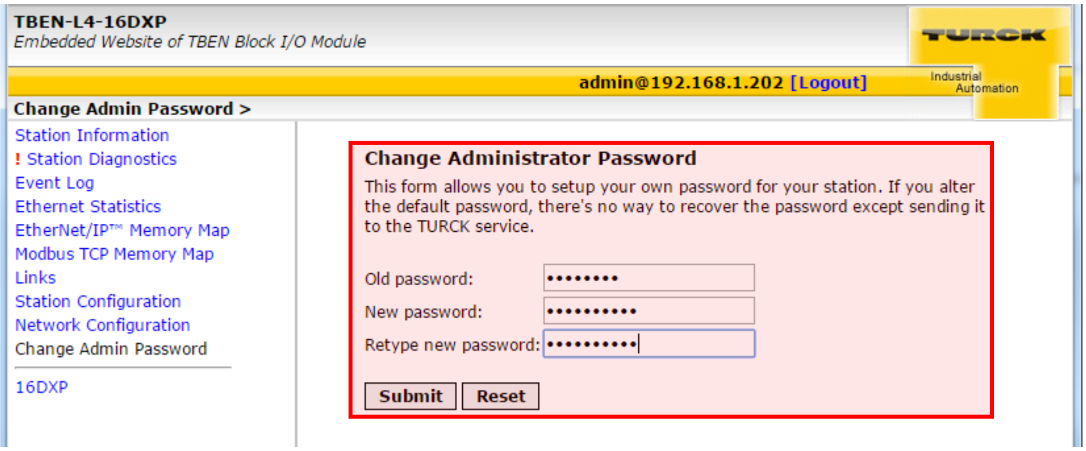
admin@192.168.1.202 [Logout]

Station Information >

Station Information

Type	TBEN-L4-16DXP
Identification Number	6814012
Firmware Revision	V3.2.7.5
Bootloader Revision	V8.0.1.0
EtherNet/IP™ Revision	V2.7.10.0
PROFINET Revision	V1.3.10.0
Modbus TCP Revision	V2.1.2.0

- To change the Admin password, select *Change Admin Password* link, follow the instructions, and click *Submit*.



TBEN-L4-16DXP
Embedded Website of TBEN Block I/O Module

admin@192.168.1.202 [Logout]

Change Admin Password >

Change Administrator Password

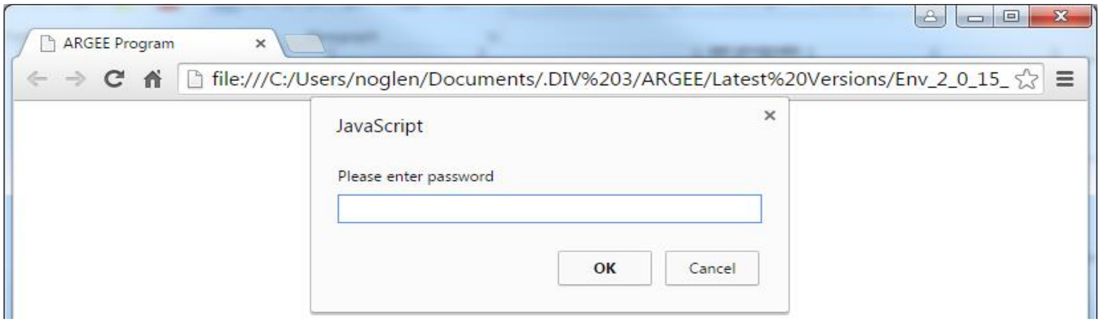
This form allows you to setup your own password for your station. If you alter the default password, there's no way to recover the password except sending it to the TURCK service.

Old password: ••••••••

New password: ••••••••••

Retype new password: ••••••••••

Now every time the user try's to log into the block, they will be prompted to input a password.



ARGEE Program
file:///C:/Users/noglen/Documents/.DIV%203/ARGEE/Latest%20Versions/Env_2_0_15_

JavaScript

Please enter password

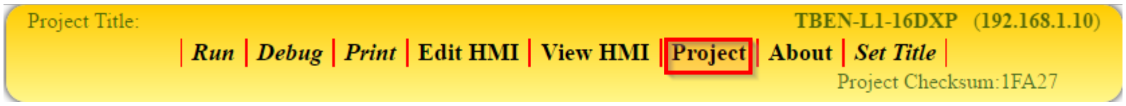
NOTE

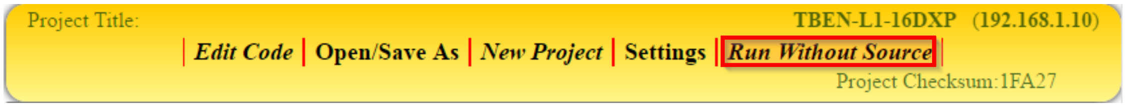
To remove this feature, the user can simply change their webserver password back to “password”.

11.3 Source Code Protection – Run Without Source

If a user wants to prevent “end users” from logging into the block and seeing or modifying code, the user will want to use the *Run Without Source* feature.

- To access the *Run Without Source* function, the user must first click on *Project* and navigate to the second ARGEE menu bar.

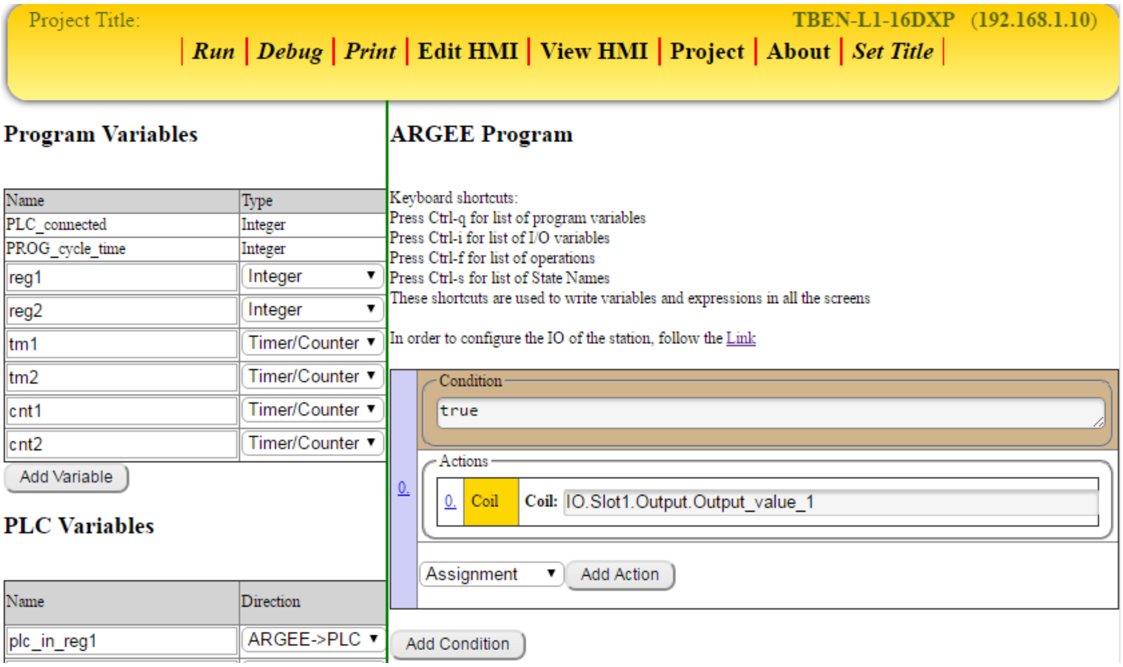




If the user clicks on *Run Without Source* and then logs out of the environment, the ARGEE program will be hidden the next time anyone logs into the block.

Example of what it looks like for the user before and after applying the Run Without Trace function:

Logging in before clicking Run Without Source:



Program Variables

Name	Type
PLC_connected	Integer
PROG_cycle_time	Integer
reg1	Integer
reg2	Integer
tm1	Timer/Counter
tm2	Timer/Counter
cnt1	Timer/Counter
cnt2	Timer/Counter

PLC Variables

Name	Direction
plc_in_reg1	ARGEE->PLC

ARGEE Program

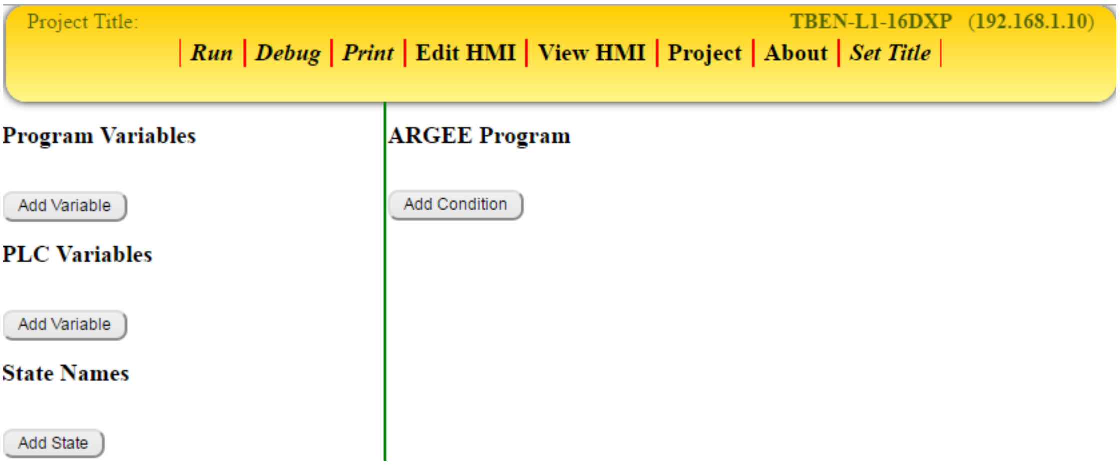
Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

Condition: true

Actions: 0. Coil Coil: IO.Slot1.Output.Output_value_1

Logging in after the user click Run Without Source:



NOTE

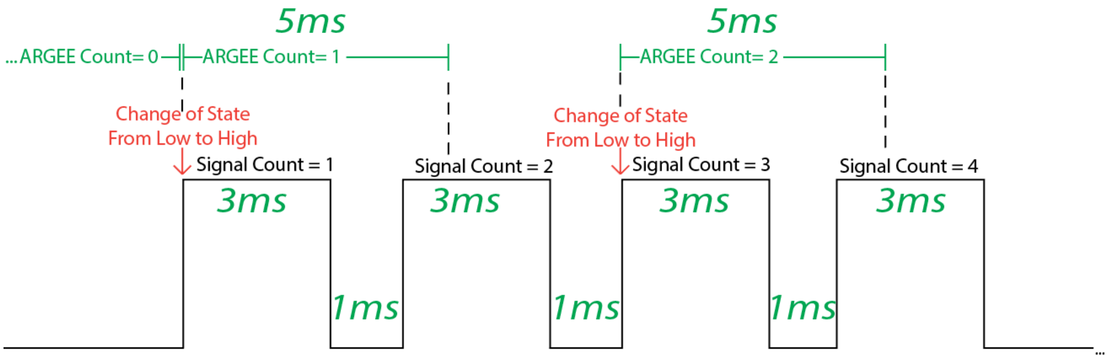
The user needs to save a Master Copy of the program before the user logs out of the environment if the user wants to view/edit the code in the future.

12 System Performance

12.1 Scan Cycle Information

The ARGEE Scan Cycle is typically between 5 – 10 ms depending on the code size. If the user attempts to use ARGEE in an application with scan cycles less than 5 ms, it is possible that ARGEE may miss the signal.

Example of Scan Cycle:



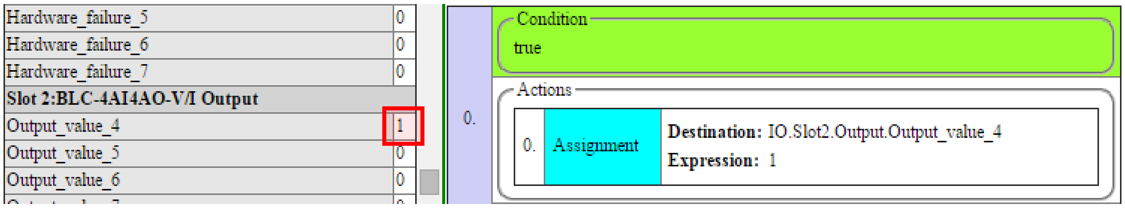
Explaining the Example: In this example, the user is hammering ARGEE with repeated 3 ms signals. Notice that ARGEE does not catch all the signals because the signal is occurring faster than ARGEE's Scan Cycle.

NOTE

ARGEE is not suited for motion control applications.

12.2 IO Variable Formats

IO.Slot2.Output.Output_value_X → This example loads the value “1” into Output 4 bit 0.



Hardware_failure_5	0
Hardware_failure_6	0
Hardware_failure_7	0
Slot 2:BLC-4AI4AO-V/I Output	
Output_value_4	1
Output_value_5	0
Output_value_6	0

0\.

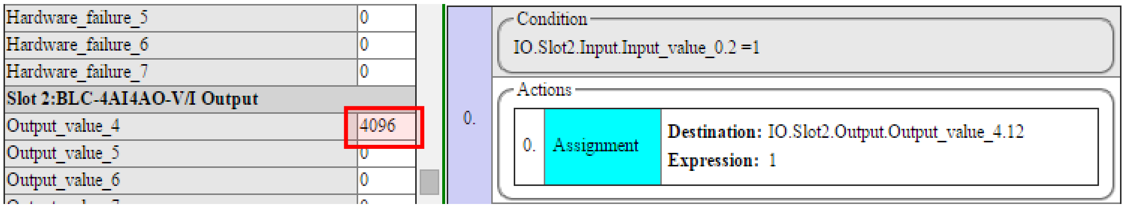
Condition

true

Actions

0. Assignment **Destination:** IO.Slot2.Output.Output_value_4
Expression: 1

IO.Slot2.Output.Output_value_X.Y → In this example, when Input_value_0 Bit_2 equals “1”, a “1” is loaded into Output_value_4 Bit_12.



Hardware_failure_5	0
Hardware_failure_6	0
Hardware_failure_7	0
Slot 2:BLC-4AI4AO-V/I Output	
Output_value_4	4096
Output_value_5	0
Output_value_6	0

0\.

Condition

IO.Slot2.Input.Input_value_0.2 =1

Actions

0. Assignment **Destination:** IO.Slot2.Output.Output_value_4.12
Expression: 1

12.3 How Actions Respond to Conditions

Action	Condition=FALSE	Condition=TRUE
Assignment	No action.	Assigns a destination variable to a result of expression evaluation.
Coil	Resets a variable to 0.	Sets the variable to 1.
Timer start	No action.	If the timer is not started – it starts the timer. Otherwise it restarts the timer. The timer is executed in the background until the accumulator >= “Expires” Preset value.
Timer On	Resets the timer accumulator and Done flag.	If timer Done flag is 0, run the timer. The timer is accumulated every millisecond until the accumulator >=“Expires” Preset value. In that case the Done flag is raised.
Timer Off	If timer Done flag is 0, run the timer. The timer is accumulated every millisecond until the accumulator >=“Expires” Preset value. In that case, the Done flag is raised.	Resets the timer accumulator and Done flag.
Comment	-	-
Count up	Increments the counter whenever the condition changes from false to true.	
Count down	Decrements the counter whenever the condition changes from false to true. (note - the Preset can be a negative value)	
Reset	-	Restarts the counter to – 0.
Trace	-	Record trace information into a trace buffer.

12.4 Defining Variable Types – (Advanced Definitions)

Type	Description	Type	Allowed arithmetic expressions	Specific actions	Size in bytes
Integer Variables	Variables are defined in the program.	All these variables are 32 bit signed integers.	All integer arithmetic	Assignment	4
Retain Integer	Variables which are automatically saved to flash.	All these variables are 32 bit signed integers.	All integer arithmetic	Assignment	8 bytes (4 bytes of data 4 bytes of additional information)
PLC Variables	Variables mapping upper level PLC (Modbus/TCP, EtherNet/IP or PROFINET) exchange data to an integer variable accessible in the program.	They are mapped to integer variables in the program	All integer arithmetic	Assignment	20
Timer/Counter	Timers Counters can be used with appropriate functions, such as “expired”, “count” and appropriate actions such as “Timer On”	All these variables are 32 bit signed integers.	Only used as argument to functions “expired” and “count”	Specific actions: Timer on, Timer off, Start timer, Count up, Count down	8
State	Integer variable that is used to designate states in state machine. Behaves identically to a regular integer variable except for 2 things: 1) Initialize – will list states 2) In the debugger, a state name matching the current value will show up	32 bit integer	All integer arithmetic	Assignment	4

Type	Description	Type	Allowed arithmetic expressions	Specific actions	Size in bytes
Local IO	Input/Output/Diagnostic points	They are mapped to integer variables in the program	All integer arithmetic	Assignment	(not allocated out of 1KB of RAM)
System variables	PLC Connected	32 bit integer		Only 1 bit is used to indicate PLC connected state	4
System variables	Max Cycle time (since program start)	32 bit integer indicating time in ms		Time from the previous cycle to the current cycle.	4

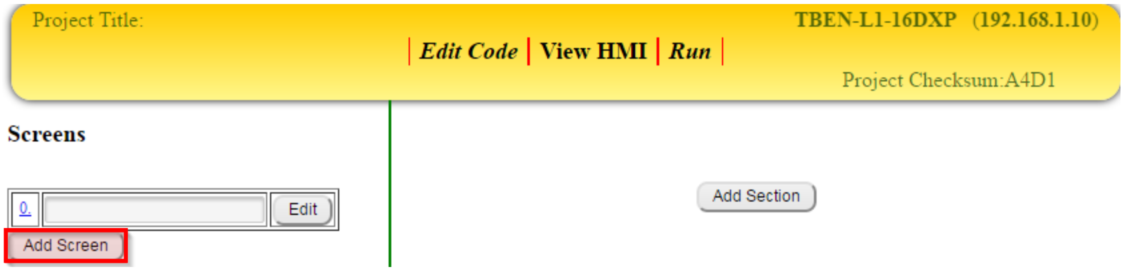
13 ARGEE HMI

Many user applications can be enhanced with the use of the AGREE HMI. The two main ARGEE HMI operations are Editable Fields and Display Fields. General Buttons are used in both types of fields.

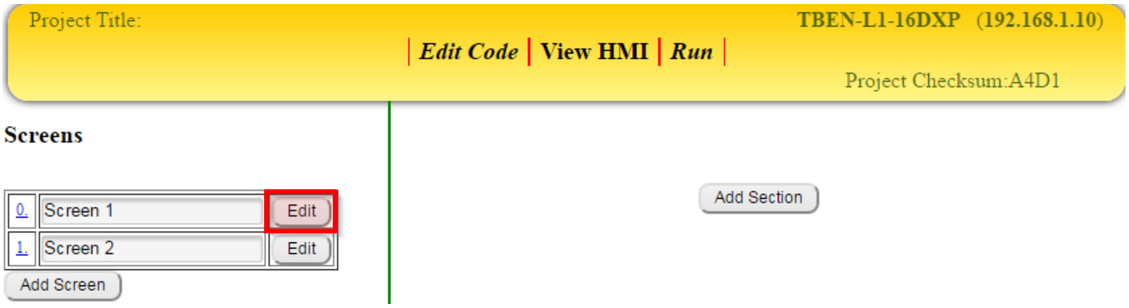
13.1 General Buttons

13.1.1 Add Screen

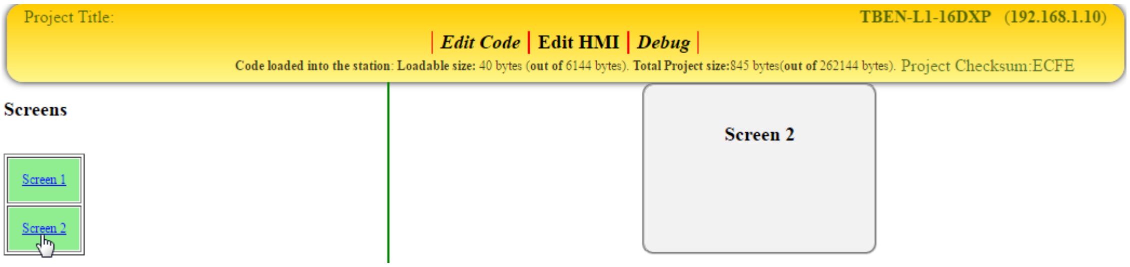
The *Add Screen* button is available under the Edit HMI tab. *Add Screen* allows the user to add several HMI screens to the project.



The user can toggle between multiple screens by clicking on the *Edit* button.



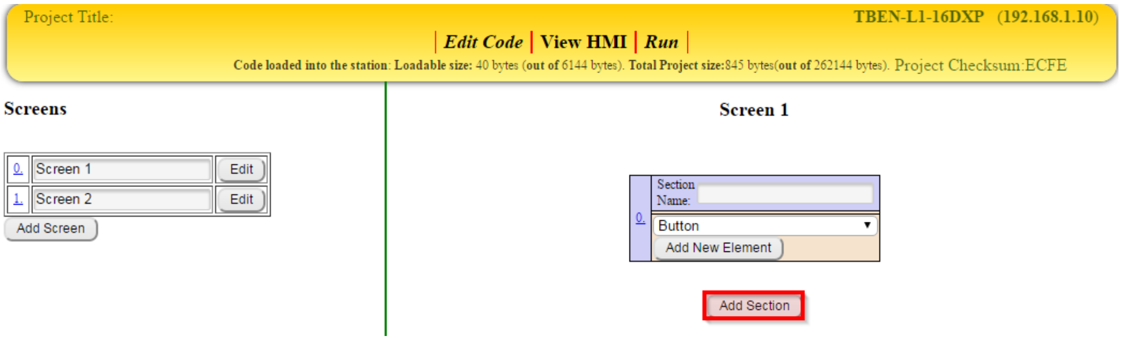
When on the View HMI tab, the user can toggle between screens by clicking on the *Screen* name.



Code loaded into the station: Loadable size: 40 bytes (out of 6144 bytes). Total Project size:845 bytes(out of 262144 bytes).

13.1.2 Add Section

The *Add Section* button is available under the Edit HMI tab. The user will click on the *Add Section* button if they want to add more sections to their HMI screen.

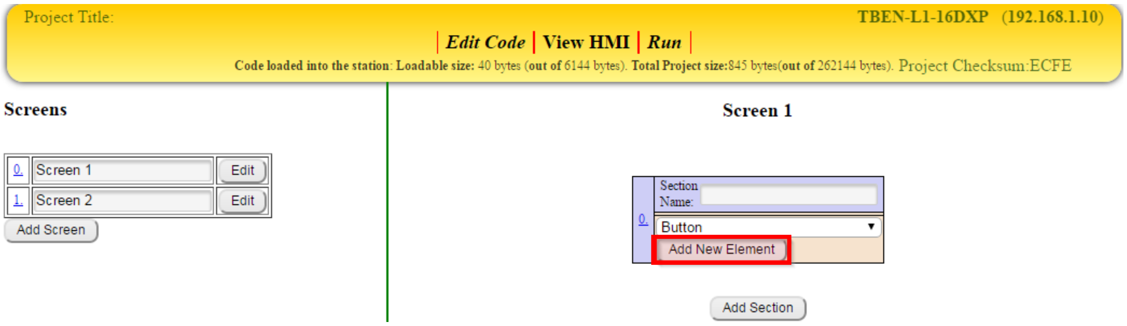


Project Title: TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 40 bytes (out of 6144 bytes). Total Project size:845 bytes(out of 262144 bytes). Project Checksum:ECFE

13.1.3 Add New Element

The user can *Add New Elements* to a specific Section of a specific Screen by selecting an *Element* from the drop down arrow and clicking *Add New Element*.



Project Title: TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 40 bytes (out of 6144 bytes). Total Project size:845 bytes(out of 262144 bytes). Project Checksum:ECFE

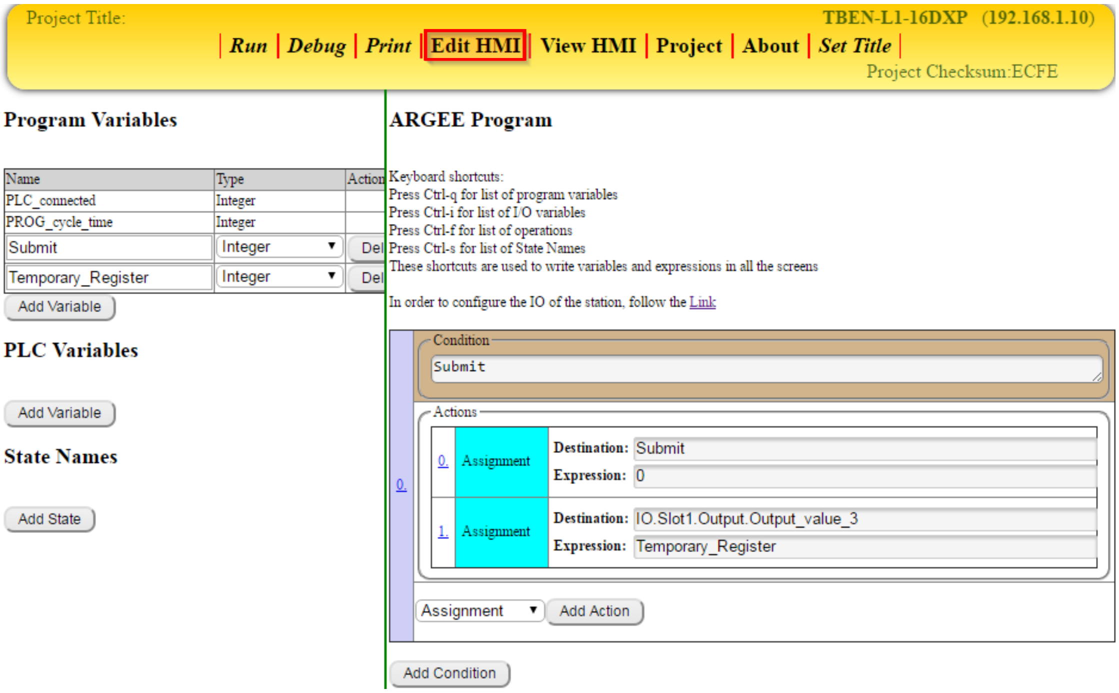
13.2 Editable Fields

13.2.1 Enter Number (and Button)

The *Enter number* element, in conjunction with the *Button* element, allows the user to manually input a value into a register while a program is running.

Example of Enter number (and Button):

- For the HMI to compile, the user must first create some code and then click *Edit HMI* from the ARGEE menu tab.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title |

Project Checksum:ECFE

Program Variables

Name	Type	Action
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer ▼	Del
Temporary_Register	Integer ▼	Del

Add Variable

PLC Variables

Add Variable

State Names

Add State

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

Condition: Submit

Actions

0. Assignment — Destination: Submit — Expression: 0

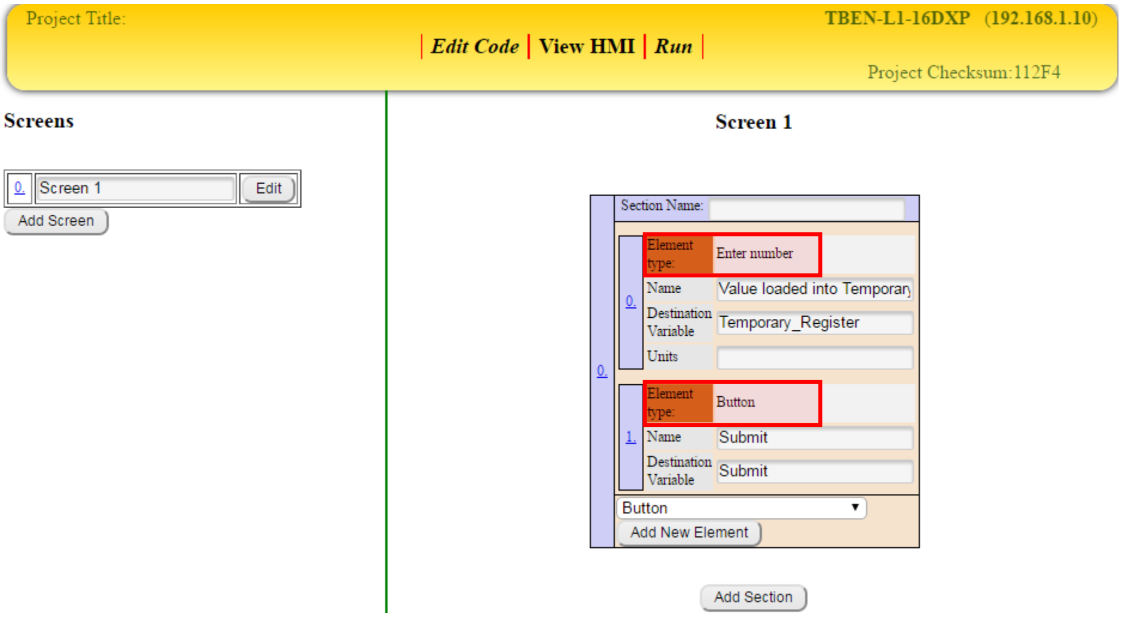
1. Assignment — Destination: IO.Slot1.Output.Output_value_3 — Expression: Temporary_Register

Assignment ▼ Add Action

Add Condition

Explaining the code: The user created two Program Variables: Submit and Temporary_Register. When Submit goes true, the code sets Submit false and loads the value that is in Temporary_Register into Output_value_3.

- The user creates an HMI screen and adds an *Enter number* and *Button* element to it.


Project Title:

TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | **View HMI** | ***Run*** |

Project Checksum:112F4

Screens

0. Screen 1 [Edit]

[Add Screen]

Screen 1

Section Name:

0.

0.
Element type: Enter number
Name: Value loaded into Temporary
Destination Variable: Temporary_Register
Units:

1.
Element type: Button
Name: Submit
Destination Variable: Submit

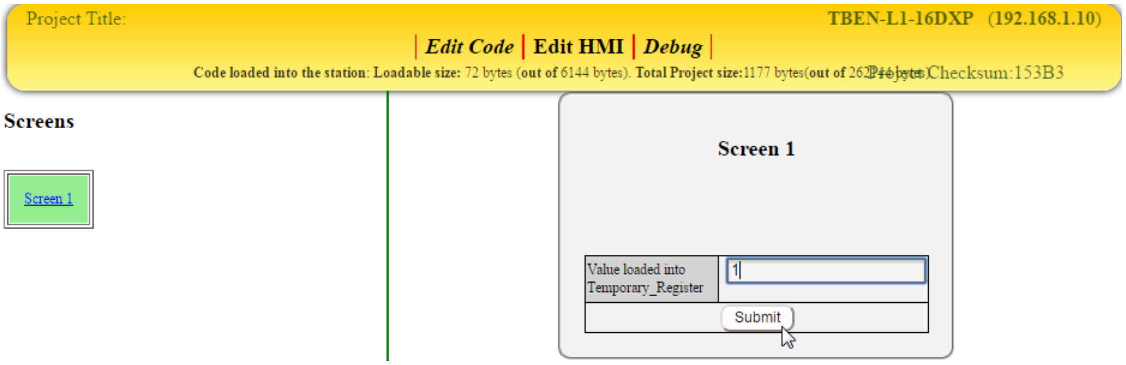
Button ▾

[Add New Element]

[Add Section]

Explaining the Example: The user named the Enter number element “Value loaded into Temporary Register”. The user then set the destination variable to be Temporary_Register. The user then named the Button element “Submit”. The user then set the destination variable to be Submit.

- The user clicks *Run* to download the code to the block and then clicks *View HMI* to view the HMI.


Project Title:

TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | **Edit HMI** | ***Debug*** |

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1177 bytes(out of 262144 bytes) Project Checksum:153B3

Screens

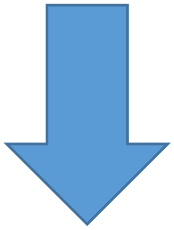
Screen 1

Screen 1

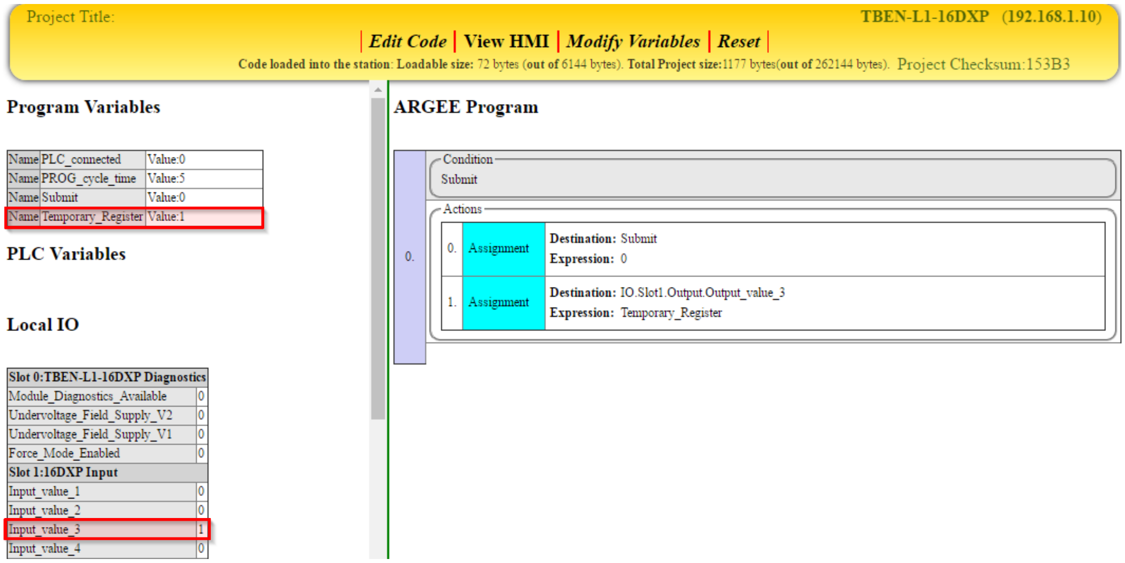
Value loaded into Temporary_Register: 1

[Submit]

Explaining the Example: The user entered the value “1” into the editable field. The user then clicked the Submit Button to load that value into Temporary_Register.



- To observe the bits moving, the user can click on *Debug* and see that Tempory_Register and Output_value_3 have the same values loaded into them.



Project Title: TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1177 bytes(out of 262144 bytes). Project Checksum:153B3

Program Variables

Name	PLC_connected	Value:0
Name	PROG_cycle_time	Value:5
Name	Submit	Value:0
Name	Temporary_Register	Value:1

PLC Variables

Local IO

Slot 0:TBEN-L1-16DXP Diagnostics	
Module_Diagnostics_Available	0
Undervoltage_Field_Supply_V2	0
Undervoltage_Field_Supply_V1	0
Force_Mode_Enabled	0
Slot 1:16DXP Input	
Input_value_1	0
Input_value_2	0
Input_value_3	1
Input_value_4	0

ARGEE Program

0.

Condition
Submit

Actions

0.	Assignment	**Destination:** Submit **Expression:** 0
1.	Assignment	**Destination:** IO.Slot1.Output.Output_value_3 **Expression:** Temporary_Register

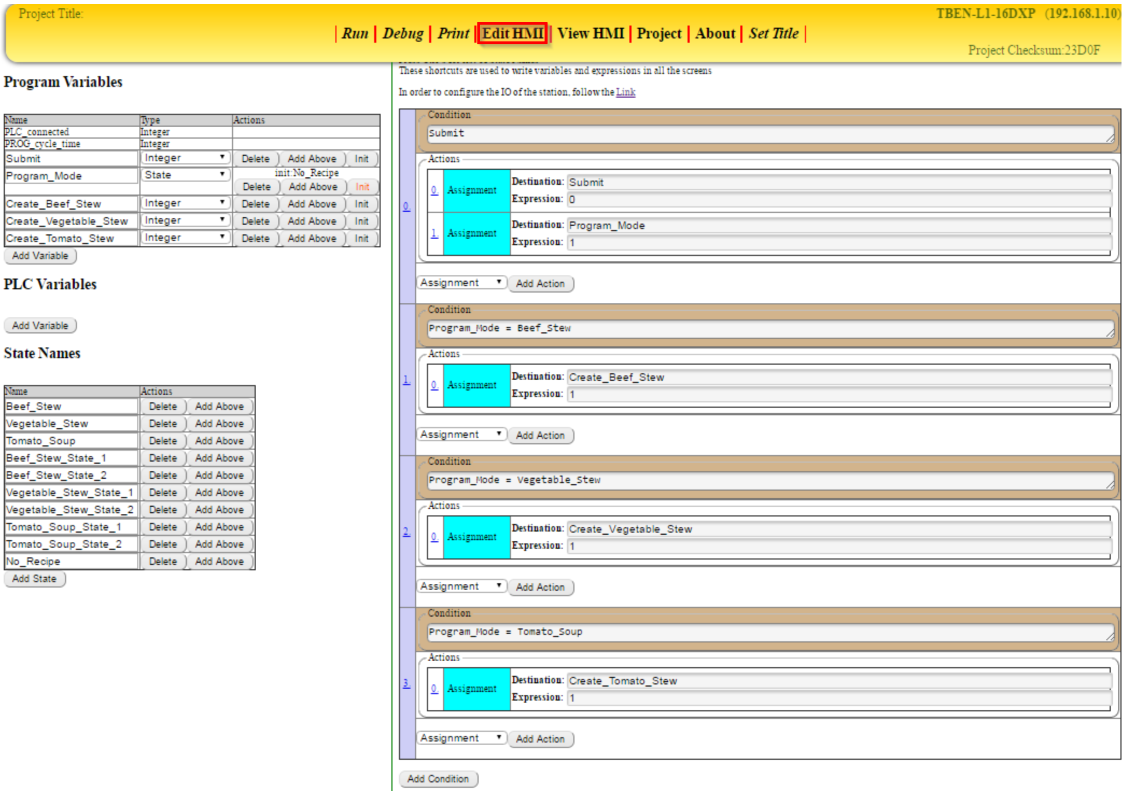
13.2.2 Enter state

The *Enter state* element is used when multiple State Machines are running on the same device. This feature is useful in recipe applications, RFID applications and even pick-to-light applications.

Example of Enter state:

The user wants to toggle between the Beef Stew, Vegetable Stew and Tomato Soup state machines.

➢ The user must first create the code and then click Edit HMI.



Project Title:

TBEN-L1-16DXP (192.168.1.10)

| Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title |

Project Checksum:23D0F

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer	Delete Add Above Init
Program_Mode	State	init:No_Recipe Delete Add Above Init
Create_Beef_Stew	Integer	Delete Add Above Init
Create_Vegetable_Stew	Integer	Delete Add Above Init
Create_Tomato_Stew	Integer	Delete Add Above Init

PLC Variables

State Names

Name	Actions
Beef_Stew	Delete Add Above
Vegetable_Stew	Delete Add Above
Tomato_Soup	Delete Add Above
Beef_Stew_State_1	Delete Add Above
Beef_Stew_State_2	Delete Add Above
Vegetable_Stew_State_1	Delete Add Above
Vegetable_Stew_State_2	Delete Add Above
Tomato_Soup_State_1	Delete Add Above
Tomato_Soup_State_2	Delete Add Above
No_Recipe	Delete Add Above

These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

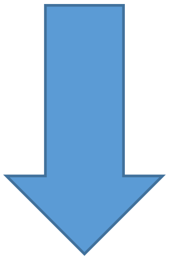
0.
Condition: Submit
Actions:
0. Assignment — Destination: Submit, Expression: 0
1. Assignment — Destination: Program_Mode, Expression: 1

1.
Condition: Program_Mode = Beef_Stew
Actions:
0. Assignment — Destination: Create_Beef_Stew, Expression: 1

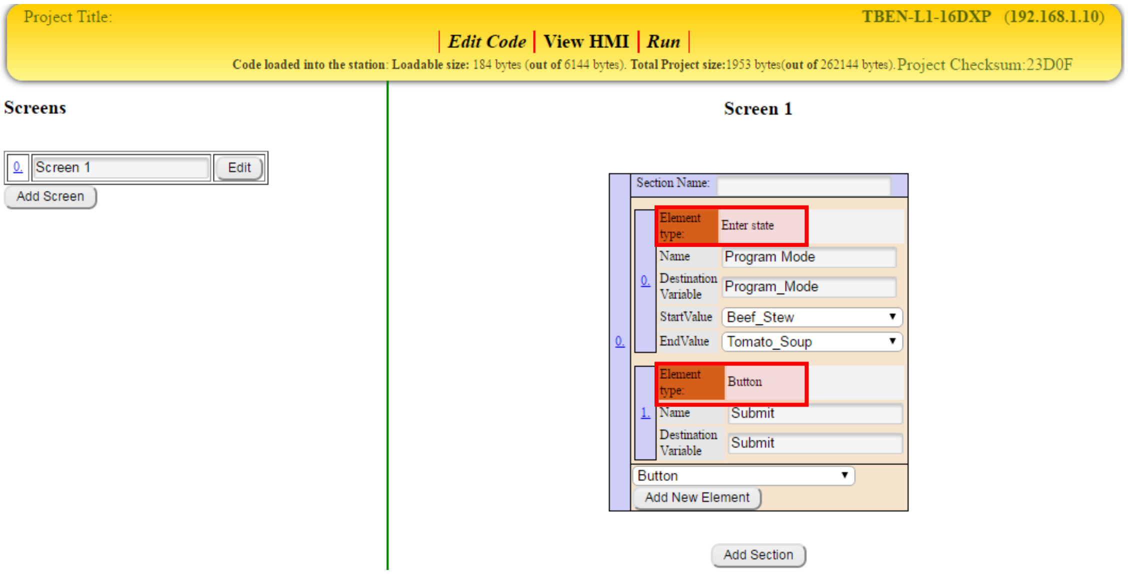
2.
Condition: Program_Mode = Vegetable_Stew
Actions:
0. Assignment — Destination: Create_Vegetable_Stew, Expression: 1

3.
Condition: Program_Mode = Tomato_Soup
Actions:
0. Assignment — Destination: Create_Tomato_Stew, Expression: 1

Explaining the code: The user created three State Machines (Beef_Stew, Vegetable_Stew and Tomato_Soup). Each State Machine has its own individual Sub-States (Beef_Stew_State_1/2, Vegetable_Stew_State_1/2, Tomato_Soup_State_1/2) associated with it. The user created five Program Variables. When Submit goes true, the code sets Submit false and loads the value "1" into Program Mode. When Program Mode goes true, it loads the value "1" into the selected stews State Machine. The other three Program Variables (Create_Beef_Stew, Create_Vegetable_Stew, and Create_Tomato_Soup) were created to signify the specific type of stews being created. They don't actually do anything in this code.



- ➢ The user creates an HMI screen and adds an *Enter state* and *Button* element to it.



Project Title: TBEN-L1-16DXP (192.168.1.10)

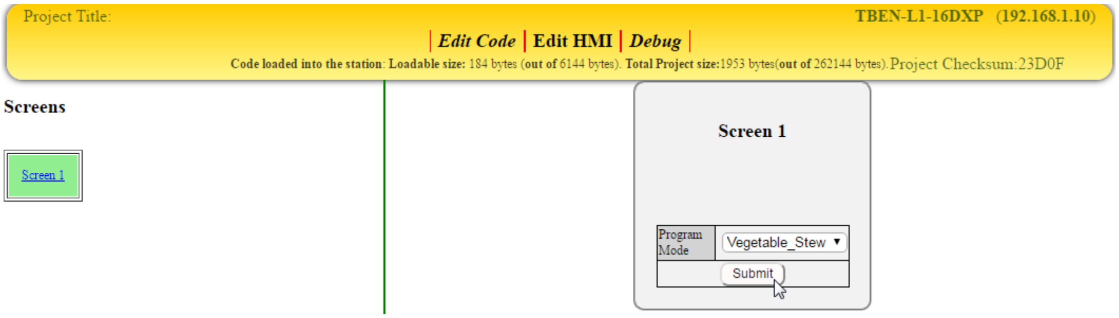
Code loaded into the station: Loadable size: 184 bytes (out of 6144 bytes). Total Project size:1953 bytes(out of 262144 bytes). Project Checksum:23D0F

Screen 1

Section Name:	
Element type:	Enter state
Name	Program Mode
Destination Variable	Program_Mode
StartValue	Beef_Stew
EndValue	Tomato_Soup
Element type:	Button
Name	Submit
Destination Variable	Submit

Explaining the Example: The user named the *Enter state* element “Program Mode”. The user then set the *Destination Variable* to be Program_Mode. The user used the *StartValue* and *EndValue* to set the limits on the states that the user wants to display in the HMI drop down menu. The user then named the Button element “Submit”. The user then set the *Destination Variable* to be Submit.

- ➢ The user clicks *Run* to download the code to the block and then clicks *View HMI* to view the HMI.



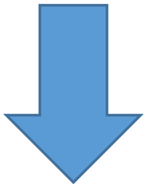
Project Title: TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 184 bytes (out of 6144 bytes). Total Project size:1953 bytes(out of 262144 bytes). Project Checksum:23D0F

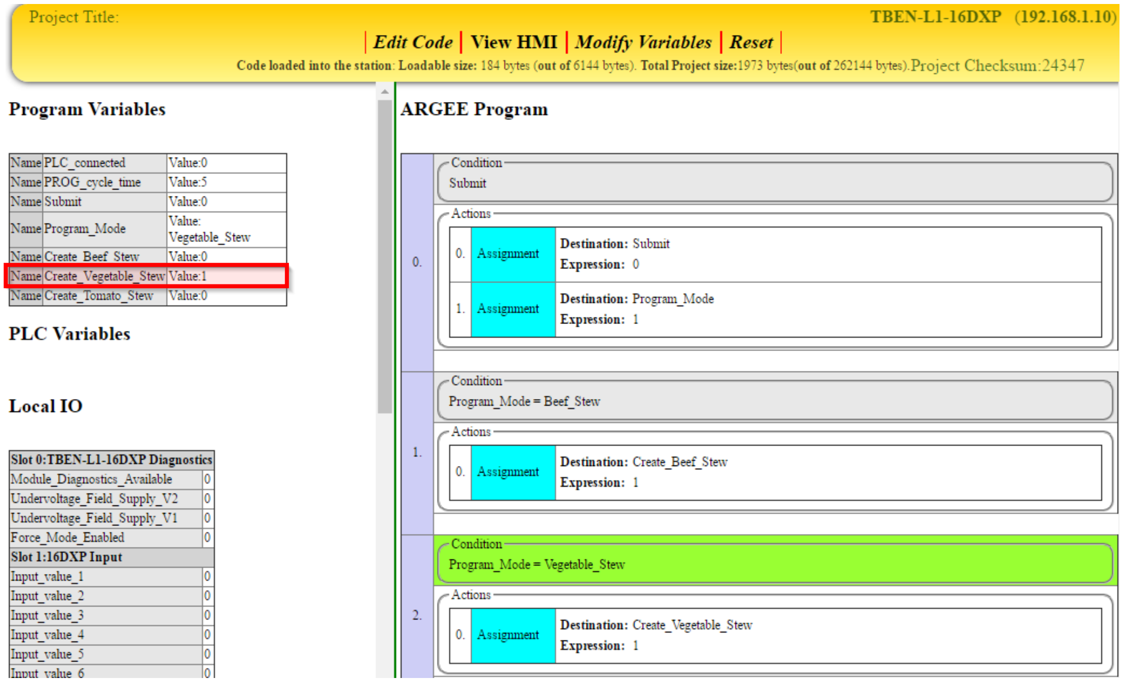
Screen 1

Program Mode	Vegetable_Stew

Explaining the Example: The user selects the recipe from the drop down arrow and then clicks the Submit Button to execute the Vegetable _Stew State Machine.



To observe the bits moving, the user can click on *Debug* and see that the Create_Vegetable_Stew register is true.



Project Title: TBEN-L1-16DXP (192.168.1.10)

Code loaded into the station: Loadable size: 184 bytes (out of 6144 bytes). Total Project size:1973 bytes(out of 262144 bytes).Project Checksum:24347

Program Variables

Name	PLC_connected	Value:0
Name	PROG_cycle_time	Value:5
Name	Submit	Value:0
Name	Program_Mode	Value: Vegetable_Stew
Name	Create_Beef_Stew	Value:0
Name	Create_Vegetable_Stew	Value:1
Name	Create_Tomato_Stew	Value:0

PLC Variables

Local IO

Slot 0:TBEN-L1-16DXP Diagnostics	
Module_Diagnostics_Available	0
Undervoltage_Field_Supply_V2	0
Undervoltage_Field_Supply_V1	0
Force_Mode_Enabled	0
Slot 1:16DXP Input	
Input_value_1	0
Input_value_2	0
Input_value_3	0
Input_value_4	0
Input_value_5	0
Input_value_6	0

ARGEE Program

0.

Condition
Submit

Actions

0.	Assignment	**Destination:** Submit **Expression:** 0
1.	Assignment	**Destination:** Program_Mode **Expression:** 1

1.

Condition
Program_Mode = Beef_Stew

Actions

0.	Assignment	**Destination:** Create_Beef_Stew **Expression:** 1

2.

Condition
Program_Mode = Vegetable_Stew

Actions

0.	Assignment	**Destination:** Create_Vegetable_Stew **Expression:** 1

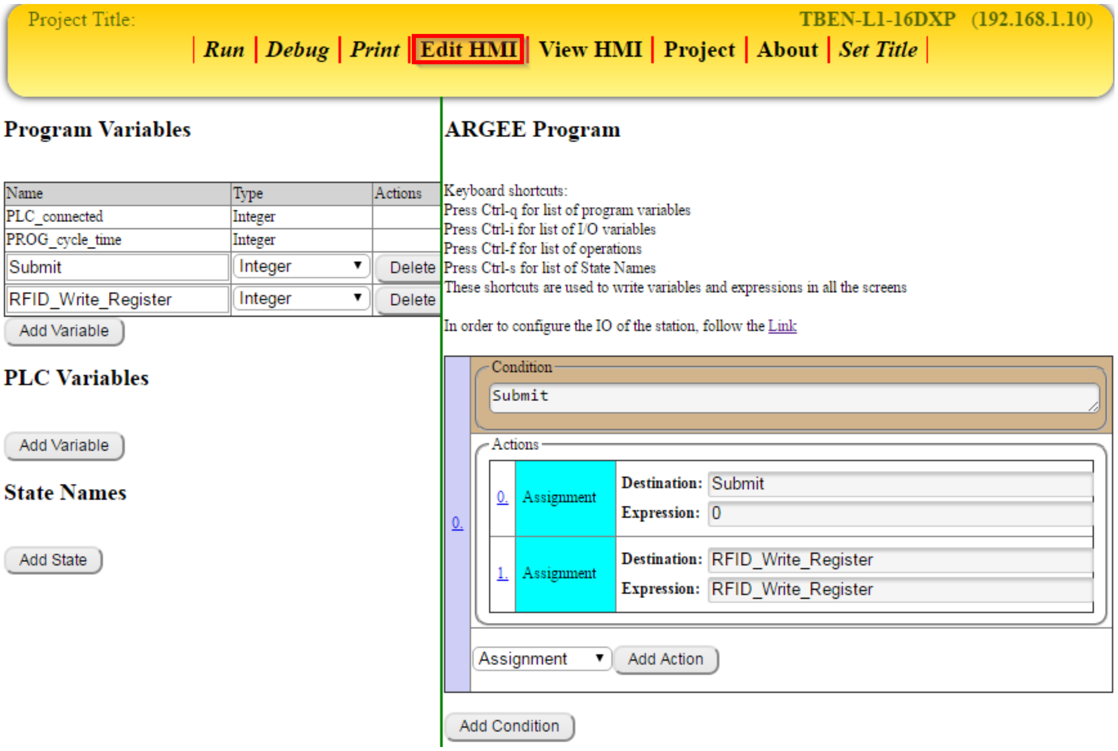
13.2.3 Edit hex number

The *Edit hex number* element allows the user to manually input a value (in Hex) into a register while a program is running.

Example of Edit hex number:

The user wants to load data into the RFID Write Register.

➢ After the code is written, the user clicks on the Edit HMI tab.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Add Variable
Add Variable
Add State
Add Action
Add Condition
Delete

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer	Delete
RFID_Write_Register	Integer	Delete

PLC Variables

State Names

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

0.

Condition: Submit

Actions

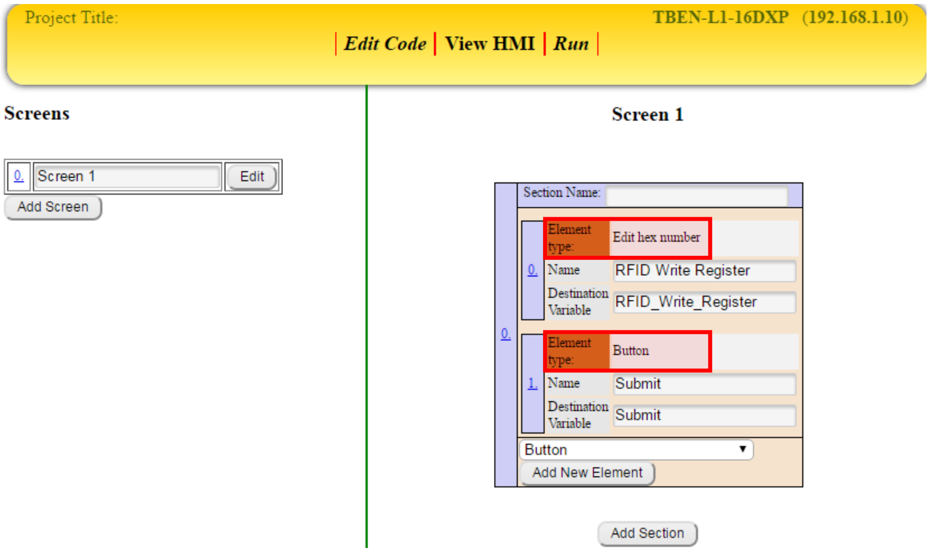
0. Assignment — Destination: Submit — Expression: 0

1. Assignment — Destination: RFID_Write_Register — Expression: RFID_Write_Register

Assignment

Explaining the code: The user created two Program Variables, Submit and RFID_Write_Register. When Submit goes true, the code sets Submit false and loads the value that is in RFID_Write_Register into RFID_Write_Register.

- The user creates an HMI screen and adds an *Edit hex number* and *Button* element to it.

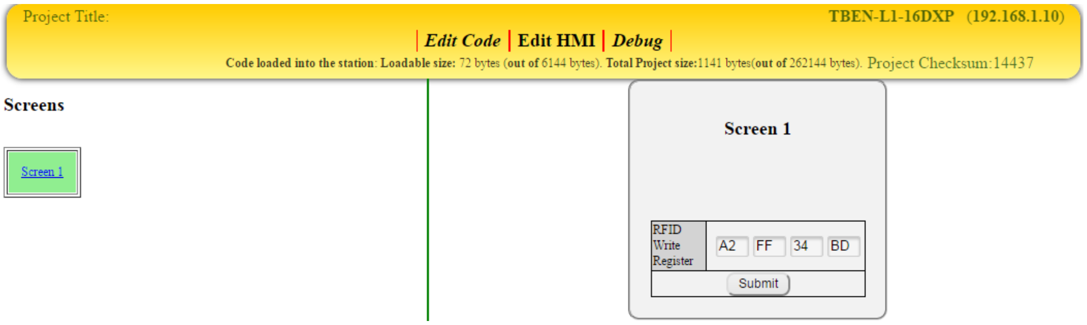


Section Name:

0.	Element type:	Edit hex number
	Name	RFID Write Register
	Destination Variable	RFID_Write_Register
1.	Element type:	Button
	Name	Submit
	Destination Variable	Submit

Explaining the Example: The user named the Edit hex number element "RFID Write Register", and then set the destination variable to be RFID_Write_Register. The user then named the Button element "Submit", and then set the destination variable to be Submit.

- The user clicks Run to download the code to the block and then clicks View HMI to view the HMI.

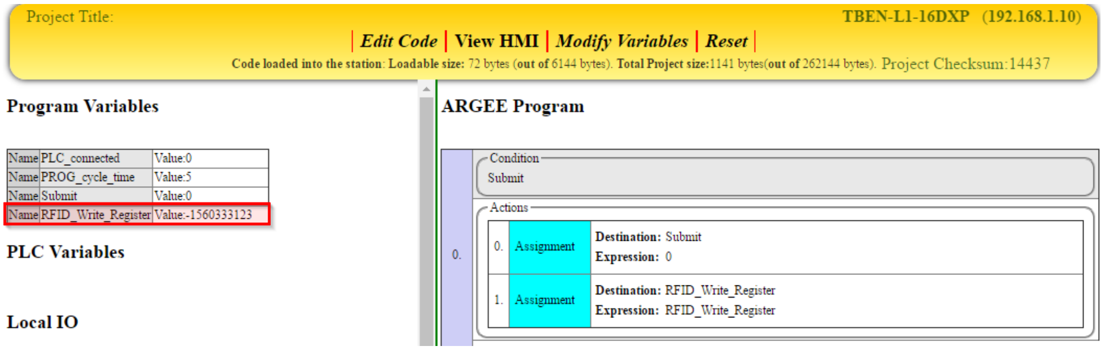

Project Title:
TBEN-L1-16DXP (192.168.1.10)
| Edit Code | Edit HMI | Debug |
Screens
Screen 1
Screen 1
Submit

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1141 bytes(out of 262144 bytes). Project Checksum:14437

RFID Write Register: A2 FF 34 BD

Explaining the Example: The user entered the hex value "A2 FF 34 BD" into the editable field. The user then clicked the Submit Button to load that value into Temporary_Register_1.

To observe the bits moving, the user can click *Debug* and see that RFID_Write_Register has the hex number A2 FF 34 BD loaded into it.



Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1141 bytes(out of 262144 bytes). Project Checksum:14437

Name	PLC_connected	Value:0
Name	PROG_cycle_time	Value:5
Name	Submit	Value:0
Name	RFID_Write_Register	Value:-1560333123

Condition: Submit

0.

0.	Assignment	Destination: Submit Expression: 0
1.	Assignment	Destination: RFID_Write_Register Expression: RFID_Write_Register

13.3 Display Fields

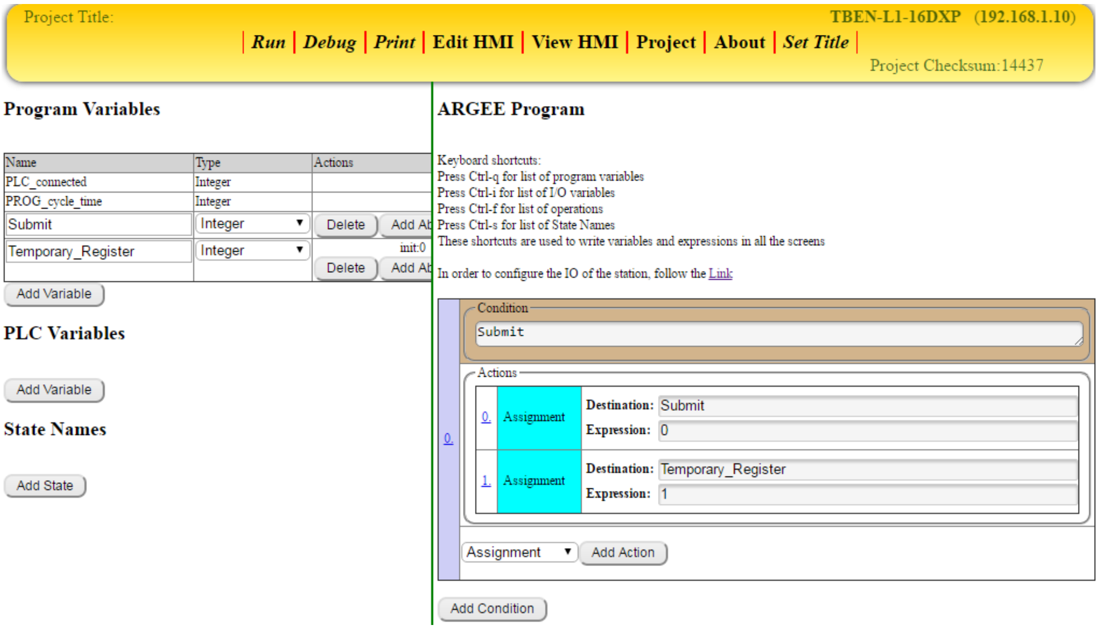
The Display Field elements in the ARGEE HMI are Display number or state, Display number with valid range, Display hex number.

13.3.1 Display number or state

The *Display number or state element* is a feature that allows the user to see the current value in a particular register.

Example of Display number or state:

The user wants to monitor the value in Temporary_Register.


Project Title:
TBEN-L1-16DXP (192.168.1.10)
Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title
Project Checksum:14437

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer	Delete Add Ab
Temporary_Register	Integer	init:0 Delete Add Ab

Add Variable
PLC Variables
Add Variable
State Names
Add State

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

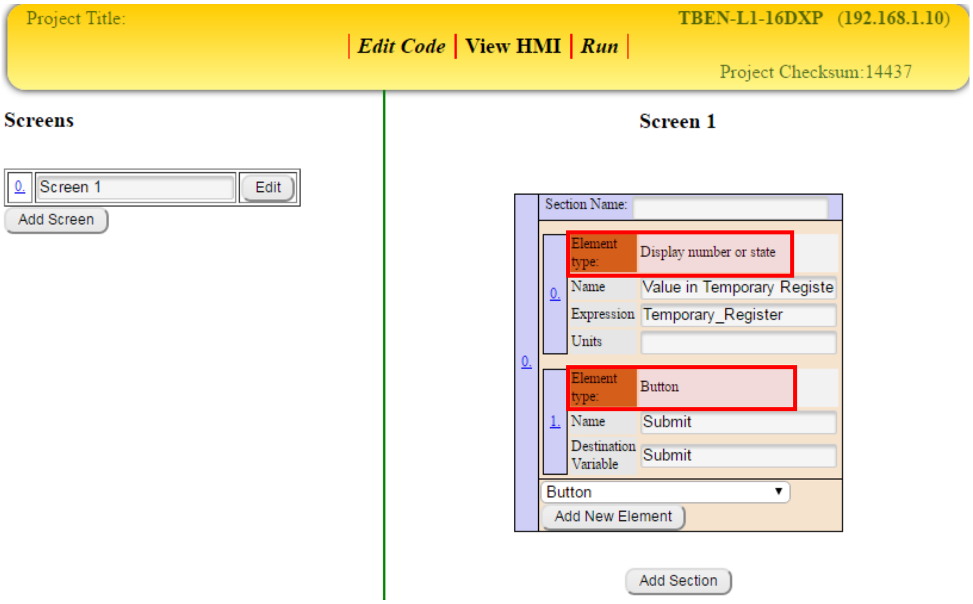
Condition: Submit

Actions:

0	Assignment	Destination:	Submit
		Expression:	0
1	Assignment	Destination:	Temporary_Register
		Expression:	1

Explaining the code: The user created two Program Variables, Submit and Temporary_Register. When Submit goes true, the code sets Submit false and loads the value “1” into Temporary_Register.

- The user creates an HMI screen and adds a *Display number or state* and *Button* element to it.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | View HMI | *Run* |

Project Checksum:14437

Screens

0. Screen 1 Edit

Add Screen

Screen 1

Section Name:

0.

0.	Element type:	Display number or state
	Name	Value in Temporary Registe
	Expression	Temporary_Register
	Units	
1.	Element type:	Button
	Name	Submit
	Destination Variable	Submit

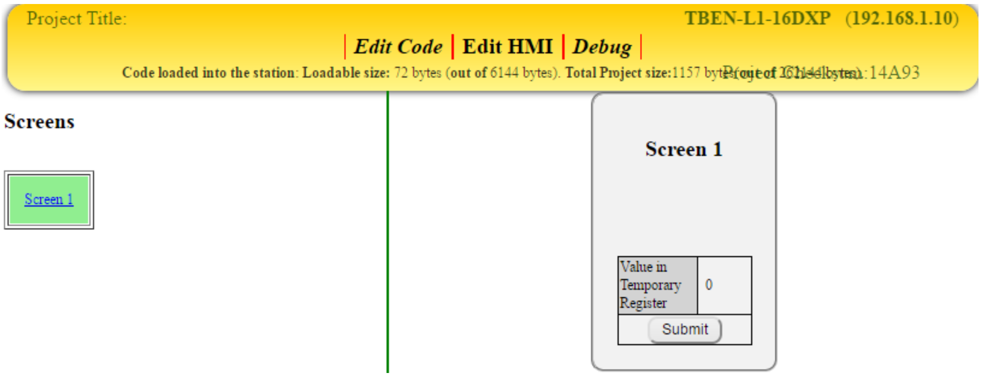
Button ▾

Add New Element

Add Section

Explaining the Example: The user named the Display number or state element “Value in Temporary Register”, and then sets the destination variable to be Temporary_Register. Next, the user named the Button element “Submit”, and then set the destination variable to be Submit.

- The user clicks *Run* to download the code to the block and then clicks *View HMI* to view the HMI.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | Edit HMI | *Debug* |

Code loaded into the station: Loadable size: 72 bytes (**out of** 6144 bytes). **Total Project size:**1157 bytes(**out of** 262144 bytes).Project Checksum:14A93

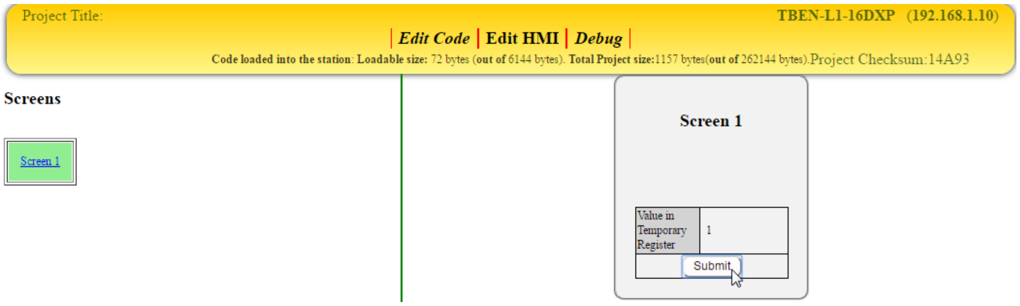
Screens

Screen 1

Screen 1

Value in Temporary Register	0

Submit



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | Edit HMI | *Debug* |

Code loaded into the station: Loadable size: 72 bytes (**out of** 6144 bytes). **Total Project size:**1157 bytes(**out of** 262144 bytes).Project Checksum:14A93

Screens

Screen 1

Screen 1

Value in Temporary Register	1

Submit

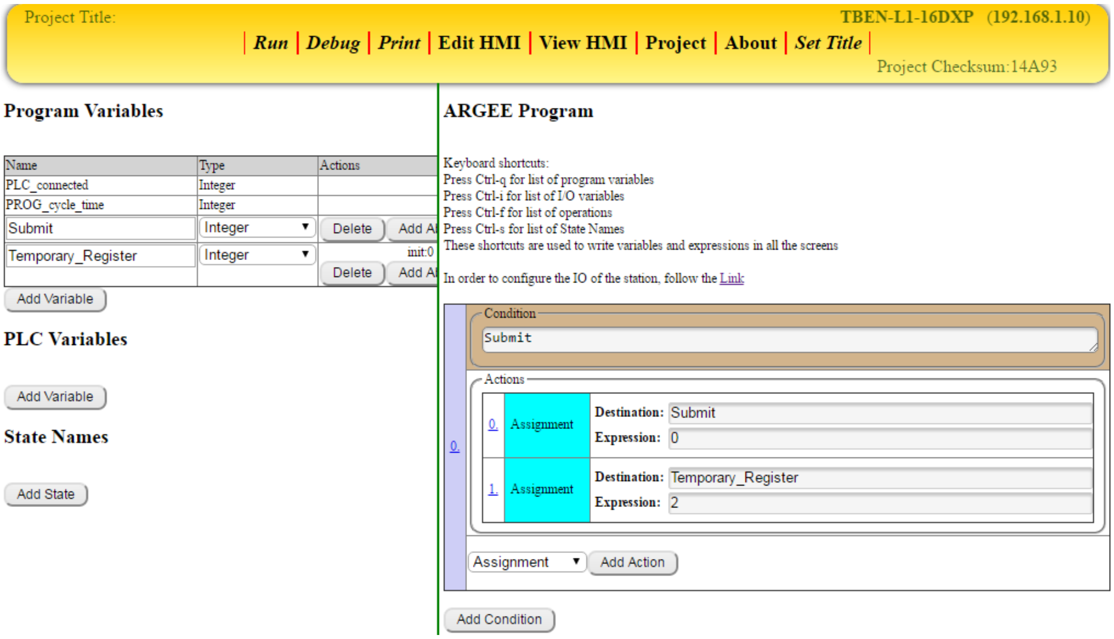
Explaining the Example: When the HMI first loads up, the value “0” is in Temporary_Register. When the user presses the Submit button, the value “1” is loaded in to Temporary_Register.

13.3.2 Display number with valid range

The *Display number with valid range* element is a feature that allows the user to see the current value in a particular register. It also lets the user know when that value has exceeded a preset range.

Example of Display number with valid range:

The user wants to monitor the value in Temporary_Register. The user also wants a visual notification when that value has exceeded a preset range.



Project Title:

TBEN-L1-16DXP (192.168.1.10)

| Run | Debug | Print | Edit HMI | View HMI | Project | About | Set Title |

Project Checksum:14A93

Program Variables

Name	Type	Actions
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer	Delete Add A
Temporary_Register	Integer	init:0 Delete Add A

PLC Variables

State Names

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

0.

Condition: Submit

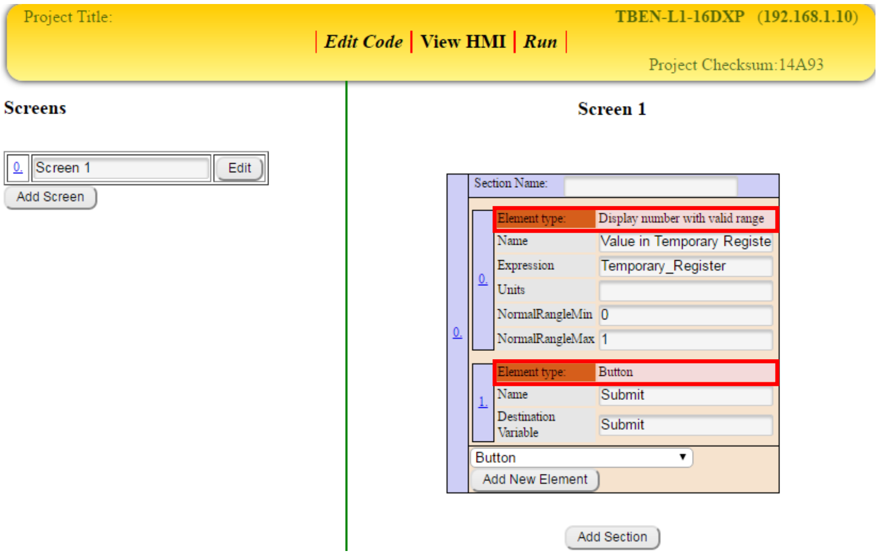
Actions

0. Assignment
Destination: Submit
Expression: 0

1. Assignment
Destination: Temporary_Register
Expression: 2

Explaining the code: The user created two Program Variables, Submit and Temporary_Register. When Submit goes true, the code sets Submit false and loads the value “2” into Temporary_Register.

- The user creates an HMI screen and adds a *Display number with valid range* and *Button* element to it.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | View HMI | *Run* |

Project Checksum:14A93

Screens

0. Screen 1 [Edit]

[Add Screen]

Screen 1

Section Name:

0. 0.

Element type:	Display number with valid range
Name	Value in Temporary Registe
Expression	Temporary_Register
Units	
NormalRangleMin	0
NormalRangleMax	1

1.

Element type:	Button
Name	Submit
Destination Variable	Submit

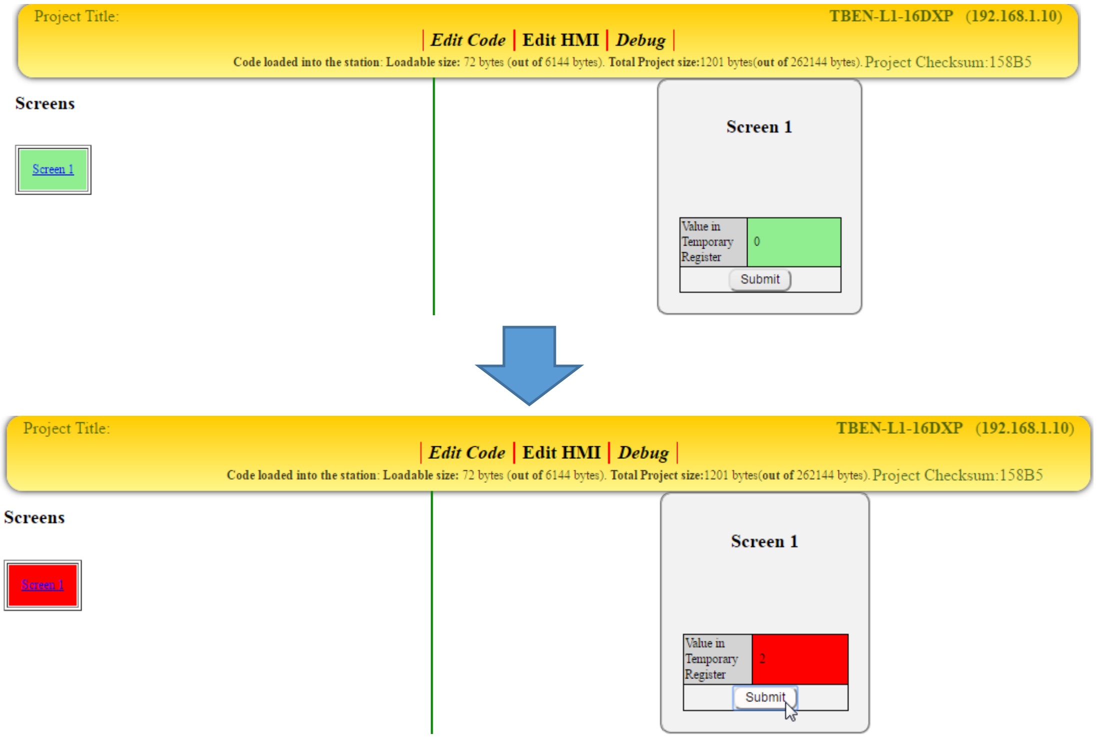
Button ▾

[Add New Element]

[Add Section]

Explaining the Example: The user named the Display number with valid range element "Value in Temporary Register". The user then set the destination variable to be Temporary_Register. The user set the range minimum to be "0" and the range maximum to be "1". The user then named the Button element "Submit", and set the destination variable to be Submit.

- The user clicks *Run* to download the code to the block and then clicks *View HMI* to view the HMI.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | Edit HMI | *Debug* |

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1201 bytes(out of 262144 bytes). Project Checksum:158B5

Screens

Screen 1

Screen 1

Value in Temporary Register	0

[Submit]

Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | Edit HMI | *Debug* |

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1201 bytes(out of 262144 bytes). Project Checksum:158B5

Screens

Screen 1

Screen 1

Value in Temporary Register	2

[Submit]

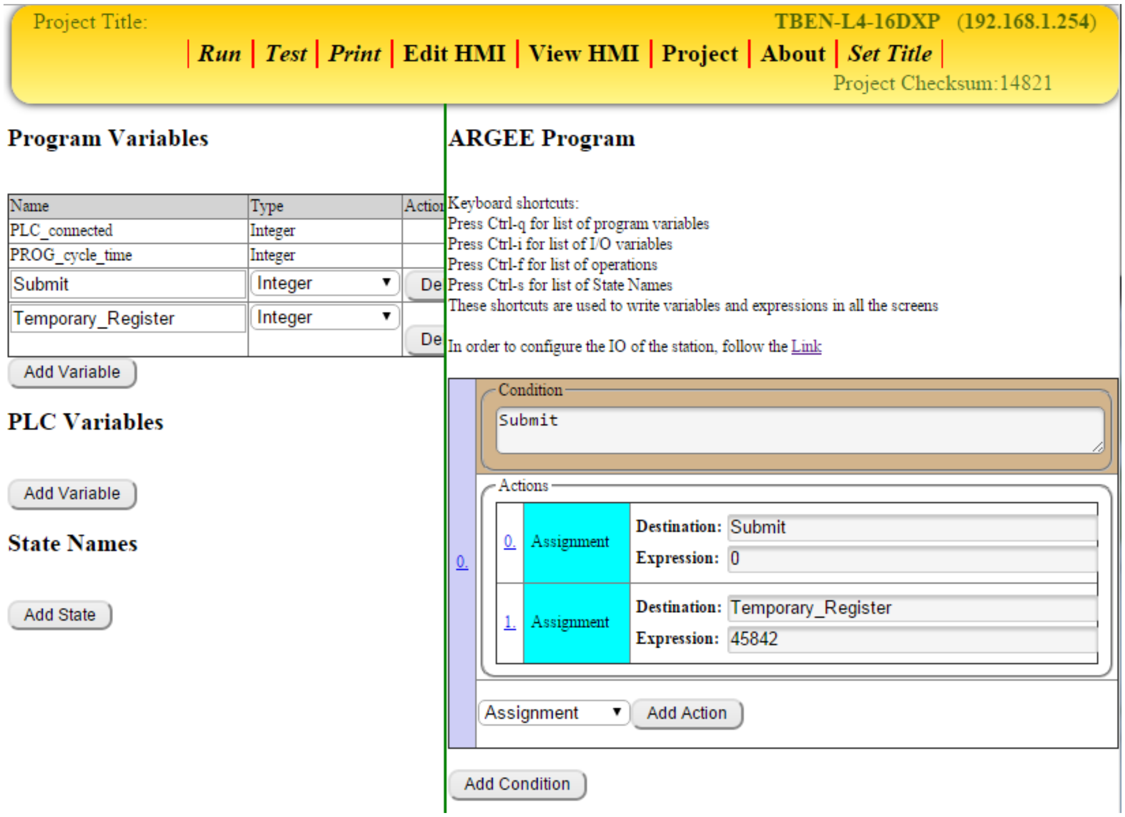
Explaining the Example: When the HMI first loads up, the value "0" is in Temporary_Register. When the user presses the Submit button, the value "2" is loaded in to Temporary_Register. The value "2" exceeds the preset maximum so the HMI goes red.

13.3.3 Display hex number

The *Display hex number* element is a feature that allows the user to see the current value in a particular register displayed in Hex.

Example of Display hex number:

The user wants to monitor the value in Temporary_Register. The user also wants to display that value in hex.


Project Title:
TBEN-L4-16DXP (192.168.1.254)
| Run | Test | Print | Edit HMI | View HMI | Project | About | Set Title |
Project Checksum:14821
Add Variable
Add Variable
Add State
Add Action
Add Condition

Program Variables

Name	Type	Action
PLC_connected	Integer	
PROG_cycle_time	Integer	
Submit	Integer ▼	De
Temporary_Register	Integer ▼	De

PLC Variables

State Names

ARGEE Program

Keyboard shortcuts:
Press Ctrl-q for list of program variables
Press Ctrl-i for list of I/O variables
Press Ctrl-f for list of operations
Press Ctrl-s for list of State Names
These shortcuts are used to write variables and expressions in all the screens

In order to configure the IO of the station, follow the Link

0.

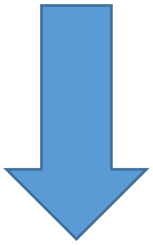
Condition: Submit

Actions

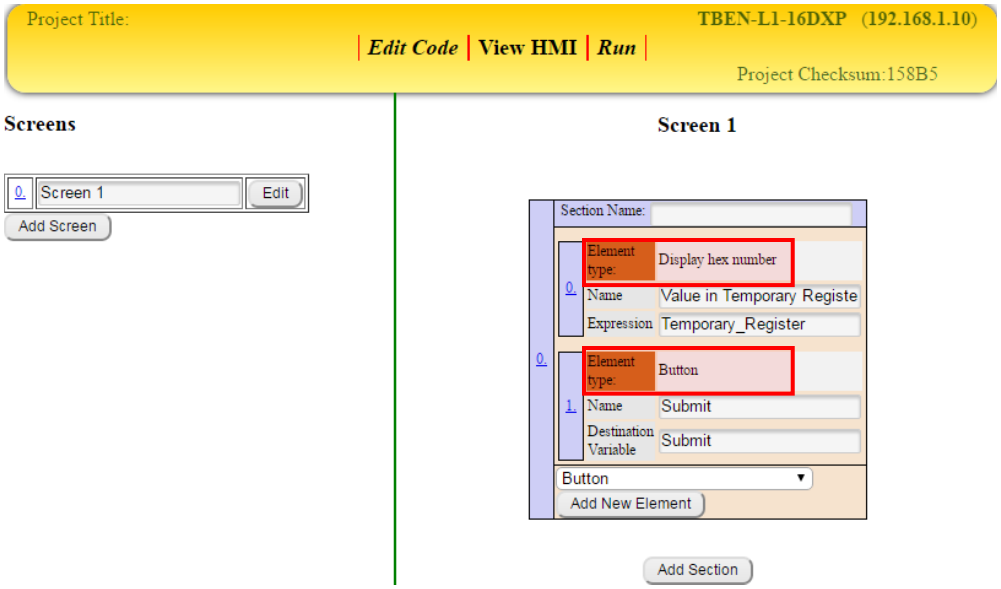
0.	Assignment	Destination: Submit Expression: 0
1.	Assignment	Destination: Temporary_Register Expression: 45842

Assignment ▼

Explaining the code: The user created two Program Variables, Submit and Temporary_Register. When Submit goes true, the code sets Submit false and loads the value “45842” into Temporary_Register.



- The user creates an HMI screen and adds a Display number or state and Button element to it.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | **View HMI** | ***Run*** |

Project Checksum:158B5

Screens

0. Screen 1 Edit

Add Screen

Screen 1

Section Name:

0.	Element type:	Display hex number
	Name	Value in Temporary Registe
	Expression	Temporary_Register
1.	Element type:	Button
	Name	Submit
	Destination Variable	Submit

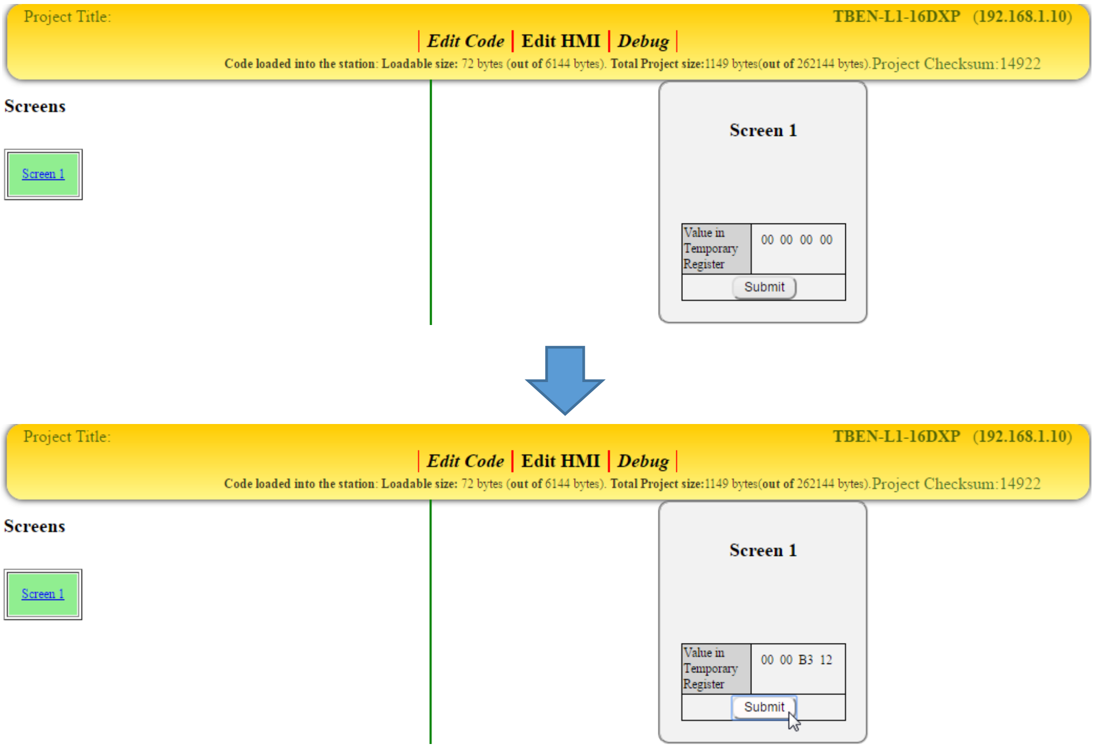
Button ▼

Add New Element

Add Section

Explaining the Example: The user named the Display hex number "Value in Temporary Register", and then set the destination variable to be Temporary_Register. Next, the user named the Button element "Submit" and set the destination variable to be Submit.

- The user clicks Run to download the code to the block and then clicks View HMI to view the HMI.



Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | **Edit HMI** | ***Debug*** |

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1149 bytes(out of 262144 bytes).Project Checksum:14922

Screens

Screen 1

Screen 1

Value in Temporary Register	00 00 00 00
Submit	

Project Title: TBEN-L1-16DXP (192.168.1.10)

| *Edit Code* | **Edit HMI** | ***Debug*** |

Code loaded into the station: Loadable size: 72 bytes (out of 6144 bytes). Total Project size:1149 bytes(out of 262144 bytes).Project Checksum:14922

Screens

Screen 1

Screen 1

Value in Temporary Register	00 00 B3 12
Submit	

Explaining the Example: When the HMI first loads up, the value "00 00 00 00" is in Temporary_Register. When the user presses the Submit button, the value "45842" is loaded in to Temporary_Register. The value "45842" is transformed into hex and is displayed in the HMI.

14 Common Applications

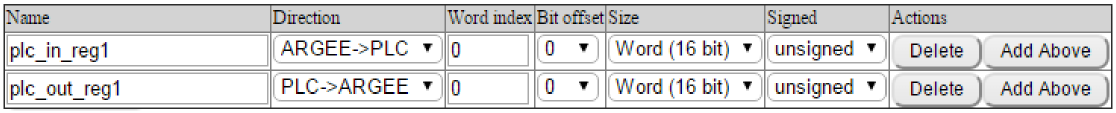
14.1 Communicating with an EtherNet/IP Master – Allen Bradley

ARGEE blocks have the ability to communicate with an EtherNet/IP (E/IP) Master. The E/IP Master can establish communication via connection points 101 & 110.

Example of Communicating with an EtherNet/IP Master:

1. The user wants to check and see if data is being passed back and forth between the ARGEE block and the E/IP Master. The first thing the user does is set up PLC variables.

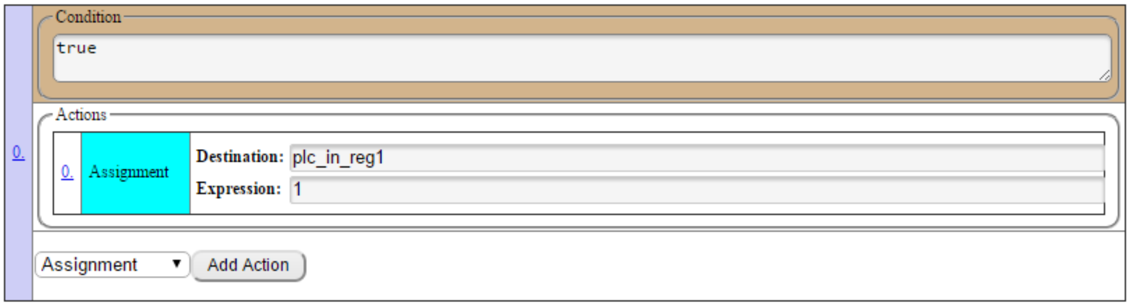
PLC Variables:



Name	Direction	Word index	Bit offset	Size	Signed	Actions
plc_in_reg1	ARGEE->PLC	0	0	Word (16 bit)	unsigned	Delete Add Above
plc_out_reg1	PLC->ARGEE	0	0	Word (16 bit)	unsigned	Delete Add Above

Explaining the Set-up: The user creates two PLC Variables, and they set the direction the data will travel. Data transmitted from ARGEE to the PLC is mapped into AB PLC Instance 101 and the data size is defined in the ARGEE PLC variable section (ARGEE->PLC). Data transmitted from the PLC to ARGEE is mapped into AB PLC Instance 110 and the data size is defined in the ARGEE PLC variable section (PLC->ARGEE).

2. The next step is to write the ARGEE code.



0.

Condition

```
true
```

Actions

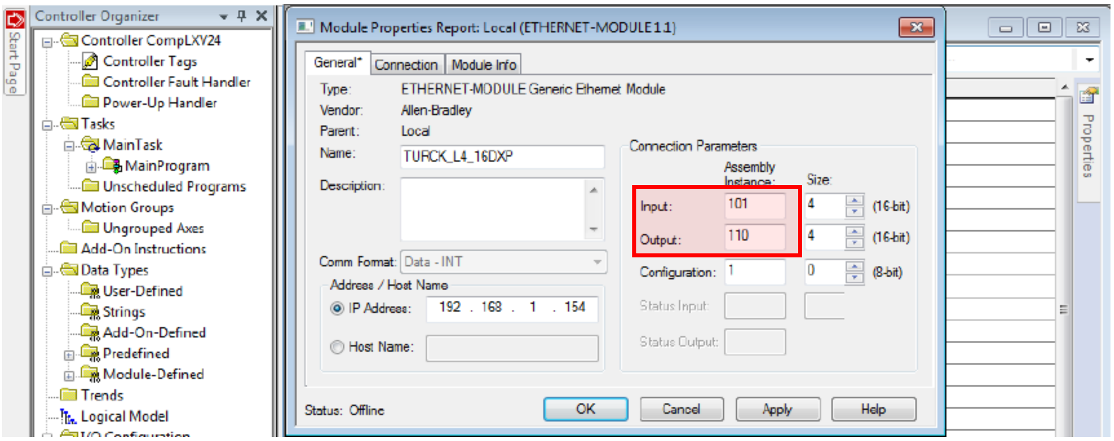
0. Assignment

Destination: plc_in_reg1

Expression: 1

Explaining the Code: The user wrote the value “1” into plc_in_reg1.

3. The third step is to set up the connection points inside the PLC.



Module Properties Report: Local (ETHERNET-MODULE 1.1)

Type: ETHERNET-MODULE Generic Ethernet Module

Vendor: Allen-Bradley

Parent: Local

Name: TURCK_L4_16DXP

Description:

Comm Format: Data - INT

Address / Host Name

● IP Address: 192 . 168 . 1 . 154

○ Host Name:

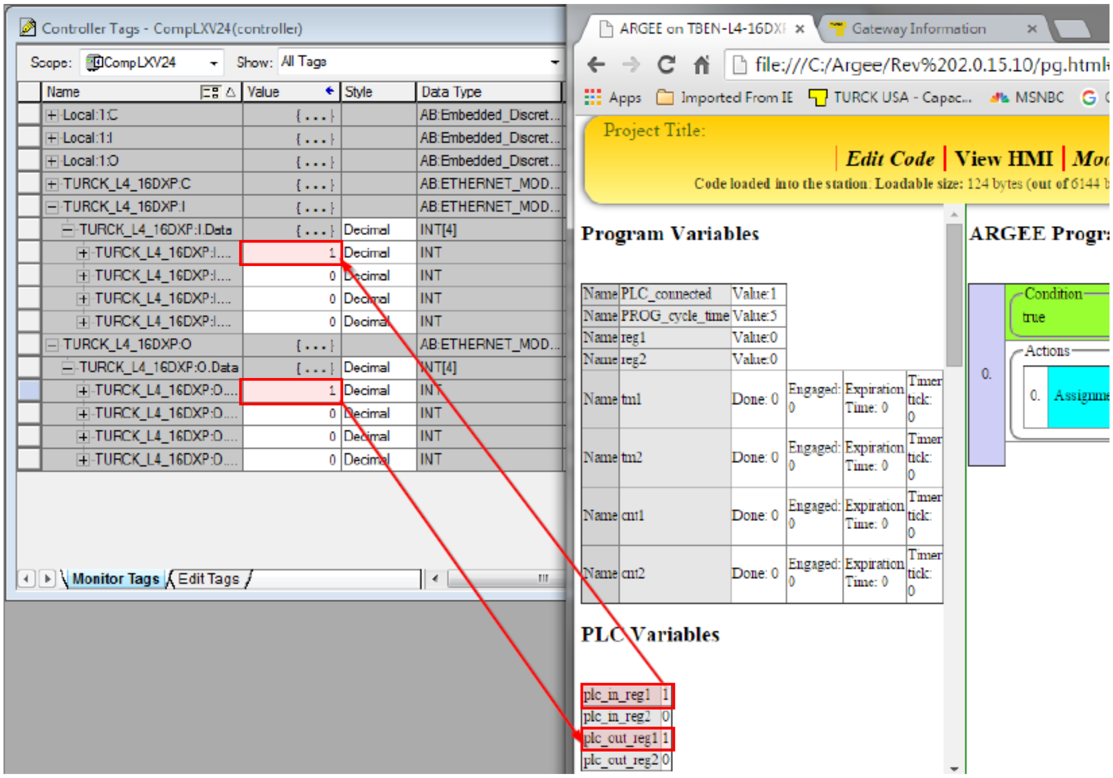
Connection Parameters

	Assembly Instance:	Size:	
Input:	101	4	(16-bit)
Output:	110	4	(16-bit)
Configuration:	1	0	(8-bit)
Status Input:			
Status Output:			

Status: Offline

Explaining the Set up: The user created a Generic Ethernet Device and set the connection points to be 101 & 110**.**

4. The last step is to connect to the device, place a value in the Output Register and verify data transfer.



Controller Tags - CompLXV24(controller)

Scope: CompLXV24 Show: All Tags

Name	Value	Style	Data Type
+ Local:1:C	{...}		AB:Embedded_Discret...
+ Local:1:I	{...}		AB:Embedded_Discret...
+ Local:1:O	{...}		AB:Embedded_Discret...
+ TURCK_L4_16DXP:C	{...}		AB:ETHERNET_MOD...
− TURCK_L4_16DXP:I	{...}		AB:ETHERNET_MOD...
− TURCK_L4_16DXP:I.Data	{...}	Decimal	INT[4]
+ TURCK_L4_16DXP:I....	1	Decimal	INT
+ TURCK_L4_16DXP:I....	0	Decimal	INT
+ TURCK_L4_16DXP:I....	0	Decimal	INT
+ TURCK_L4_16DXP:I....	0	Decimal	INT
− TURCK_L4_16DXP:O	{...}		AB:ETHERNET_MOD...
− TURCK_L4_16DXP:O.Data	{...}	Decimal	INT[4]
+ TURCK_L4_16DXP:O....	1	Decimal	INT
+ TURCK_L4_16DXP:O....	0	Decimal	INT
+ TURCK_L4_16DXP:O....	0	Decimal	INT
+ TURCK_L4_16DXP:O....	0	Decimal	INT

Monitor Tags / Edit Tags

ARGEE on TBEN-L4-16DXP | Gateway Information

file:///C:/Argee/Rev%202.0.15.10/pg.html

Project Title:

Edit Code | View HMI | Mo

Code loaded into the station: Loadable size: 124 bytes (out of 6144 b

Program Variables

Name	PLC_connected	Value:1			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0

PLC Variables

plc_in_reg1	1
plc_in_reg2	0
plc_out_reg1	1
plc_out_reg2	0

ARGEE Progr

Explaining the Example: The user inserted the value “1” into the PLC’s Output register “0”, bit “0”. The data transfer is verified by observing the PLC registers and the ARGEE registers.

14.2 Communicating with a Modbus TCP/IP – Red Lion

ARGEE blocks have the ability to communicate with a Modbus TCP/IP Master. The Modbus Master can establish communication via registers 0x4000 (register 16384 in decimal) and 0x4400 (register 17408 in decimal). 0x4000 is a read only register, while 0x4400 is a read/write register.

NOTE

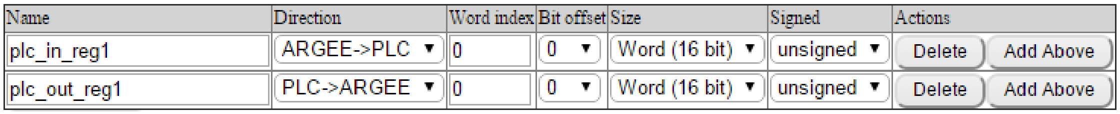
Some Modbus Masters automatically increment the register value by one. For example, register 16384 might be 16385. If the user is having connection issues, the user should try and increment the register value by one.

Example of Communicating with a Modbus TCP/IP Master:

The user wants to check and see if data is being passed back and forth between the ARGEE block and the Modbus Master.

1. The first thing the user does is set up the PLC variables.

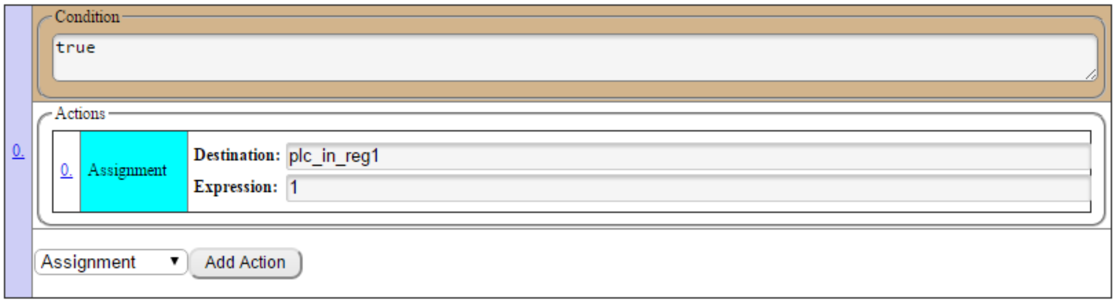
PLC Variables:



Name	Direction	Word index	Bit offset	Size	Signed	Actions
plc_in_reg1	ARGEE->PLC ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above
plc_out_reg1	PLC->ARGEE ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above

Explaining the Setup: The user creates two PLC Variables, then sets the direction the data will travel. Data transmitted from the ARGEE block to Modbus Master is mapped into register 0x4000(hex) and data size as defined in the ARGEE PLC variable section (ARGEE->PLC). Max input data size is 0x80(hex). Data transmitted from the Modbus Master to the ARGEE block is mapped into register 0x4400(hex), and data size as defined in the ARGEE PLC variable section (PLC->ARGEE). Max output data size is 0x80(hex).

2. The next step is to write the ARGEE code.



Condition

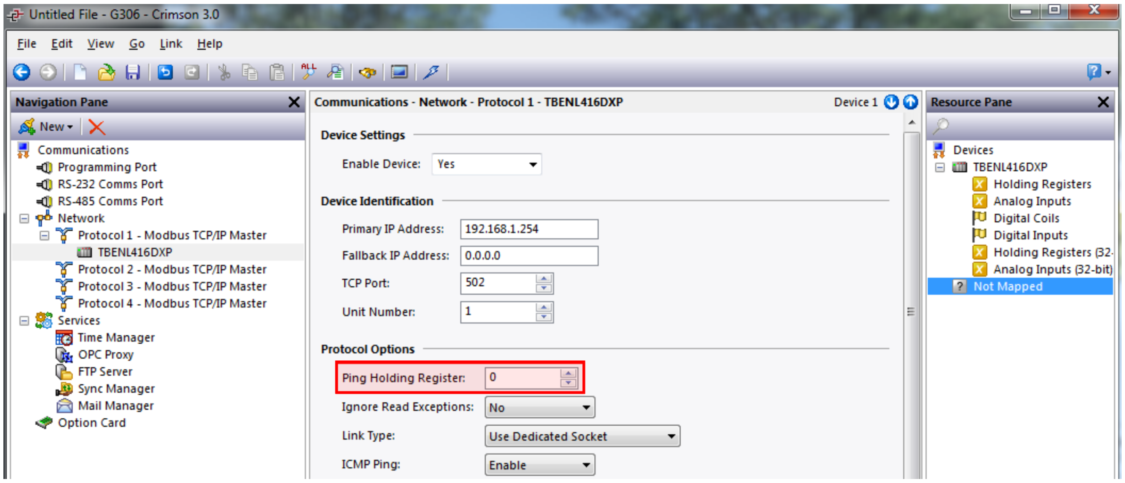
```
true
```

Actions

0. 0. Assignment Destination: plc_in_reg1 Expression: 1

Explaining the Code: The user wrote the value “1” into plc_in_reg1.

3. The third step is to connect a device to the Modbus Master.



Communications - Network - Protocol 1 - TBENL416DXP Device 1

Device Settings

Enable Device: Yes

Device Identification

Primary IP Address: 192.168.1.254

Fallback IP Address: 0.0.0.0

TCP Port: 502

Unit Number: 1

Protocol Options

Ping Holding Register: 0

Ignore Read Exceptions: No

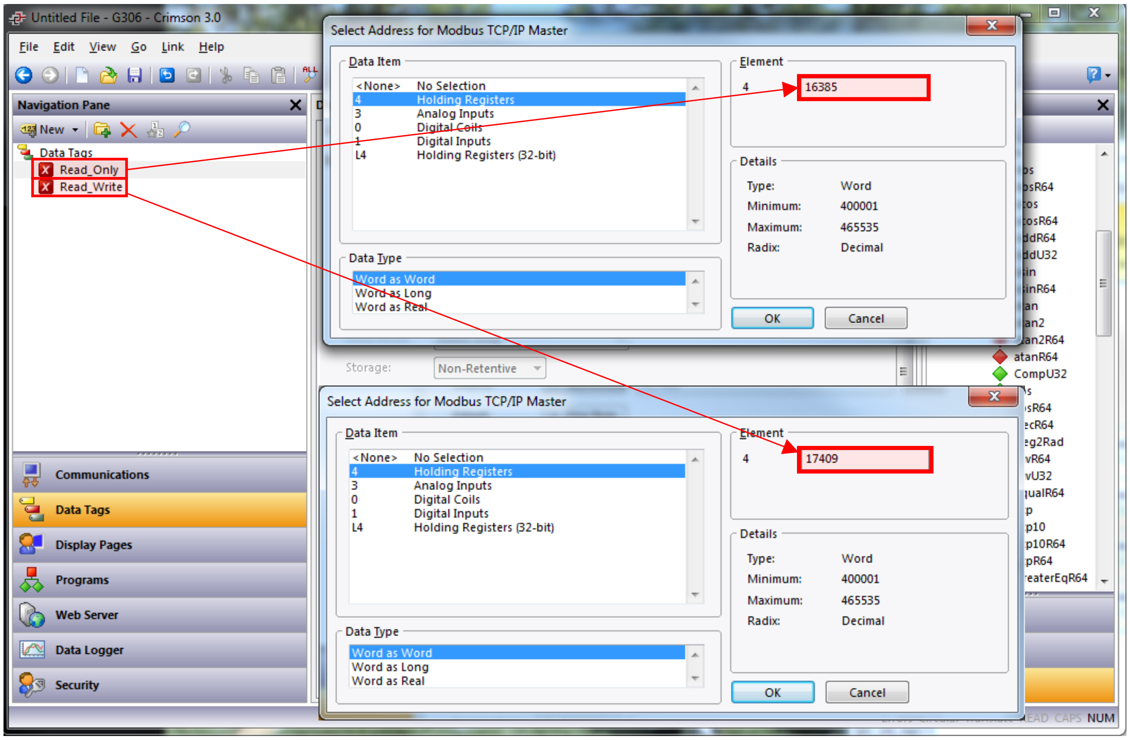
Link Type: Use Dedicated Socket

ICMP Ping: Enable

NOTE

If the user is using a Red Lion HMI, the user needs to set the Ping Holding register to zero.

4. The fourth step is to create tags and assign them to the correct registers.



Select Address for Modbus TCP/IP Master

Data Item

<None>	No Selection
4	Holding Registers
3	Analog Inputs
0	Digital Coils
1	Digital Inputs
L4	Holding Registers (32-bit)

Element: 4 16385

Details

Type: Word
Minimum: 400001
Maximum: 465535
Radix: Decimal

Data Type: Word as Word / Word as Long / Word as Real

Select Address for Modbus TCP/IP Master

Data Item

<None>	No Selection
4	Holding Registers
3	Analog Inputs
0	Digital Coils
1	Digital Inputs
L4	Holding Registers (32-bit)

Element: 4 17409

Details

Type: Word
Minimum: 400001
Maximum: 465535
Radix: Decimal

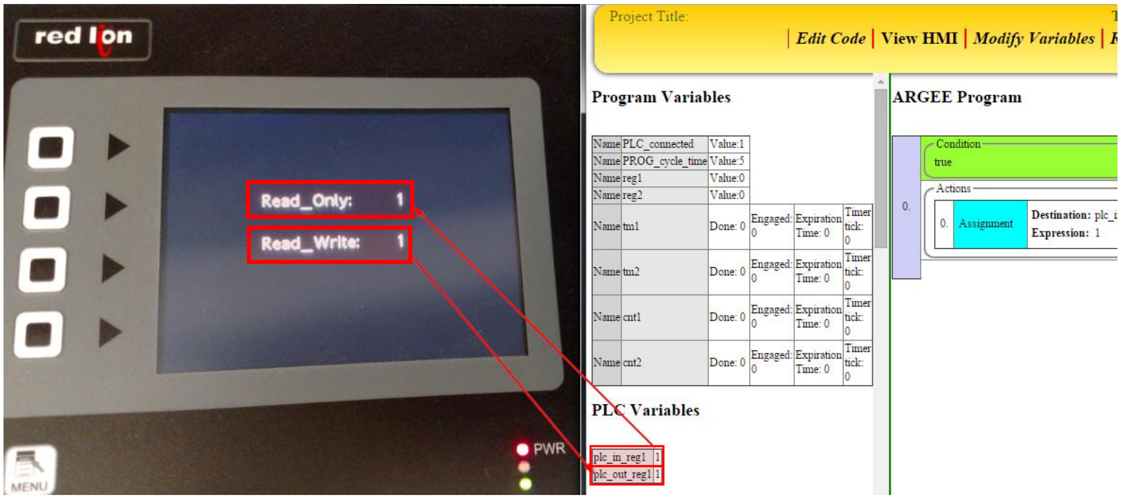
Data Type: Word as Word / Word as Long / Word as Real

NOTE

Red Lion Modbus master register addressing = Original address +1.

Example: Original address = 0x400(hex) = 16384
Red Lion address = 16384 + 1 = 16385

5. The last step is to connect to the device, place a value in the Modbus TCP Master’s Output Register and verify data transfer.



Program Variables

Name	PLC_connected	Value:1			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0

PLC Variables

plc_in_reg1	1
plc_out_reg1	1

ARGEE Program

0.
Condition
true
Actions
0. Assignment Destination: plc_i
Expression: 1

Explaining the Example: The user inserted the value “1” into the HMI’s Output register “17409”. The data transfer is verified by observing the HMI screen and the ARGEE registers.

14.3 Communicating with a PROFINET Master – Siemens

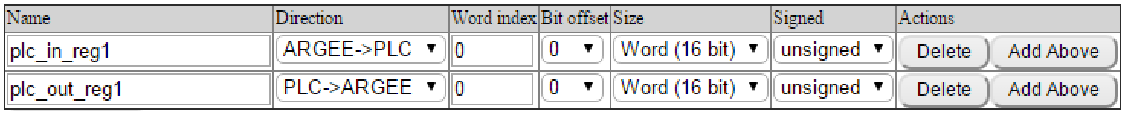
ARGEE blocks have the ability to communicate with a PROFINET Master. The PROFINET Master can establish communication via an ARGEE GSD file.

Example of Communicating with a PROFINET Master:

The user wants to check and see if data is being passed back and forth between the ARGEE block and the PROFINET Master.

1. The first thing the user does is set up the PLC variables.

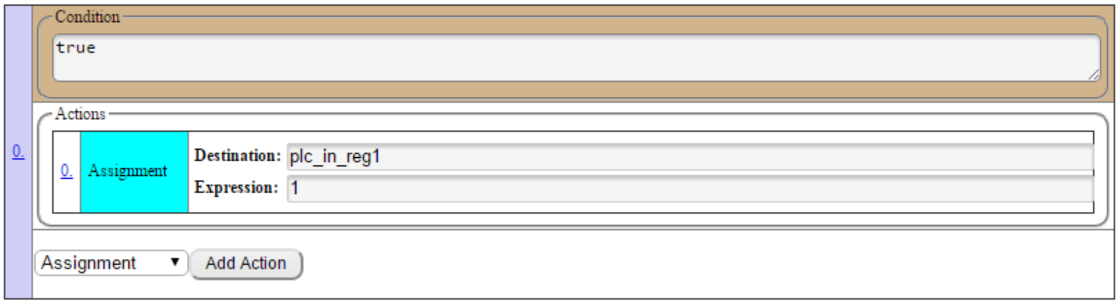
PLC Variables:



Name	Direction	Word index	Bit offset	Size	Signed	Actions
plc_in_reg1	ARGEE->PLC ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above
plc_out_reg1	PLC->ARGEE ▼	0	0 ▼	Word (16 bit) ▼	unsigned ▼	Delete Add Above

Explaining the Set-up: The user creates two PLC Variables. The user sets the direction the data will travel. Data transmitted from ARGEE to the PLC is mapped into the Siemens input address and the data size is defined in the ARGEE PLC variable section (ARGEE->PLC). Data transmitted from the PLC to ARGEE is mapped into the Siemens output address and the data size is defined in the ARGEE PLC variable section (PLC->ARGEE).

2. The next step is to write the ARGEE code.



Condition

```
true
```

Actions

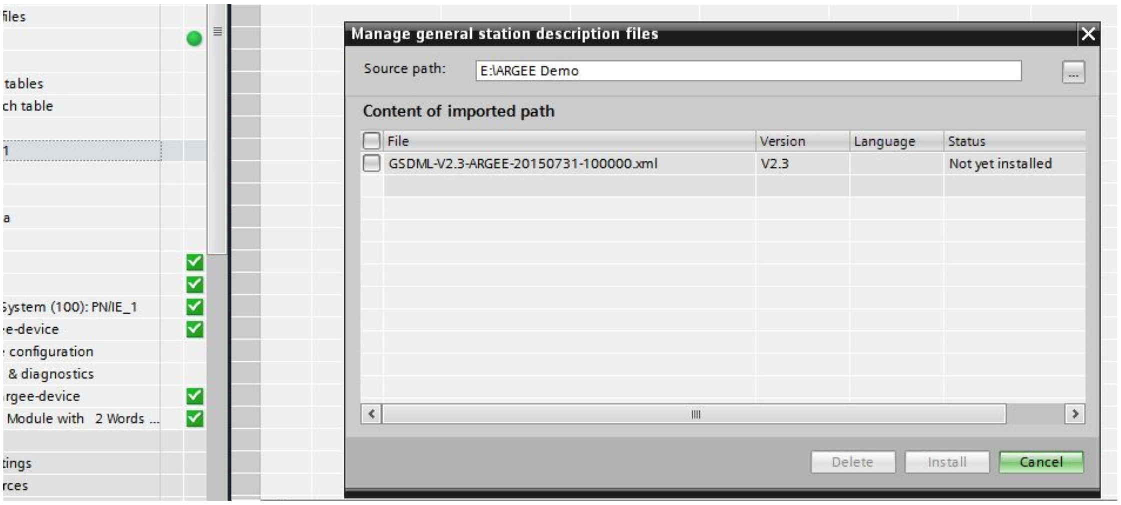
0. Assignment

Destination: plc_in_reg1

Expression: 1

Explaining the Code: The user wrote the value “1” into plc_in_reg1.

3. The third step is to install the ARGEE GSD file.



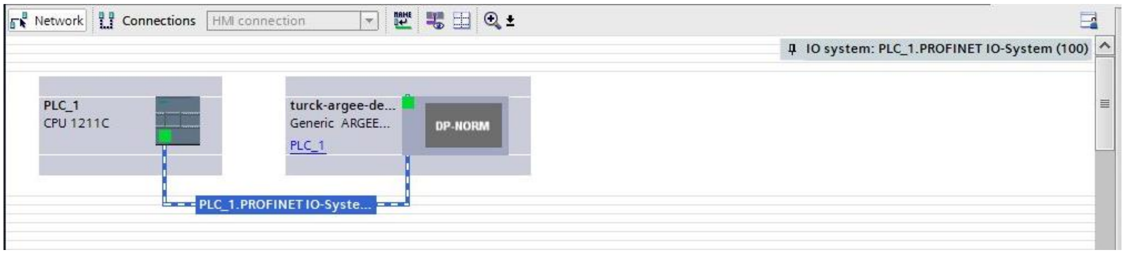
Manage general station description files

Source path: E:\ARGEE Demo

Content of imported path

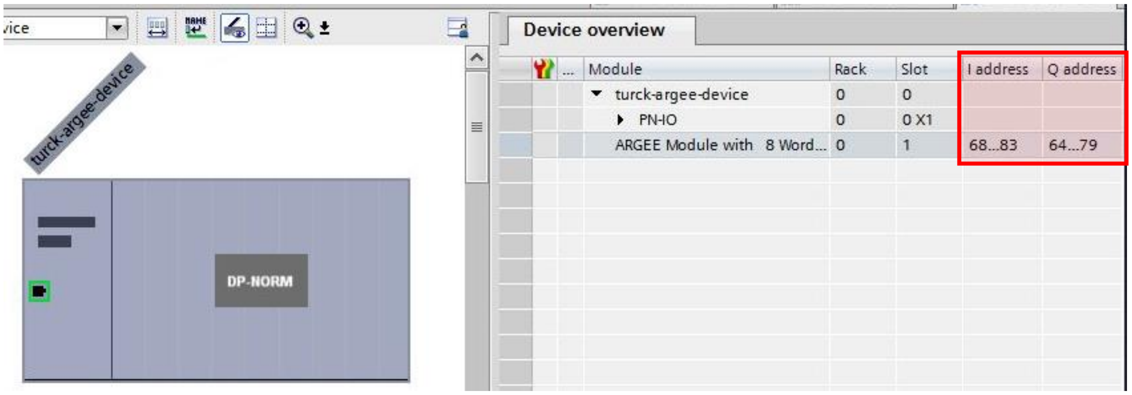
□ File	Version	Language	Status
□ GSDML-V2.3-ARGEE-20150731-100000.xml	V2.3		Not yet installed

4. The fourth step is to add the device to the program.



Explaining the Set up: The user created an ARGEE Device in the devices and networks area.

5. The fifth step is to set up device addresses.

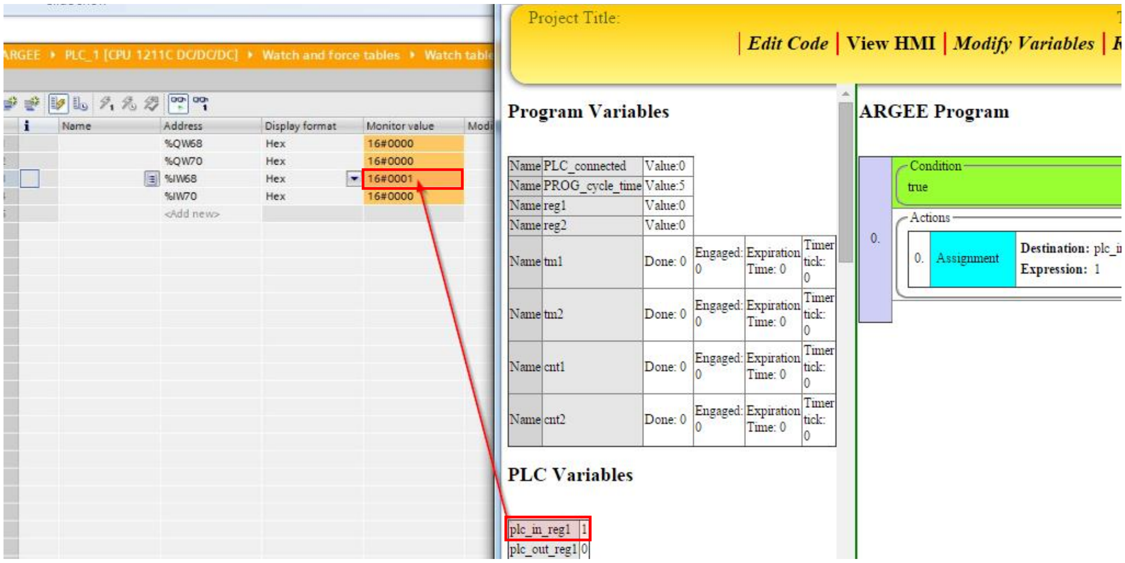


Device overview

...	Module	Rack	Slot	I address	Q address
	turck-argee-device	0	0		
	PN-IO	0	0 X1		
	ARGEE Module with 8 Word...	0	1	68...83	64...79

Explaining the Set up: The user defines the “I address” and “Q address” in the device overview.

The user can now verify the data has been transferred.


RGEE ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Watch and force tables ▸ Watch tabl
Project Title:
Edit Code | View HMI | Modify Variables | R

i	Name	Address	Display format	Monitor value	Modi
		%QW68	Hex	16#0000	
		%QW70	Hex	16#0000	
		%IW68	Hex	16#0001	
		%IW70	Hex	16#0000	
		<Add new>			

Program Variables

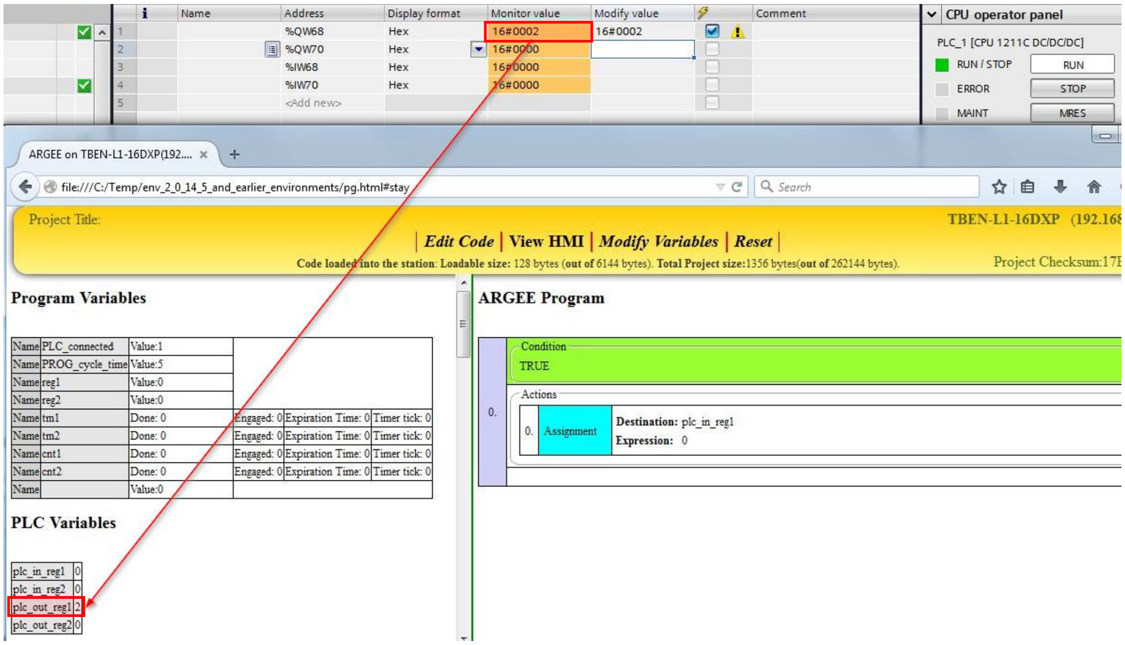
Name	PLC_connected	Value:0			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0

PLC Variables

plc_in_reg1	1
plc_out_reg1	0

ARGEE Program

0.
Condition
true
Actions
0. Assignment
Destination: plc_i
Expression: 1


CPU operator panel
PLC_1 [CPU 1211C DC/DC/DC]
RUN / STOP
RUN
ERROR
STOP
MAINT
MRES
ARGEE on TBEN-L1-16DXP(192....
file:///C:/Temp/env_2_0_14_5_and_earlier_environments/pg.html#stay
Search
Project Title:
TBEN-L1-16DXP (192.168
| Edit Code | View HMI | Modify Variables | Reset |
Project Checksum:17E

	i	Name	Address	Display format	Monitor value	Modify value		Comment
1			%QW68	Hex	16#0002	16#0002	☑	
2			%QW70	Hex	16#0000		□	
3			%IW68	Hex	16#0000		□	
4			%IW70	Hex	16#0000		□	
5			<Add new>				□	

Code loaded into the station: Loadable size: 128 bytes (out of 6144 bytes). Total Project size:1356 bytes(out of 262144 bytes).

Program Variables

Name	PLC_connected	Value:1			
Name	PROG_cycle_time	Value:5			
Name	reg1	Value:0			
Name	reg2	Value:0			
Name	tm1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	tm2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt1	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name	cnt2	Done: 0	Engaged: 0	Expiration Time: 0	Timer tick: 0
Name		Value:0			

PLC Variables

plc_in_reg1	0
plc_in_reg2	0
plc_out_reg1	2
plc_out_reg2	0

ARGEE Program

0.
Condition
TRUE
Actions
0. Assignment
Destination: plc_in_reg1
Expression: 0

Explaining the example: The user inserted the value “2” into the PLC’s Output register and verifies the data transfer.

14.4 Using State Variables

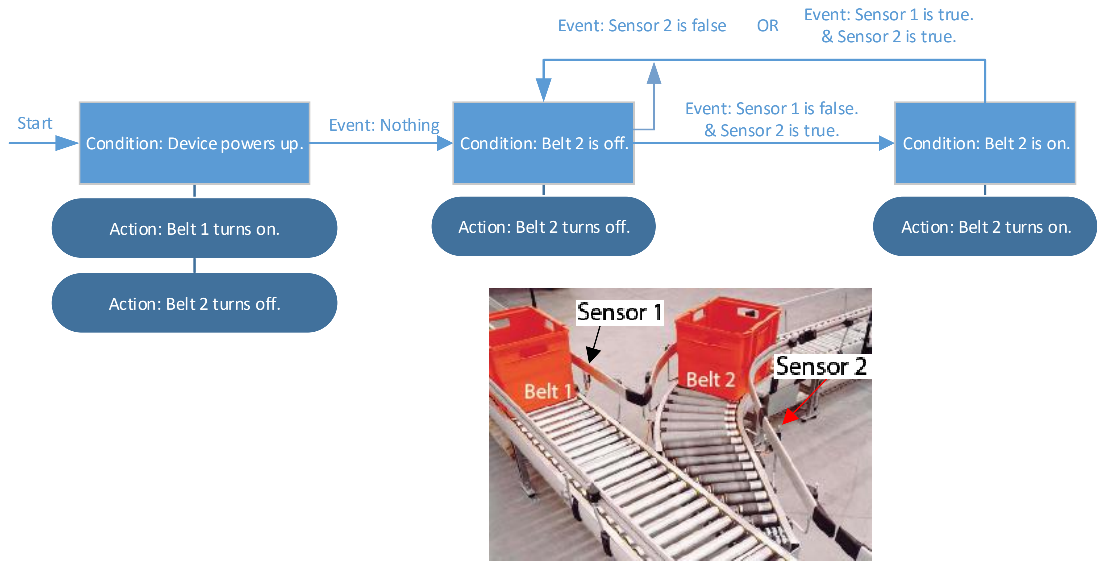
State Variables are helpful in keeping track of the signal as it steps through the code. Before the user creates State Variables, it is a good idea to create a State Machine.

14.4.1 State Machine

A state machine is drawing on a piece of paper that shows how the signal transitions from one state to another.

Example of a State Machine:

The user wants to use their ARGEE block to create a Traffic Cop. A Traffic Cop is a device that merges two conveyer belts together without causing a box collision. The first thing the user does is gets out a piece of paper and draws up a State Machine.


Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Sensor 1
Belt 1
Belt 2
Sensor 2

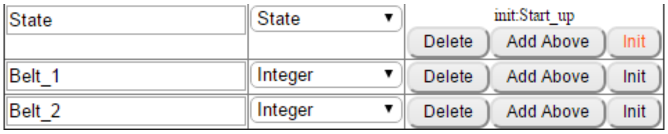
Explaining the State Machine: All the States are in light blue boxes. All the Events occur on the arrows. All Actions are in dark blue ovals.

14.4.2 State Variables

Example of a State Variables:

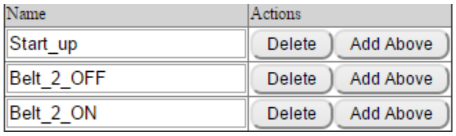
The user is satisfied with the Traffic Cop State Machine. The user now creates Program and State Variables.

Program Variables



State	State ▼	init:Start_up Delete Add Above Init
Belt_1	Integer ▼	Delete Add Above Init
Belt_2	Integer ▼	Delete Add Above Init

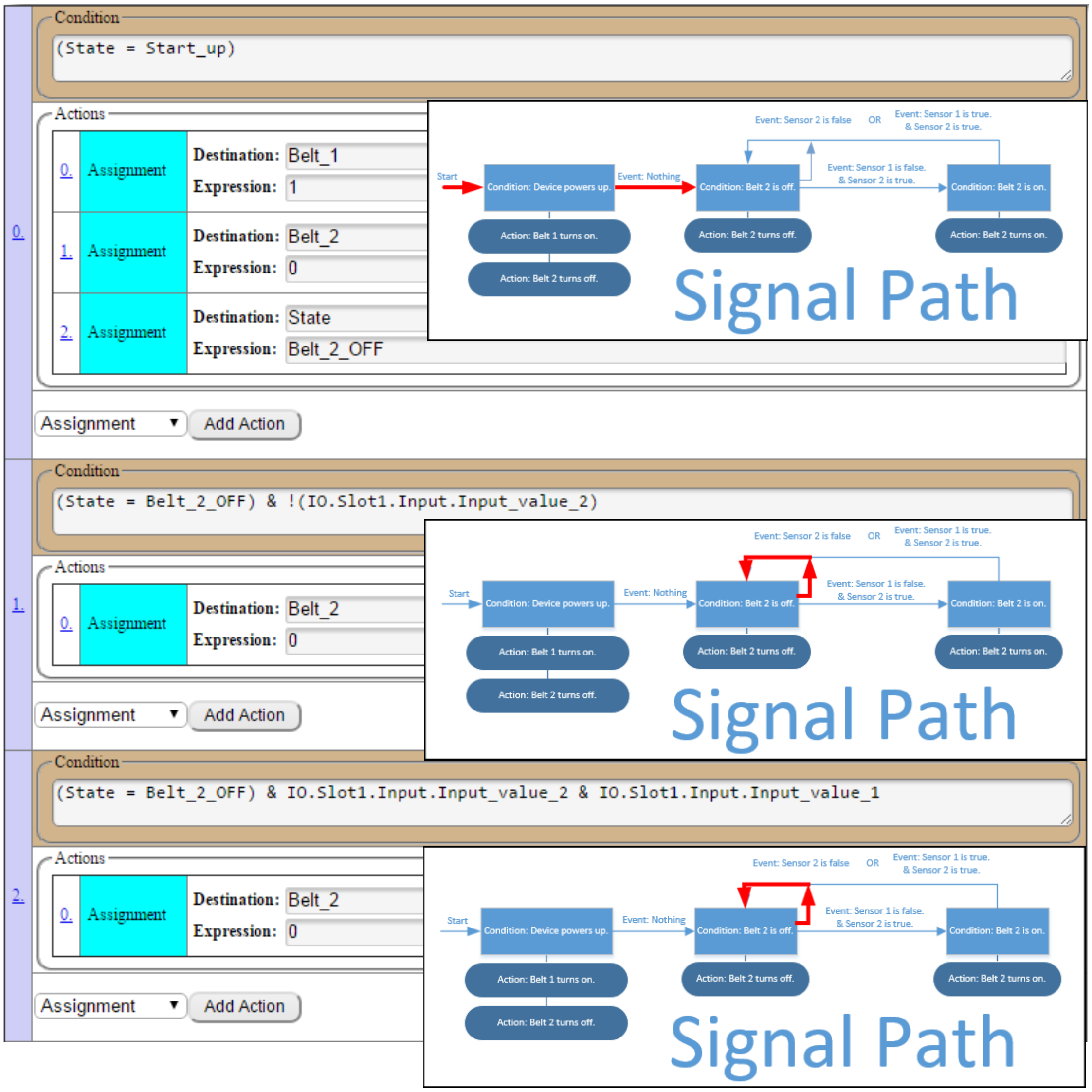
State Variables



Name	Actions
Start_up	Delete Add Above
Belt_2_OFF	Delete Add Above
Belt_2_ON	Delete Add Above

NOTE

Program Variable “State” is initialized to Start-up.



0.

Condition

```
(State = Start_up)
```

Actions

0. Assignment — Destination: Belt_1 — Expression: 1

1. Assignment — Destination: Belt_2 — Expression: 0

2. Assignment — Destination: State — Expression: Belt_2_OFF

Assignment ▼
Add Action
Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Action: Belt 2 turns off.
Signal Path

1.

Condition

```
(State = Belt_2_OFF) & !(IO.Slot1.Input.Input_value_2)
```

Actions

0. Assignment — Destination: Belt_2 — Expression: 0

Assignment ▼
Add Action
Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Action: Belt 2 turns off.
Signal Path

2.

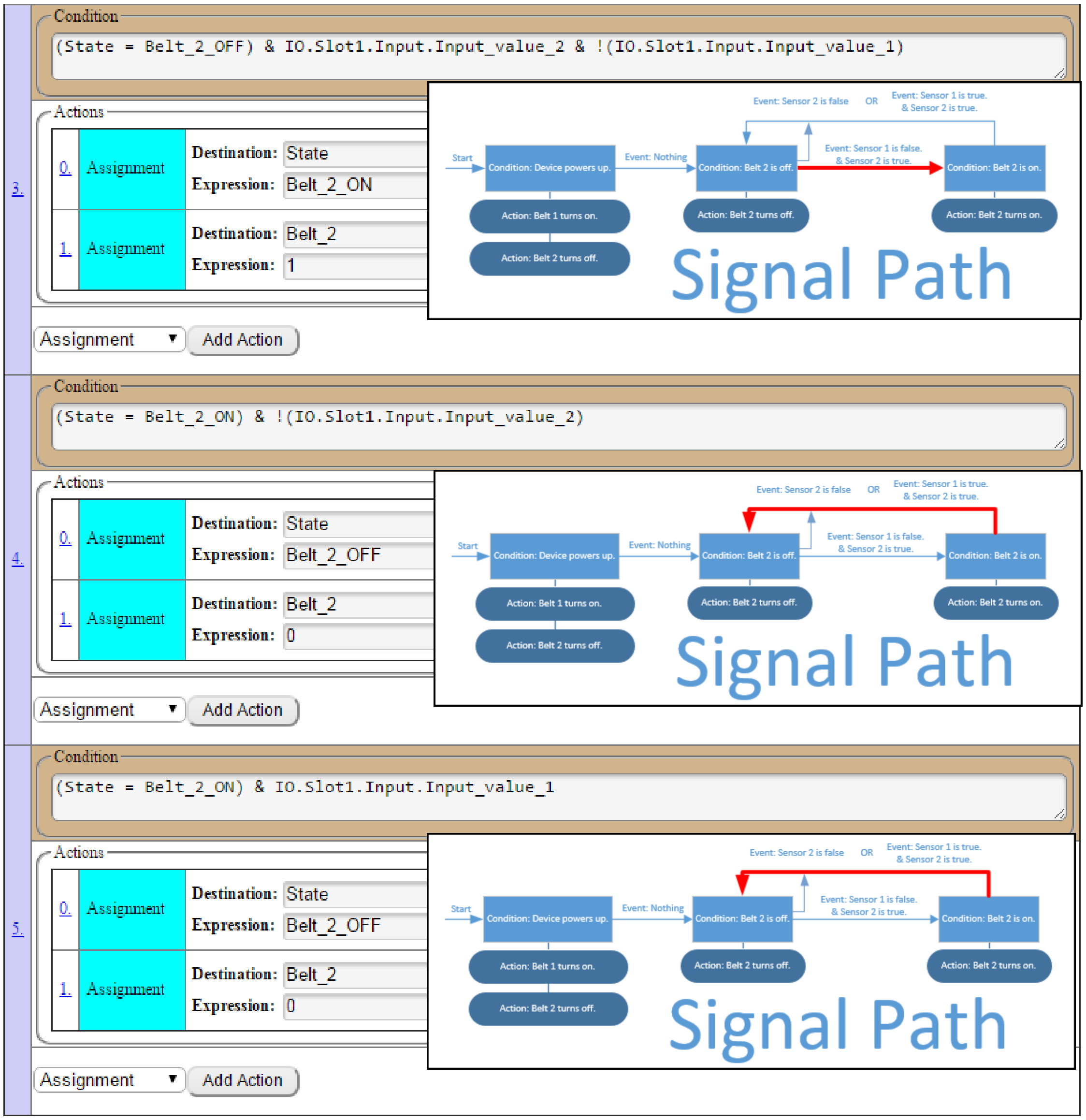
Condition

```
(State = Belt_2_OFF) & IO.Slot1.Input.Input_value_2 & IO.Slot1.Input.Input_value_1
```

Actions

0. Assignment — Destination: Belt_2 — Expression: 0

Assignment ▼
Add Action
Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Action: Belt 2 turns off.
Signal Path



3.

Condition

```
(State = Belt_2_OFF) & IO.Slot1.Input.Input_value_2 & !(IO.Slot1.Input.Input_value_1)
```

Actions

0.	Assignment	Destination: State Expression: Belt_2_ON
1.	Assignment	Destination: Belt_2 Expression: 1

Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Signal Path
Assignment ▾
Add Action

4.

Condition

```
(State = Belt_2_ON) & !(IO.Slot1.Input.Input_value_2)
```

Actions

0.	Assignment	Destination: State Expression: Belt_2_OFF
1.	Assignment	Destination: Belt_2 Expression: 0

Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Signal Path
Assignment ▾
Add Action

5.

Condition

```
(State = Belt_2_ON) & IO.Slot1.Input.Input_value_1
```

Actions

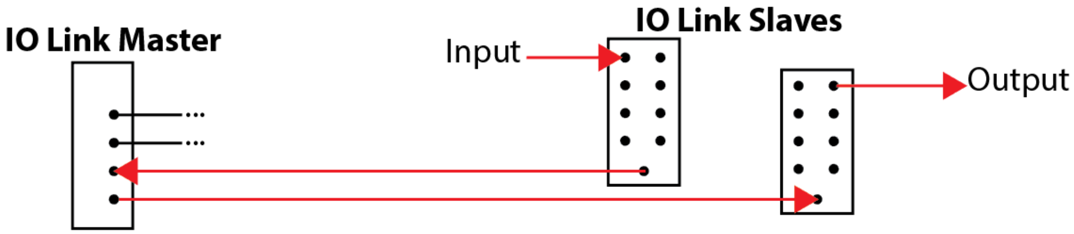
0.	Assignment	Destination: State Expression: Belt_2_OFF
1.	Assignment	Destination: Belt_2 Expression: 0

Event: Sensor 2 is false
OR
Event: Sensor 1 is true. & Sensor 2 is true.
Start
Condition: Device powers up.
Event: Nothing
Condition: Belt 2 is off.
Event: Sensor 1 is false. & Sensor 2 is true.
Condition: Belt 2 is on.
Action: Belt 1 turns on.
Action: Belt 2 turns off.
Action: Belt 2 turns off.
Action: Belt 2 turns on.
Signal Path
Assignment ▾
Add Action

Explaining the Example: When the device is powered up, Belt 1 is turned on and Belt 2 is turned off. If Sensor 2 goes true (or a box shows up on Belt 2), ARGEE will check and see if Sensor 1 is true (or if a box is on Belt 1). If Sensor 1 is true then Belt 2 stays off. If Sensor 1 is false, Belt 2 turns on and clears the box on Belt 2.

14.5 Working with IO-Link

When a user combines IO-Link technology with AGREE, the application solutions that can be created become endless. IO-Link can support digital and analog signals. Because there are so many IO-Link configurations, it is recommended that the user read the *Turck IO-Link master manual* before attempting any IO-Link applications.



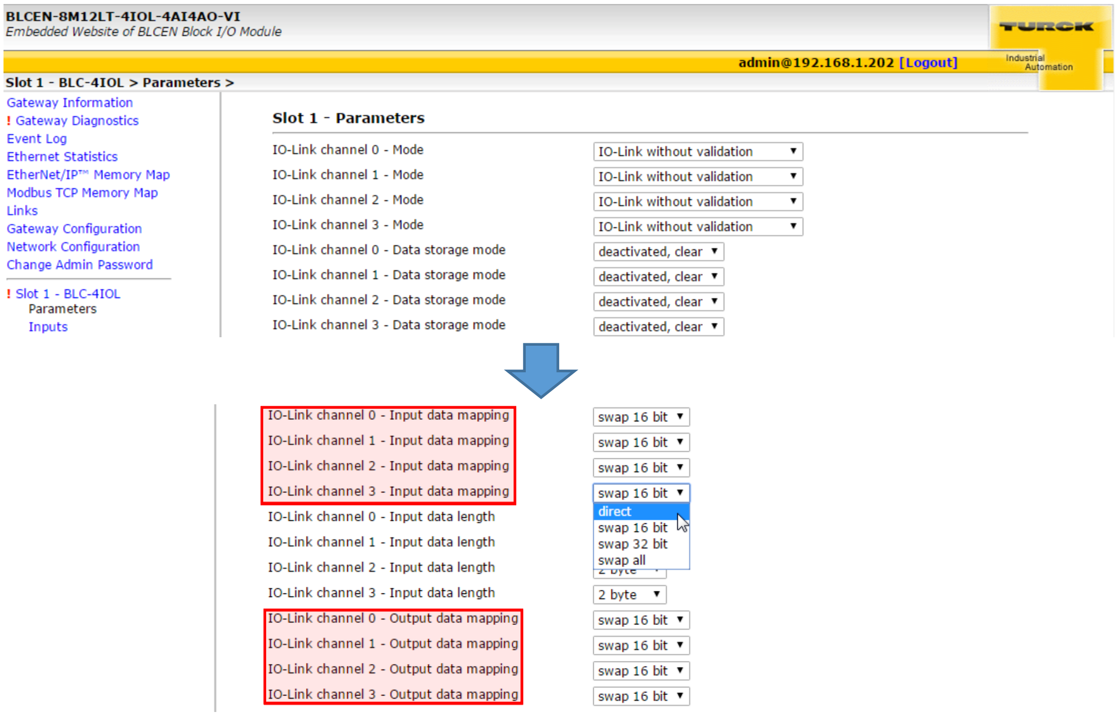
NOTE

Depending on the fieldbus used, it may be necessary to swap process data. The process data mapping can be changed through the webserver parameters. “Input data mapping”, “Output data mapping”, “Input data length”, and “Output data length”. More information can be found in the Turck IO-Link master manual chapter 4, page 4-4.

Example of IO-Link:

The user wants to use a digital input on an IO-Link slave to turn on a digital output on a different IO-Link slave using EtherNet/IP.

- ➢ The first thing the user has to do is change the Input and Output Data Mapping parameter from “swap 16 bit” to “direct”.



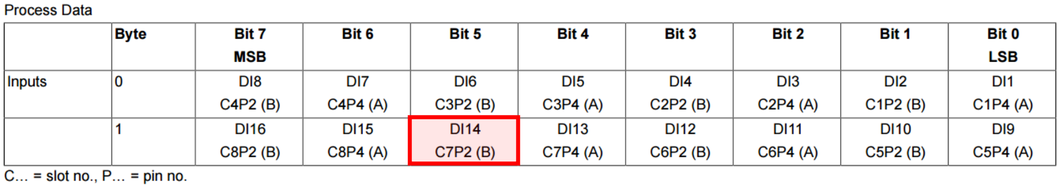
Slot 1 - Parameters

IO-Link channel 0 - Mode	IO-Link without validation
IO-Link channel 1 - Mode	IO-Link without validation
IO-Link channel 2 - Mode	IO-Link without validation
IO-Link channel 3 - Mode	IO-Link without validation
IO-Link channel 0 - Data storage mode	deactivated, clear
IO-Link channel 1 - Data storage mode	deactivated, clear
IO-Link channel 2 - Data storage mode	deactivated, clear
IO-Link channel 3 - Data storage mode	deactivated, clear
IO-Link channel 0 - Input data mapping	swap 16 bit
IO-Link channel 1 - Input data mapping	swap 16 bit
IO-Link channel 2 - Input data mapping	swap 16 bit
IO-Link channel 3 - Input data mapping	swap 16 bit direct swap 16 bit swap 32 bit swap all
IO-Link channel 0 - Input data length	
IO-Link channel 1 - Input data length	
IO-Link channel 2 - Input data length	2 byte
IO-Link channel 3 - Input data length	2 byte
IO-Link channel 0 - Output data mapping	swap 16 bit
IO-Link channel 1 - Output data mapping	swap 16 bit
IO-Link channel 2 - Output data mapping	swap 16 bit
IO-Link channel 3 - Output data mapping	swap 16 bit

Explaining the Example: The user logged into the webserver and changed the Input and Output Data Mapping parameter from “swap 16 bit” to “direct”, and then clicked submit.

- The next thing the user does is look at the data map of the two IO-Link slaves and determine which Input and Output to link together.

Slave 1 Data Map (A TBIL-M1-16DIP Connected to Port 0)

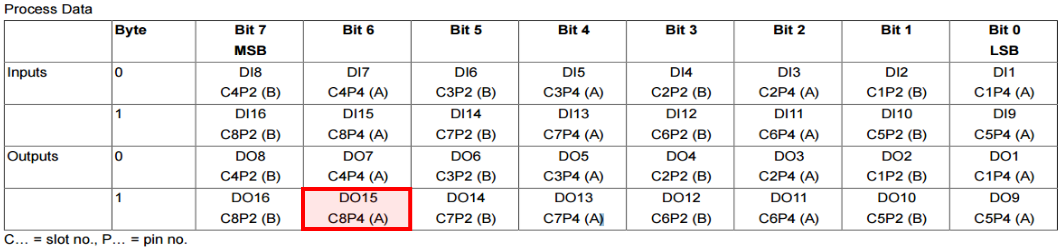


Process Data

	Byte	Bit 7 MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 LSB
Inputs	0	DI8 C4P2 (B)	DI7 C4P4 (A)	DI6 C3P2 (B)	DI5 C3P4 (A)	DI4 C2P2 (B)	DI3 C2P4 (A)	DI2 C1P2 (B)	DI1 C1P4 (A)
	1	DI16 C8P2 (B)	DI15 C8P4 (A)	DI14 C7P2 (B)	DI13 C7P4 (A)	DI12 C6P2 (B)	DI11 C6P4 (A)	DI10 C5P2 (B)	DI9 C5P4 (A)

C… = slot no., P… = pin no.

Slave 2 Data Map (A TBIL-M1-16DXP Connected to Port 1)



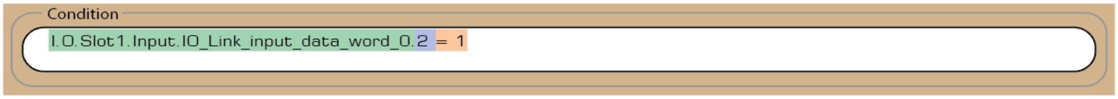
Process Data

	Byte	Bit 7 MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 LSB
Inputs	0	DI8 C4P2 (B)	DI7 C4P4 (A)	DI6 C3P2 (B)	DI5 C3P4 (A)	DI4 C2P2 (B)	DI3 C2P4 (A)	DI2 C1P2 (B)	DI1 C1P4 (A)
	1	DI16 C8P2 (B)	DI15 C8P4 (A)	DI14 C7P2 (B)	DI13 C7P4 (A)	DI12 C6P2 (B)	DI11 C6P4 (A)	DI10 C5P2 (B)	DI9 C5P4 (A)
Outputs	0	DO8 C4P2 (B)	DO7 C4P4 (A)	DO6 C3P2 (B)	DO5 C3P4 (A)	DO4 C2P2 (B)	DO3 C2P4 (A)	DO2 C1P2 (B)	DO1 C1P4 (A)
	1	DO16 C8P2 (B)	DO15 C8P4 (A)	DO14 C7P2 (B)	DO13 C7P4 (A)	DO12 C6P2 (B)	DO11 C6P4 (A)	DO10 C5P2 (B)	DO9 C5P4 (A)

C… = slot no., P… = pin no.

Explaining the user’s decision: The user wants Connector 7 Pin 2 (Port B) on the input block to turn on Connector 8 Pin 4 (Port A) on the output block.

IO-Link Command Structure:



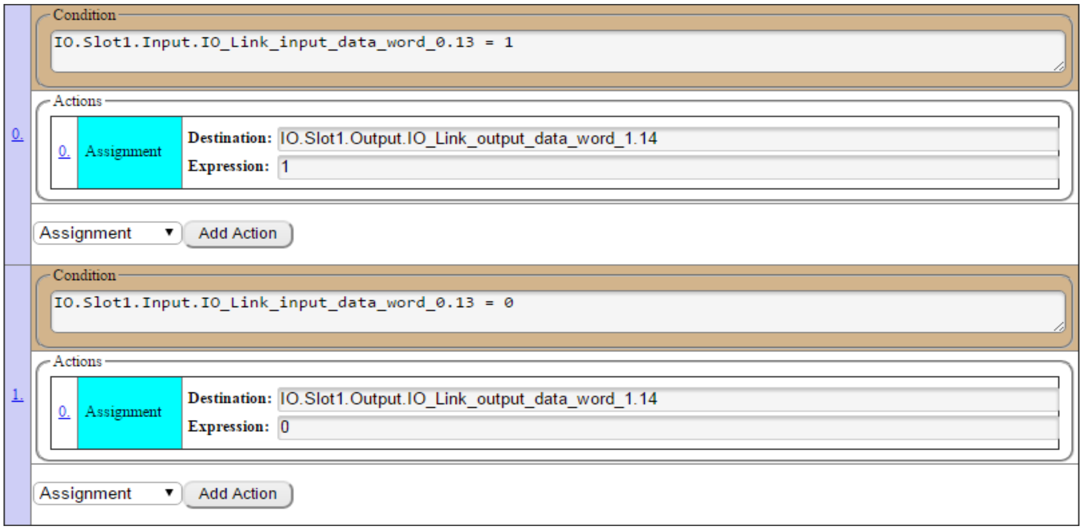
Condition

I.O.Slot1.Input.IO_Link_input_data_word_0.2 = 1

🟩 = *The I/O location and data word that is being targeteted.*

🟦 = *The specific bit in the data word that is being targeted.*

🟧 = *The value that is being loaded into that bit location*



0.

Condition

```
IO.Slot1.Input.IO_Link_input_data_word_0.13 = 1
```

Actions

0. Assignment

Destination: IO.Slot1.Output.IO_Link_output_data_word_1.14

Expression: 1

1.

Condition

```
IO.Slot1.Input.IO_Link_input_data_word_0.13 = 0
```

Actions

0. Assignment

Destination: IO.Slot1.Output.IO_Link_output_data_word_1.14

Expression: 0

Explaining the example: When Connector 7 Port B on the input slave block is true, Connector 8 Port A on the output slave block is true. When Connector 7 Port B on the input slave block is false, Connector 8 Port A on the output slave block is false.

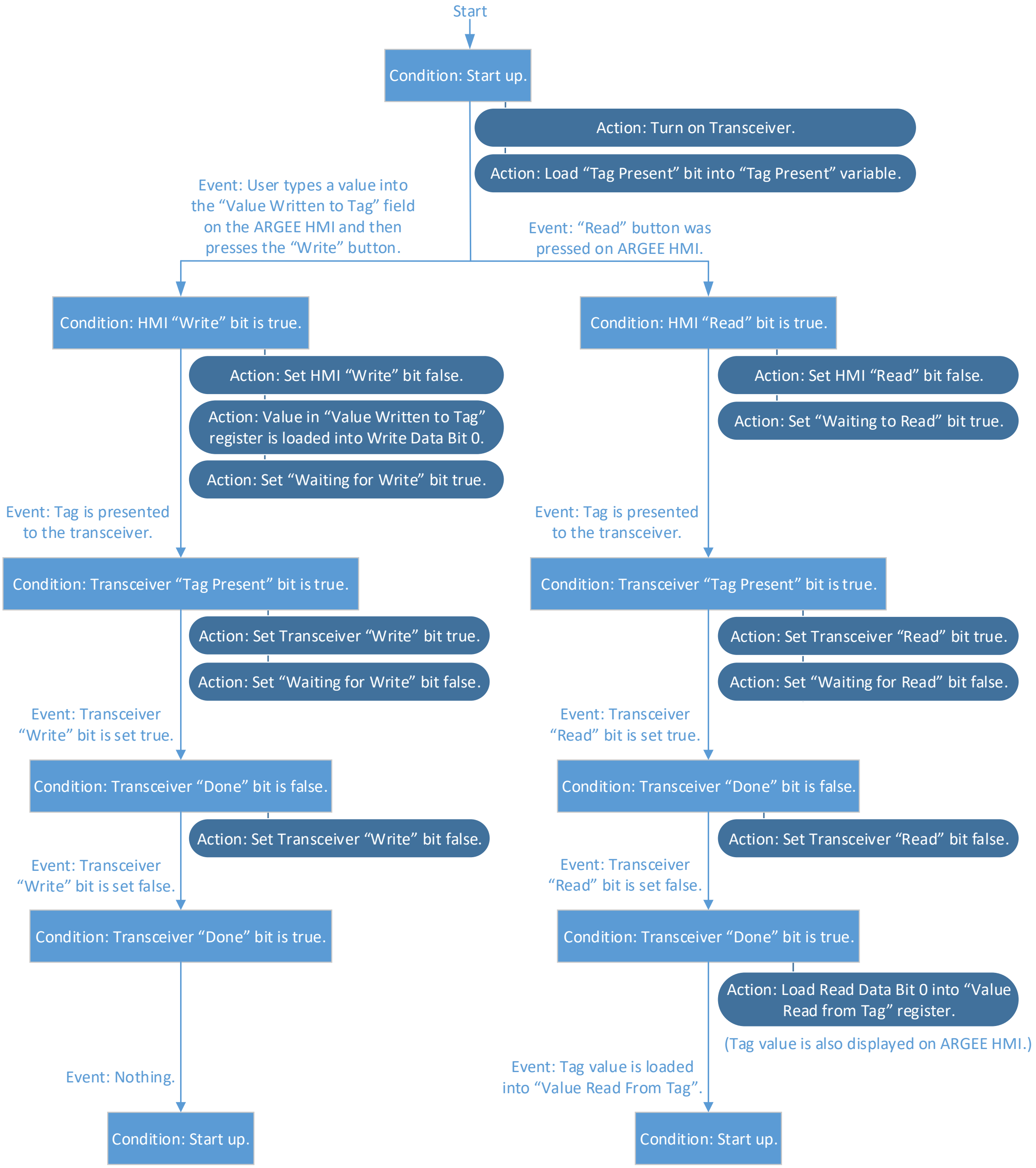
14.6 Working with RFID

If a user needs to solve a simple tracking application, using RFID technology (powered by ARGEE) might be the solution. Many factors influence RFID Read/Write applications. The user can reference the RFID user manual for more information about RFID.

Example of RFID:

The user wants to create an ARGEE HMI that can read from and write to RFID tags.

➢ The first thing the user must do is to break out a pen and paper and draw up a state machine.

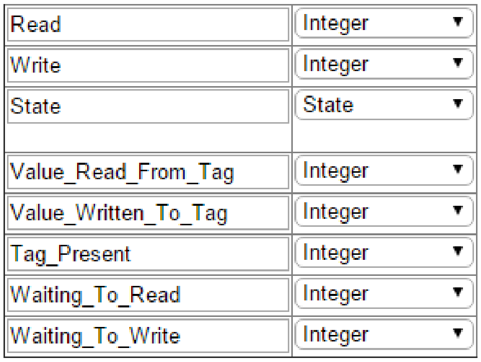


Start
Condition: Start up.
Action: Turn on Transceiver.
Action: Load “Tag Present” bit into “Tag Present” variable.
Event: User types a value into the “Value Written to Tag” field on the ARGEE HMI and then presses the “Write” button.
Event: “Read” button was pressed on ARGEE HMI.
Condition: HMI “Write” bit is true.
Action: Set HMI “Write” bit false.
Action: Value in “Value Written to Tag” register is loaded into Write Data Bit 0.
Action: Set “Waiting for Write” bit true.
Event: Tag is presented to the transceiver.
Condition: Transceiver “Tag Present” bit is true.
Action: Set Transceiver “Write” bit true.
Action: Set “Waiting for Write” bit false.
Event: Transceiver “Write” bit is set true.
Condition: Transceiver “Done” bit is false.
Action: Set Transceiver “Write” bit false.
Event: Transceiver “Write” bit is set false.
Condition: Transceiver “Done” bit is true.
Event: Nothing.
Condition: Start up.
Condition: HMI “Read” bit is true.
Action: Set HMI “Read” bit false.
Action: Set “Waiting to Read” bit true.
Event: Tag is presented to the transceiver.
Condition: Transceiver “Tag Present” bit is true.
Action: Set Transceiver “Read” bit true.
Action: Set “Waiting for Read” bit false.
Event: Transceiver “Read” bit is set true.
Condition: Transceiver “Done” bit is false.
Action: Set Transceiver “Read” bit false.
Event: Transceiver “Read” bit is set false.
Condition: Transceiver “Done” bit is true.
Action: Load Read Data Bit 0 into “Value Read from Tag” register.
(Tag value is also displayed on ARGEE HMI.)
Event: Tag value is loaded into “Value Read From Tag”.
Condition: Start up.

Explaining the RFID State Machine: When the device powers up, the transceiver gets turned on. From the ARGEE HMI screen the user can select two paths: *Read from Tag*, or *Write to Tag* path. If the user wants to read a tag value, simply click the *Read* button on the HMI and present a tag to the transceiver. If the user wants to write to a tag, just type a value into the HMI, click *Write* on the HMI, and then present a tag to the transceiver. After a Read/Write has occurred, the program goes back to the "Startup" state and waits for the next command.

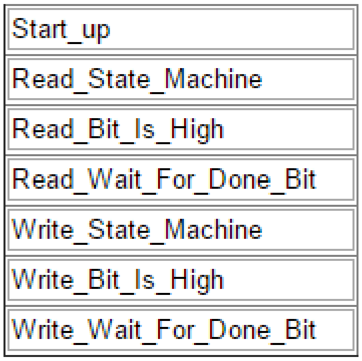
Writing the Code:

Program Variables



Read	Integer
Write	Integer
State	State
Value_Read_From_Tag	Integer
Value_Written_To_Tag	Integer
Tag_Present	Integer
Waiting_To_Read	Integer
Waiting_To_Write	Integer

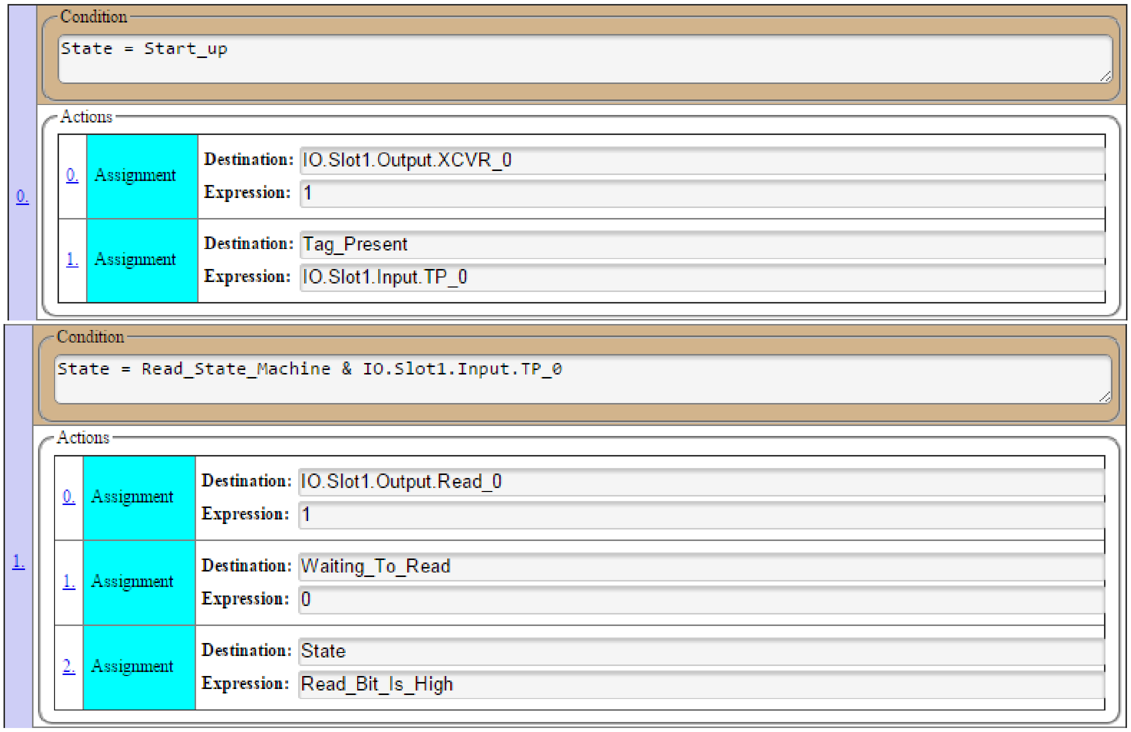
State Variables



Start_up
Read_State_Machine
Read_Bit_Is_High
Read_Wait_For_Done_Bit
Write_State_Machine
Write_Bit_Is_High
Write_Wait_For_Done_Bit

NOTE

Set initial state for Program Variable "State" to "Start_up".



0.

Condition

```
State = Start_up
```

Actions

0.	Assignment	Destination: IO.Slot1.Output.XCVR_0 Expression: 1
1.	Assignment	Destination: Tag_Present Expression: IO.Slot1.Input.TP_0

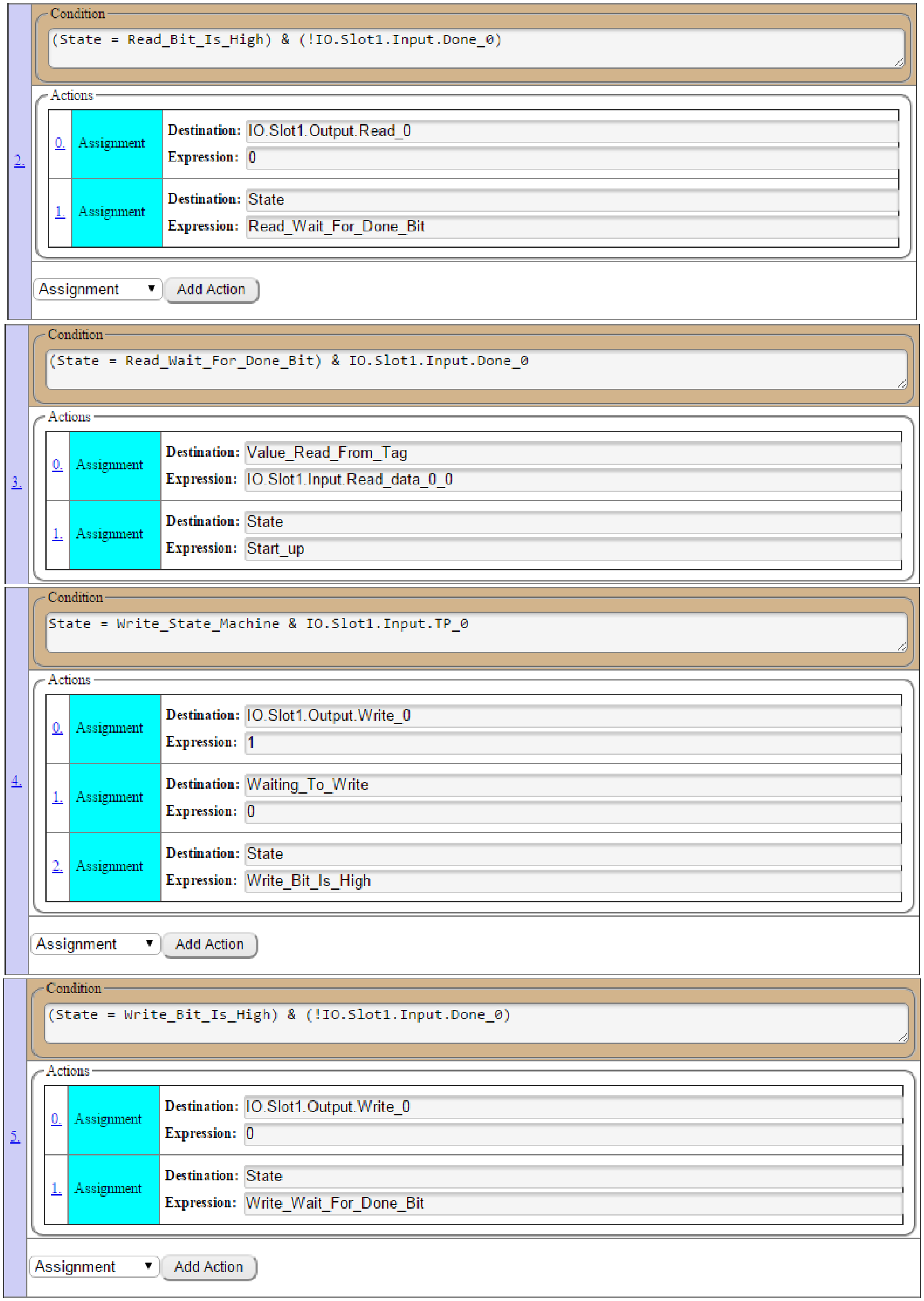
1.

Condition

```
State = Read_State_Machine & IO.Slot1.Input.TP_0
```

Actions

0.	Assignment	Destination: IO.Slot1.Output.Read_0 Expression: 1
1.	Assignment	Destination: Waiting_To_Read Expression: 0
2.	Assignment	Destination: State Expression: Read_Bit_Is_High



2.

Condition

```
(State = Read_Bit_Is_High) & (!IO.Slot1.Input.Done_0)
```

Actions

0.	Assignment	**Destination:** **Expression:**	IO.Slot1.Output.Read_0 0
1.	Assignment	**Destination:** **Expression:**	State Read_Wait_For_Done_Bit

Assignment ▼ | Add Action

3.

Condition

```
(State = Read_Wait_For_Done_Bit) & IO.Slot1.Input.Done_0
```

Actions

0.	Assignment	**Destination:** **Expression:**	Value_Read_From_Tag IO.Slot1.Input.Read_data_0_0
1.	Assignment	**Destination:** **Expression:**	State Start_up

4.

Condition

```
State = Write_State_Machine & IO.Slot1.Input.TP_0
```

Actions

0.	Assignment	**Destination:** **Expression:**	IO.Slot1.Output.Write_0 1
1.	Assignment	**Destination:** **Expression:**	Waiting_To_Write 0
2.	Assignment	**Destination:** **Expression:**	State Write_Bit_Is_High

Assignment ▼ | Add Action

5.

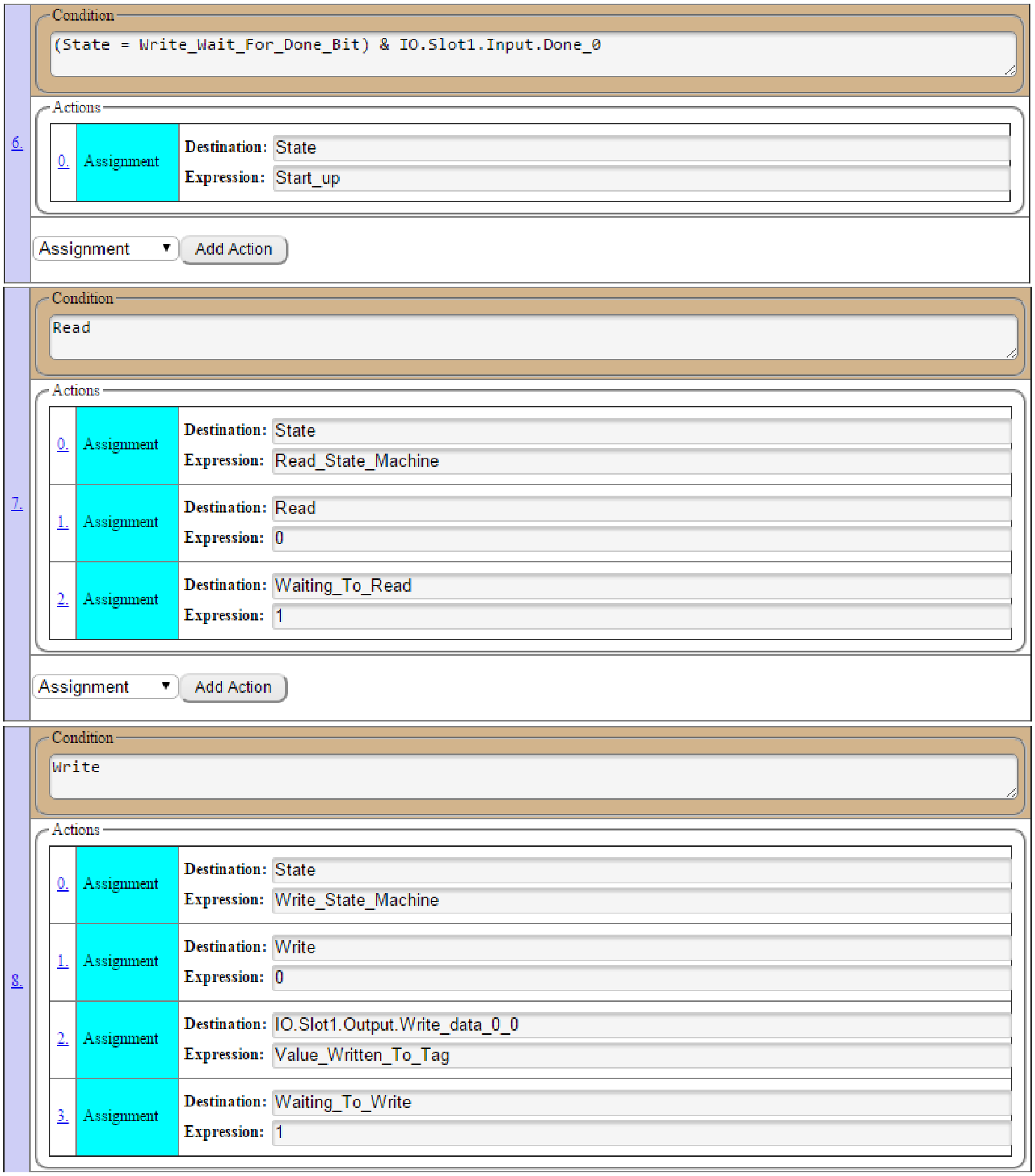
Condition

```
(State = Write_Bit_Is_High) & (!IO.Slot1.Input.Done_0)
```

Actions

0.	Assignment	**Destination:** **Expression:**	IO.Slot1.Output.Write_0 0
1.	Assignment	**Destination:** **Expression:**	State Write_Wait_For_Done_Bit

Assignment ▼ | Add Action



6.

Condition

```
(State = Write_Wait_For_Done_Bit) & IO.Slot1.Input.Done_0
```

Actions

0.	Assignment	Destination:	State
		Expression:	Start_up

7.

Condition

```
Read
```

Actions

0.	Assignment	Destination:	State
		Expression:	Read_State_Machine
1.	Assignment	Destination:	Read
		Expression:	0
2.	Assignment	Destination:	Waiting_To_Read
		Expression:	1

8.

Condition

```
Write
```

Actions

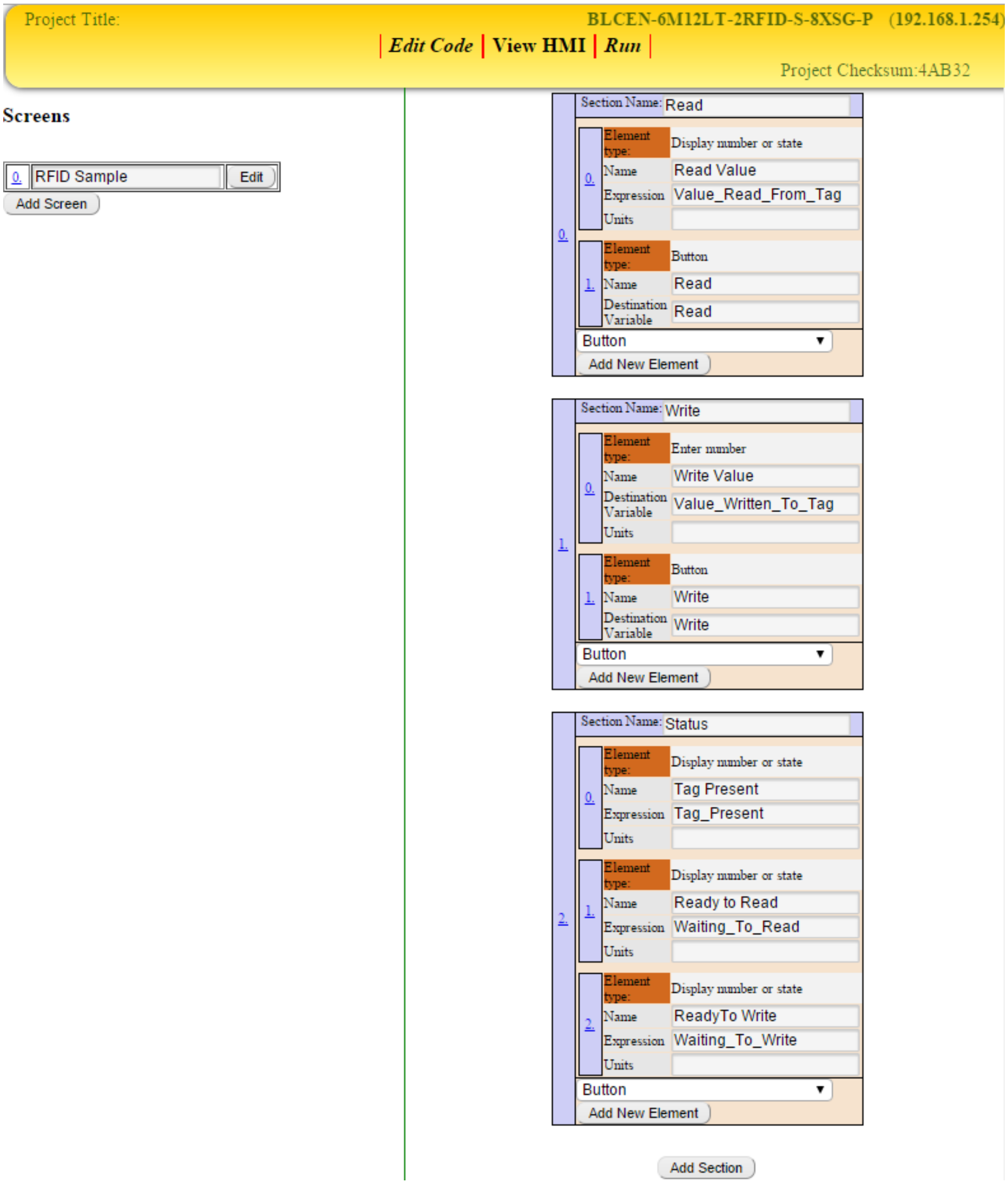
0.	Assignment	Destination:	State
		Expression:	Write_State_Machine
1.	Assignment	Destination:	Write
		Expression:	0
2.	Assignment	Destination:	IO.Slot1.Output.Write_data_0_0
		Expression:	Value_Written_To_Tag
3.	Assignment	Destination:	Waiting_To_Write
		Expression:	1

Explaining the Code: The user created the “Read” and “Write” program variables so the ARGEE HMI buttons would have a way to start the RFID operation. The “Tag Present”, “Waiting To Read” and “Waiting To Write” program variables are status bits that can be displayed on the HMI. Everything else is identical to the state machine explanation.

The RFID example continues on the next page.

Building the RFID HMI:

➢ After the code is written, the user clicks *Edit HMI* from the ARGEE menu bar.



Project Title: BLCEN-6M12LT-2RFID-S-8XSG-P (192.168.1.254)

| *Edit Code* | **View HMI** | *Run* |

Project Checksum:4AB32

Screens

0. RFID Sample Edit

Add Screen

0. Section Name: Read

- 0. Element type: Display number or state
  - Name: Read Value
  - Expression: Value_Read_From_Tag
  - Units:
- 1. Element type: Button
  - Name: Read
  - Destination Variable: Read

Button ▼

Add New Element

1. Section Name: Write

- 0. Element type: Enter number
  - Name: Write Value
  - Destination Variable: Value_Written_To_Tag
  - Units:
- 1. Element type: Button
  - Name: Write
  - Destination Variable: Write

Button ▼

Add New Element

2. Section Name: Status

- 0. Element type: Display number or state
  - Name: Tag Present
  - Expression: Tag_Present
  - Units:
- 1. Element type: Display number or state
  - Name: Ready to Read
  - Expression: Waiting_To_Read
  - Units:
- 2. Element type: Display number or state
  - Name: ReadyTo Write
  - Expression: Waiting_To_Write
  - Units:

Button ▼

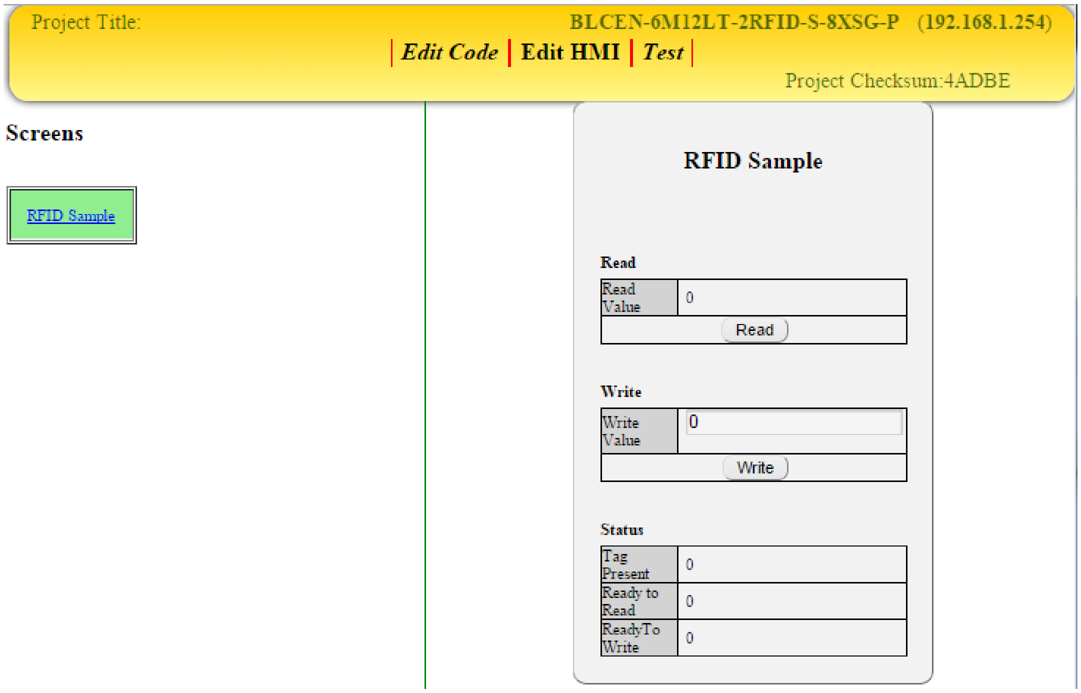
Add New Element

Add Section

Explaining the RFID HMI: The user creates a “Read” section with two elements. One element displays the tag value and the other is a button that starts the “Read” operation. They also create a “Write” section with two elements. One element allows the user to enter a value and the other is a button that starts the “Write” operation. The “Status” section just displays status bits.

Working with the RFID HMI:

➢ To start working with the RFID HMI, the users clicks *Run* and then *View HMI* from the ARGEE menu bar.



Project Title: BLCEN-6M12LT-2RFID-S-8XSG-P (192.168.1.254)

| Edit Code | Edit HMI | Test |

Project Checksum:4ADBE

Screens

RFID Sample

RFID Sample

Read

Read Value	0
Read	

Write

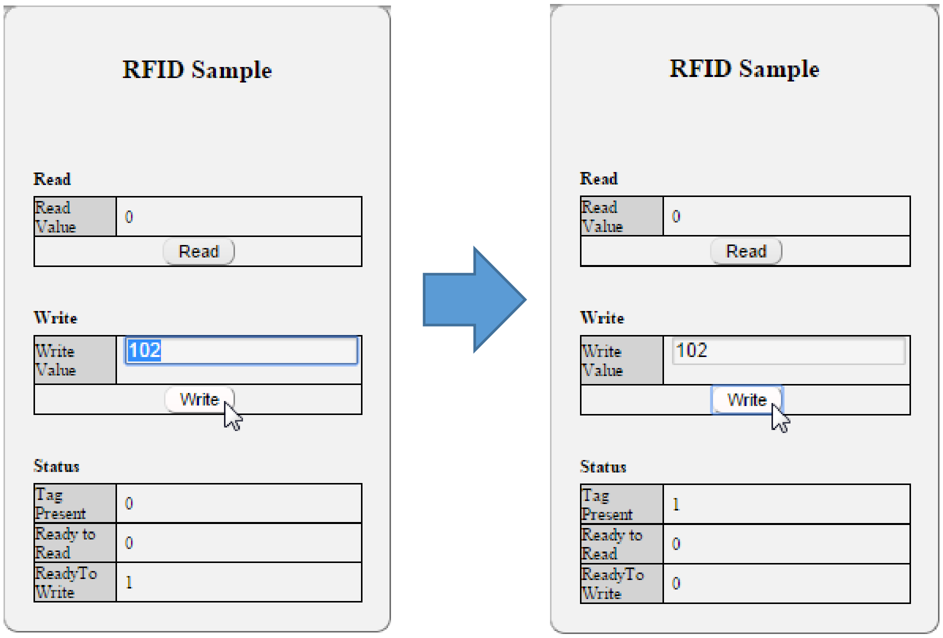
Write Value	0
Write	

Status

Tag Present	0
Ready to Read	0
ReadyTo Write	0

Writing to a tag:

➢ If the user wants to write to a tag, they must first type a value into the *Write Value* field and click the *Write* button.



RFID Sample

Read

Read Value	0
Read	

Write

Write Value	102
Write	

Status

Tag Present	0
Ready to Read	0
ReadyTo Write	1

RFID Sample

Read

Read Value	0
Read	

Write

Write Value	102
Write	

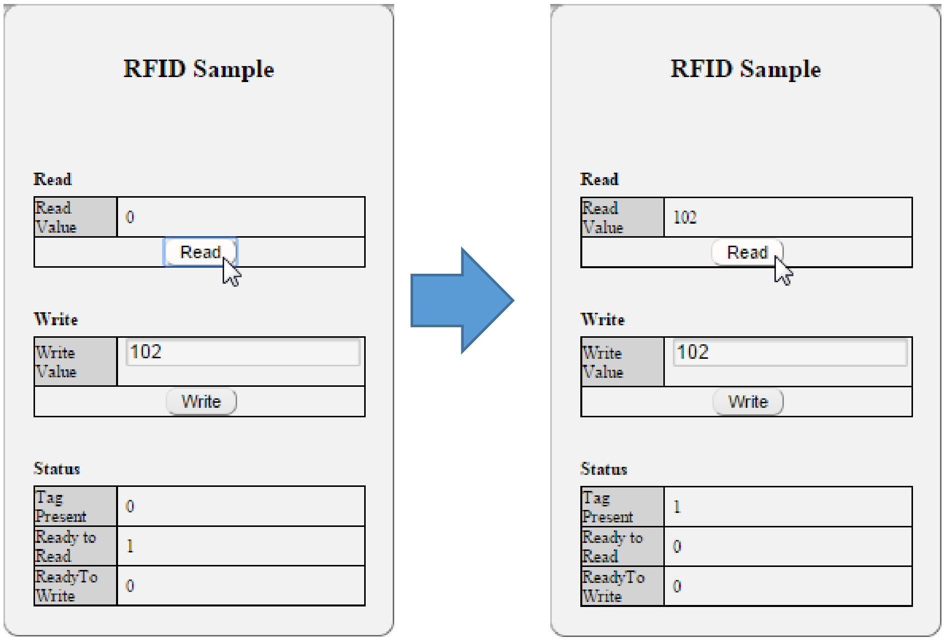
Status

Tag Present	1
Ready to Read	0
ReadyTo Write	0

Explaining "Write": When the user types a value into the *Write Value* field and clicks *Write*, the "Ready To Write" status bit is true. When a tag is presented to the transceiver, the "Ready To Write" status bit goes false and the "Tag Present" bit goes true.

Reading from a tag:

- If the user wants to read tag, simply click the *Read* button and then present a tag to the transceiver.



RFID Sample

Read

Read Value	0

Read

Write

Write Value	102

Write

Status

Tag Present	0
Ready to Read	1
ReadyTo Write	0

RFID Sample

Read

Read Value	102

Read

Write

Write Value	102

Write

Status

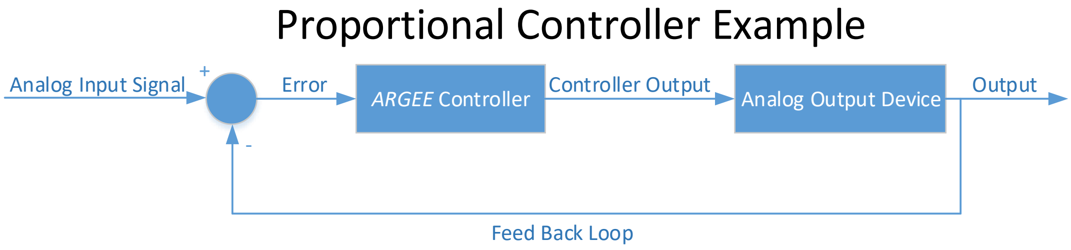
Tag Present	1
Ready to Read	0
ReadyTo Write	0

Explaining "Read": When the user clicks "Read", the "Ready To Read" status bit is true. When a tag is presented to the transceiver, the "Ready To Read" status bit goes false and the "Tag Present" bit goes true. The tag value is displayed in the "Read Value" field.

14.7 Working with Analog

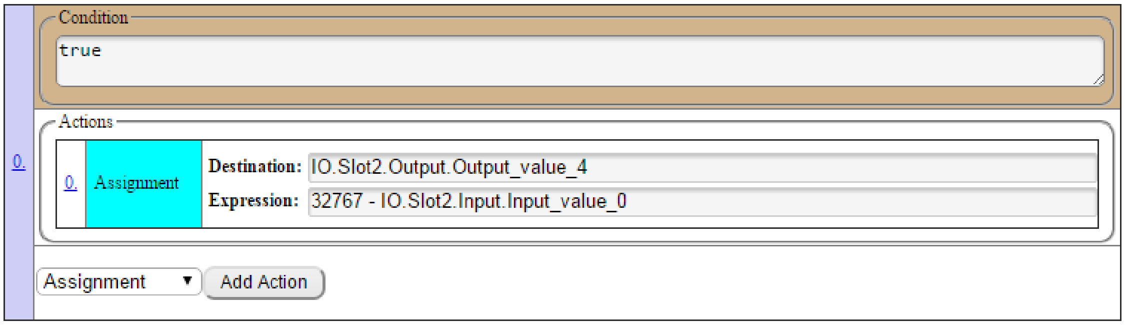
If the user wants to use an Analog input signal to track errors and make corrections to an Analog output signal (Similar to a proportional Controller), they no longer need a PLC. ARGEE has the ability to apply logic and math to analog signals.

Explaining a Proportional Controller: A proportional controller continuously calculates the difference between the output and the input. The purpose of a proportional controller is to minimize the difference (error) by adjusting the controller's output.



Example of Analog - Proportional Controller
The user wants to create a simple proportional controller, where an Analog input signal inversely controls an Analog output signal. The user must first teach the analog input sensor what the minimum and maximum ranges of the application are (Read sensor data sheet to learn how to teach minimum and maximum ranges). The user is using a 4-20 mA input signal and outputting a 0-10 VDC signal.

Write the Code:



true

Destination: IO.Slot2.Output.Output_value_4
Expression: 32767 - IO.Slot2.Input.Input_value_0

Explaining the Code: Analog sensors use 16 bit signed integers. Therefore the range of the analog input signal is from -32767 -> +32767. The user want's an inversely proportional controller, so they are taking 32767 – Input_value_0 and loading that value into Output_value_4.

15 Appendix

15.1 I/O Variable Definitions

15.1.1 Slot “0” Diagnostics Definitions

Term	Definition
Module_Diagnostics_Available	Module Diagnostics Bit
Station_Configuration_Changed	Station Configuration Changed Bit
Overcurrent_Isys	Station Overcurrent Register Bit
Overvoltage_Field_Supply_V1 - Overvoltage_Field_Supply_V2	Station Overvoltage Register Bit
Undervoltage_Field_Supply_V1 - Undervoltage_Field_Supply_V1	Station Under Voltage Register Bit
Modulebus_Communication_Lost	Module communication register Bit
Modulebus_Configuration_Error	Module Error Bit
Force_Mode_Enabled	Force Mode Enabled Bit

15.1.2 Slot 1 or 2 Input Definitions

Term	Definition
Input_Value_0 – Input_Value_7	Input Channel Registers
XCVR_DETUNED_0 - XCVR_DETUNED_1	Transceiver Detuned Bit
TFR_0 – TFR_1	Transfer Data Bit
TP_0 – TP_1	Tag Present Bit
XCVR_ON_0 - XCVR_ON_1	Transceiver On Bit
XCVR_CON_0 - XCVR_CON_1	Transceiver Connected Bit
Error_0 – Error_1	Error Bit
Busy_0 – Busy_1	Busy Bit
Done_0 – Done_1	Done Bit
Error_code_0_0 - Error_code_2_0	Error Code Bits
Read_data_0_0 – Read_data_7_0	Read Data Registers

15.1.3 Diagnostics Definitions

Term	Definition
Output_signal_overcurrent_1 - Output_signal_overcurrent_16	Signal Overcurrent Error Bit
Overcurrent_on_sensor_group	Sensor Overcurrent Error Bit
Overcurrent_supply_VAUX1/2_at_channels_1-7	Supply Overcurrent Error Bit
Overcurrent_VAUX1/2_Digital_In_CH1-16	AUX Power Overcurrent Error Bit
Measued_value_out_of_range_0 - Measued_value_out_of_range_3	Measured Value Out of Range Bit
Wire_break_0 – Wire_break_3	Wire Break Bit. Used for wire break detection
Hardware_failure_0 – Hardware_failure_7	Hardware Failure Bit
Output_value_out_of_range_4 - Output_value_out_of_range_7	Output Value Out of Range Bit
Output_signal_overcurrent_0 - Output_signal_overcurrent_16	Output Signal Overcurrent Bit
Transc_param_not_supported_0/1	Transceiver Parameter Not Supported Bit
Module_parameter_invalid_0/1	Module Parameter Invalid Bit

Term	Definition
Hardware_failure_transceiver_0/1	Transceiver Hardware Failure Bit
Transc_power_supply_error_0/1	Transceiver Power Supply Error Bit

15.1.4 Slot 1 or 2 Output Definitions

Term	Definition
Output_value_0 – Output_value_7	Output channel register
Reset_0 – Reset_1	Transceiver Reset Bit
XCVR_Info_0 - XCVR_Info_1	Transceiver Information Bit
TAG_Info_0 - TAG_Info_1	Tag Information Bit
Write_0 – Write_1	Write Bit
Read_0 – Read_1	Read Bit
Tag_ID_0 – Tag_ID_1	Tag ID Bit
Next_0 – Next_1	Next Bit
XCVR_0 – XCVR_1	Turn Transceiver On Bit
Byte_count_0 – Byte_count_2	The Byte Count Bytes
Domain_0 – Domain_1	Domain Bit
Address_0 – Address_1	Set Read/Write Address Bit
Write_data_0_0 - Write_data_7_0	Write Registers