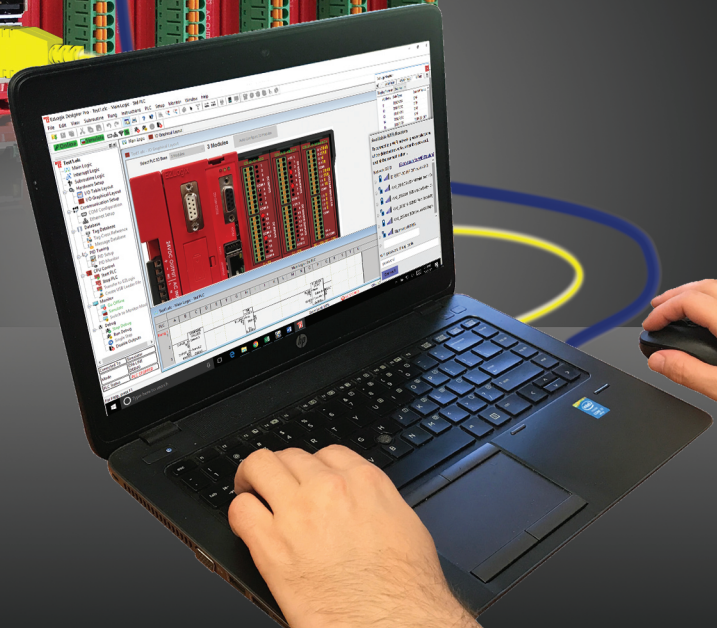
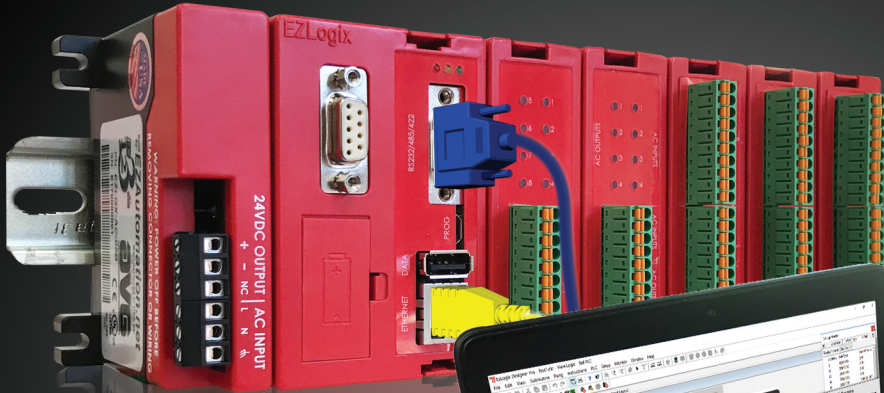


EZLogix

Software Manual



EZAutomation.net
Innovative Low Cost Automation Equipment Made in *America* 

This page intentionally left blank.



Software Manual
Revision A.2.0

WARNING!

Programmable control devices such as EZRACK PLC are not fail-safe devices and as such must not be used for stand-alone protection in any application. Unless proper safeguards are used, unwanted start-ups could result in equipment damage or personal injury. The operator must be made aware of this hazard and appropriate precautions must be taken.

In addition, consideration must be given to the use of an emergency stop function that is independent of the EZRACK PLC.

The diagrams and examples in this user manual are included for illustrative purposes only. The manufacturer cannot assume responsibility or liability for actual use based on the diagrams and examples.

Trademarks

This publication may contain references to products produced and/or offered by other companies. The product and company names may be trademarked and are the sole property of their respective owners. EZAutomation disclaims any proprietary interest in the marks and names of others.

**© Copyright 2017, EZAutomation
All Rights Reserved**

No part of this manual shall be copied, reproduced, or transmitted in any way without the prior written consent of EZAutomation. EZAutomation retains the exclusive rights to all information included in this document.

Designed and Built by AVG

4140 Utica Ridge Rd. Bettendorf, IA 52722-1327

Marketed by EZAutomation

4140 Utica Ridge Road Bettendorf, IA 52722-1327

Phone: 1-877-774-3279 (EASY) Fax: 1-877-775-3279 www.ezautomation.net

Table of Contents

Chapter 1: Getting Started

1.1 EZRack PLC Designer Pro.....	11
1.1.1 System Requirements	11
1.1.2 Installation	11
1.2 EZSTART (Creating/Opening PLC project)	14
1.2.1 Programming Ladder Logic	18
1.2.2 Creating a Complete Rung	20

Chapter 2: EZRack PLC Designer Pro User Interface

2.1 Main Programming Screen	23
2.2 Standard Toolbar	25
2.3 Instructions Toolbars	26
2.3.1 Relay/Boolean Operations.....	26
2.3.2 Compare Operations.....	26
2.3.3 Math Operations.....	27
2.3.4 Bitwise Operations.....	27
2.3.5 Move Operations	28
2.3.6 Timer/Counter/Drum.....	28
2.3.7 Program Control Operations.....	28
2.3.8 String Operations	28
2.3.9 Communications Operations	29
2.3.10 Data Log Operations	29
2.3.11 Datatype Conversion Operations.....	29
2.3.12 Process Alarms / Faults Operations	29
2.3.13 Analog Operations	30
2.3.14 Function Blocks Operations	30
2.3.15 IIoT Operations	30
2.4 PLC TOOLBAR	31
2.4.1 Project Operations	31
2.4.2 Online / Simulate Operations	31
2.4.3 Debug Operations	31
2.5 MENUS	32
2.5.1 File Menu	32
2.5.2 Edit Menu.....	38
2.5.3 View Menu.....	41

- 2.5.4 Subroutine Menu 43
- 2.5.5 Rung Menu..... 44
- 2.5.6 Instructions Menu..... 46
- 2.5.7 PLC Menu 48
- 2.5.8 Setup Menu..... 51
- 2.5.9 Monitor Menu..... 75
- 2.5.10 Window Menu 77
- 2.5.11 Help Menu 78
- 2.5.12 Right-Click Menus 79
- 2.6 Project View / Quick Access Bar 80
- 2.7 Operator Bar 82
- 2.8 I/O Graphical View 83

Chapter 3: Instructions for Programming EZRack PLC

- 3.1 Ladder Logic Programming in EZRack PLC 86
- 3.2 Memory Map 87
 - 3.2.1 System Discretes 88
 - 3.2.2 System Registers 89
- 3.3 RLL Instructions in EZRack PLC..... 91
 - 3.3.1 Available Data Types (Creating and Using) 96
 - 3.3.2 Auto Generated Tags / Auto Fill Tag Address 99
 - 3.3.3 Relay/Boolean Instructions..... 107
 - 3.3.4 Compare Instructions..... 114
 - 3.3.5 Math Instructions..... 121
 - 3.3.6 Bit Logic Instructions..... 132
 - 3.3.7 Move Instructions 137
 - 3.3.8 Timer/Counter/Drum Instructions..... 144
 - 3.3.9 Program Control Instructions 156
 - 3.3.11 Communication Instructions..... 169
 - 3.3.12 Data Logging Instructions 181
 - 3.3.13 Datatype Conversion 188
 - 3.3.14 Process Alarms/Faults..... 191
 - 3.3.15 Analog 197
 - 3.3.16 Function Blocks 204
 - 3.3.17 IloT 228

Chapter 4: Simulating / Monitoring / Debugging PLC Logic

- 4.1 PLC Simulator Functions 235
 - 4.1.1 Simulating your PLC Logic 235
 - 4.1.2 Simulator Functions 236
 - 4.1.3 Simulator IO View 236
 - 4.1.4 Simulator Debugging..... 236
- 4.2 Online Mode 237
 - 4.2.1 Edit Online..... 238
 - 4.2.2 Monitor Online 240
 - 4.2.3 Forcing I/O 244
- 4.3 Debugging PLC Logic 245
 - 4.3.1 Debug Mode 245
 - 4.3.2 Breakpoints 247
 - 4.3.3 Run/Single Step..... 249

Chapter 5: Message Display on EZMarquee

- 5.1 Message Display on EZMarquee 252
- 5.2 Message Controller Function..... 253
 - 5.2.1 Message Database 255
 - 5.2.3 Displaying Messages 261
 - 5.2.4 Example..... 262

Chapter 6: PID Loop

- 6.1 Introduction to PID 267
- 6.2 PID Setup..... 270
- 6.3 PID Monitor..... 281

Chapter 7: EZRack Communications (Modbus, ASCII, etc.)

- 7.1 Supported EZRack Communications 286
 - 7.1.1 EZRack Serial Communications 286
 - 7.1.2 EZRack Ethernet Communications..... 286
- 7.2 Modbus Communications 287
 - 7.2.1 Setup EZRack as an Ethernet Modbus Master 289
 - 7.2.2 Setup EZRack as an Ethernet Modbus Slave 293
 - 7.2.3 Setup EZRack as a Serial Modbus Master (Modbus RTU)..... 294

- 7.2.4 Setup EZRack as a Serial Modbus Slave (Modbus RTU) 300
- 7.2.5 Modbus Tips and Troubleshooting 303
- 7.3 ASCII Communication 305
 - 7.3.1 Setup EZRack to Send Out ASCII Communications 307
 - 7.3.2 Setup EZRack to Receive ASCII Communications..... 308

Chapter 8: IIoT (Industrial Internet of Things)

- 8.1 IIOT (Industrial Internet of Things) 310
 - 8.1.1 MQTT 311
- 8.2 MQTT Essentials..... 313
 - 8.2.1 Basic Concepts 313
 - 8.2.2 MQTT More Details and Examples 315
- 8.3 Basic MQTT Setup on EZRack PLC..... 324
- 8.4 Broker Setup 326
- 8.5 EZRack PLC IIoT (MQTT) Example 328
- 8.6 MQTT HIVEMQ Essentials 332
- 8.7 EZ-IIoT Subscriber Utility..... 337
 - 8.7.1 Install EZ-IIoT Subscriber Utility 337
 - 8.7.2 EZ-IIoT Subscriber Utility Setup 338
 - 8.7.3 EZ-IIoT Subscriber Utility Functions 340
 - 8.7.4 EZ-IIoT Subscriber Utility Best Practices 349

Chapter 9: EtherNet/IP

- 9.1 EtherNet/IP Basics 351
 - 9.1.1 Implicit vs Explicit Messaging..... 351
 - 9.1.1 Explicit Messaging Details..... 352
 - 9.1.3 Implicit Messaging Details 353
- 9.2 EtherNet/IP Adapter Setup 354
 - 9.2.1 EZRack PLC Setup 354
 - 9.2.2 Allen-Bradley Setup 356
 - 9.2.3 Troubleshooting..... 358

Chapter 10: EZRack Modules

- 10.1 Basic Modules 360
- 10.2 Specialty Modules 366
 - 10.2.1 High Speed Counter (EZRPL-IO-HSCNT) 367
 - 10.2.2 Resistance Temperature Detector Module (EZRPL-IO-4RTD)..... 376
 - 10.2.3 Thermocouple Modules (EZRPL-IO-4THIE) 379

Chapter 11: Sparkplug B (IIoT / MQTT) Setup and Basic Info

- 11.1 Sparkplug B IIoT (MQTT) Basic Setup..... 387
- 11.2 Basic Ignition MQTT Modules Setup..... 393
- 11.3 Advanced Sparkplug Setup (Security and Encryption)..... 394
 - 11.3.1 Encryption and Certificate Basics..... 394
 - 11.3.2 EZRack PLC Encryption and Certificate Authority Setup:..... 396
 - 11.3.3 Ignition Encryption and Keystore Setup 397
- 11.4 Redundancy Setup (EZRack PLC and Ignition): 398
- 11.5 Store and Forward Setup: 399
 - 11.5.1 Store and Forward Time Zone Setup 400
- 11.6 Troubleshooting Sparkplug B Setup: 402

Technical Support

For technical questions please consult this manual, the hardware manual or the EZRack PLC Designer Pro Programming Software Help. Finally you can find answers in the download's section of our website www.ezautomation.net. If you still need assistance, please call our technical support at 1-877-774-EASY or FAX us at 1-877-775-EASY.

Manual Revision History

Revision	Date	Pages Affected	Changes Made
A.1	June 2017	All	Original
A.1.1	July 2017	Section 8.7	Added Information
A.2.0	October 2017	All	Name change, auto addressing change, Add Ch. 9 and Ch. 10
A.2.1	January 2018	Section 2.5.8, Section 3.2, Chapter 11	BCD32 support dropped, Sparkplug support added, Upgrade Firmware



Chapter 1: Getting Started

In this Chapter...

1.1 EZRack PLC Designer Pro.....	11
1.1.1 System Requirements.....	11
1.1.2 Installation.....	11
1.2 EZSTART (Creating/Opening PLC project).....	14
1.2.1 Programming Ladder Logic.....	18
1.2.2 Creating a Complete Rung.....	20

1.1 EZRack PLC Designer Pro

EZRack PLC Designer Pro is an intuitive and simple to use Relay Ladder Logic (RL) Editor for programming EZAutomation's EZRack PLC. EZRack PLC Designer Pro includes functionality to create new EZRack PLC programs, edit existing EZRack PLC programs, simulate EZRack PLC programs, go online and monitor the EZRack PLC, and debug any EZRack PLC programs.

1.1.1 System Requirements


The EZRack PLC Designer Pro works on Windows PCs running Windows 7 or Windows 10 and requires at least 60 MB of free space on hard drive for installation.

Use an EZ-PGMCBL cable, EZ-WiFi module or Micro-USB to transfer/write your program from the PC to EZRack PLC.

After initial setup the EZRack PLC can be connected to over Ethernet.

1.1.2 Installation

The EZRack PLC Designer Pro is distributed as a single setup file. The setup file for the Designer Pro is **EZRack PLC Designer Pro x.x.x (FULL) Setup.exe**.

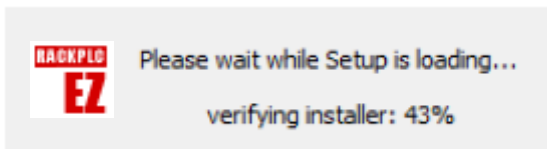
 EZRack PLC Designer Pro 2.0.0 (FULL) Setup

Installation of the EZRack PLC Designer Pro is quick and simple. Just run the setup file and follow the on screen instructions. The default directory where the software installs is "**C:\Program Files (x86)\EZAutomation\EZRack PLC Designer Pro**". You may also choose to install EZRack PLC Designer Pro in another directory as specified in installation settings. If you are familiar with the installation process, you may skip the detailed instructions.

To Install

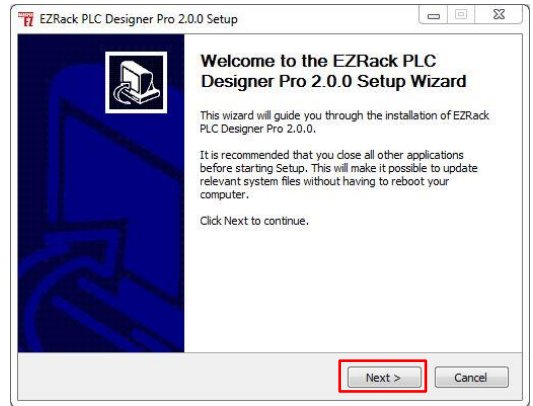
Below are the detailed instructions for installing the software. Just follow the instructions step by step to install EZRack PLC Designer Pro on your hard drive.

1. Double click on the Setup file. It will verify the installer files.



Once verifying is complete, it will show you the next window.

2. Click “Next” button.

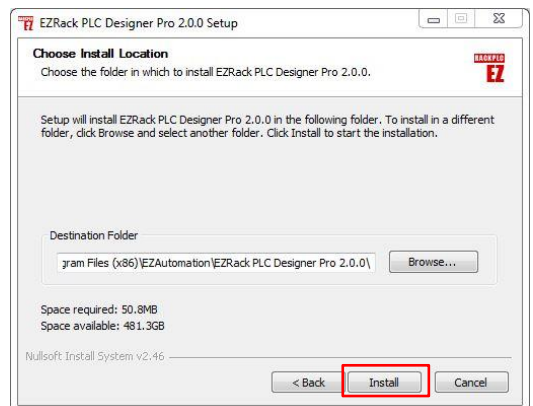


3. Please read the License Agreement text and if you accept, click on “I Agree”

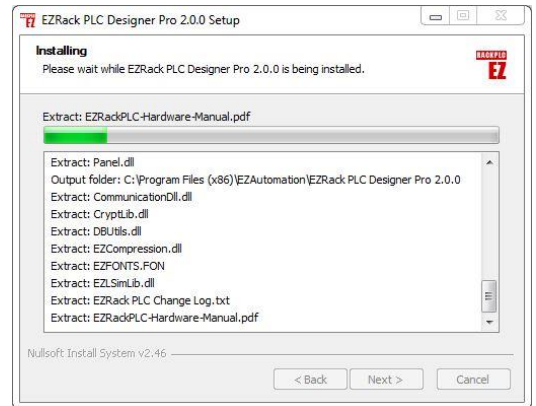


4. The setup program will display the dialog box below to allow you to choose the installation folder. As a default, the folder is “C:\Program Files (x86)\EZAutomation\EZRack PLC Designer Pro”. You can change this if you would like to.

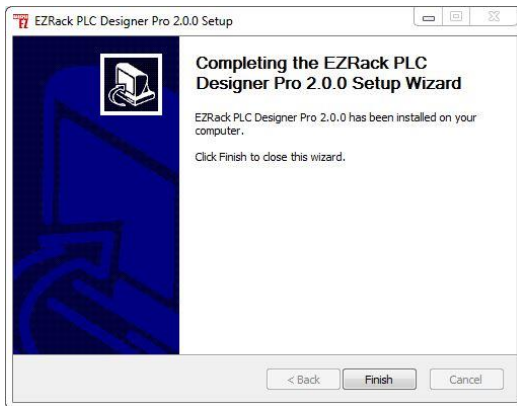
Click the **Install** button to start installation.



5. During installation you will see a dialog box which will list in detail the files being copied to your hard drive for installation.



6. After copying necessary files and making registry entries, the installation is complete, and you will see the dialog box below. Click the **Finish** button to finish the installation. The setup program will place a shortcut icon on your desktop.



To Uninstall

If you need to uninstall this program, you can use the **Uninstall** command from **Start->All Programs->EZRack PLC-> Uninstall**.

The uninstaller will prompt you to make sure that you want to uninstall EZRack PLC Designer Pro and all its components from your computer. If you select YES, all the components of EZRack PLC Designer Pro will be uninstalled from your computer.

1.2 EZSTART (Creating/Opening PLC project)

You can start the EZRack PLC Designer Pro in one of the following 2-ways:

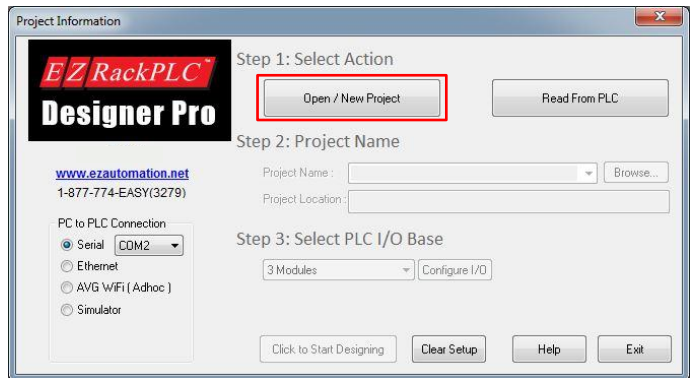
1. Click the EZRack PLC Designer Pro Icon.
2. Select the program using **Start>All Programs>EZAutomation>EZRack PLC Designer Pro.**

The dialog box allows you to select Programming mode, Project folder location, and Project Name. In addition, you can configure the EZRack PLC I/O base (defining I/O module locations and addresses) from this dialog box.

Step 1: Select ACTION

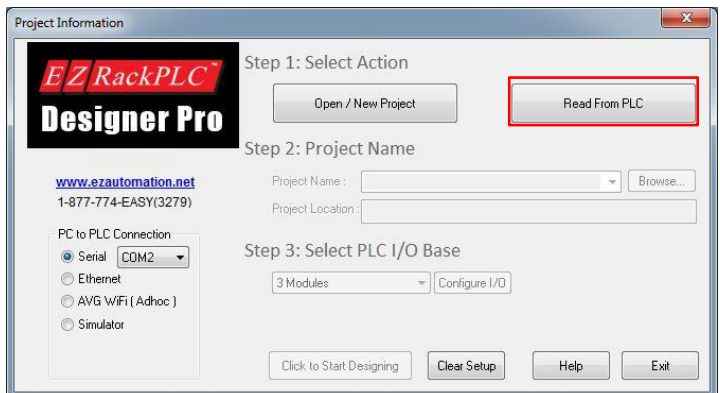
Open / New Project:

Select this mode to create a new program or edit an existing program in OFFLINE mode.



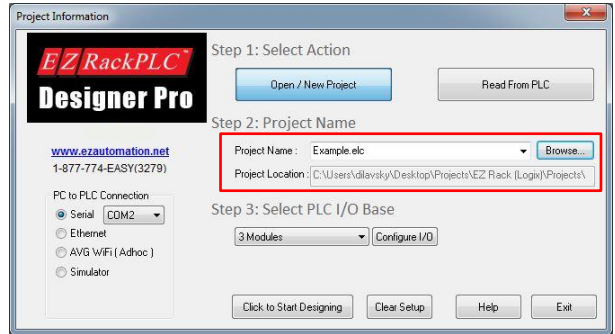
Read From PLC

This mode allows you to first read an existing project from a PLC, save it on PC, and then edit your program OFF-LINE. You may want to use this mode if you do not have your program on your computer. For reading back the program, *the EZRack PLC can be in RUN or PROGRAM mode.*



Step 2: Select Project Name

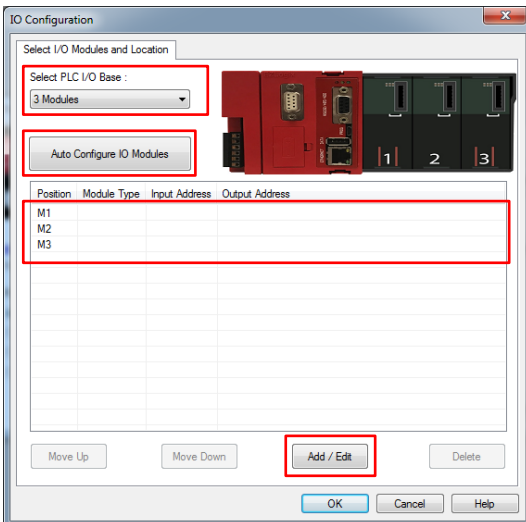
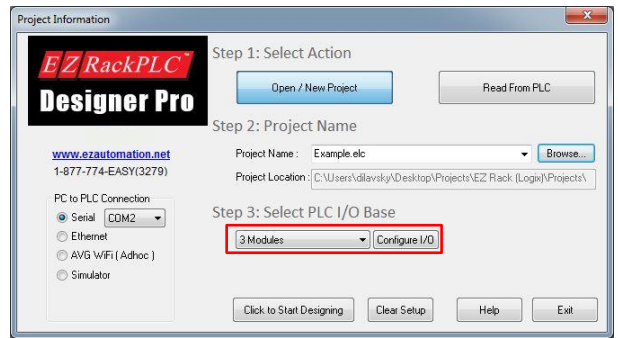
Enter the name of the project. The Project Location field indicates the folder name where the Project will be saved. If need be, use the **Browse** button to select a different Project location.



Step 3: Selecting and Configuring I/O Base

(You can do this later when you start programming)

Select the I/O base for your PLC. Currently, EZ Rack PLC offers I/O bases for 3, 5, and 7 modules. After selecting the I/O base size, click on the **Configure I/O** button to define the placement and the addresses of the I/O modules (See dialog box on left).



The Module slot positions are identified as M1, M2, M3 etc., on the I/O base. The dialog box shows only the available module positions for the selected I/O base. For example, a 3-module base will show only M1-M3 positions, while a 7-slot base will display rows M1-M7.

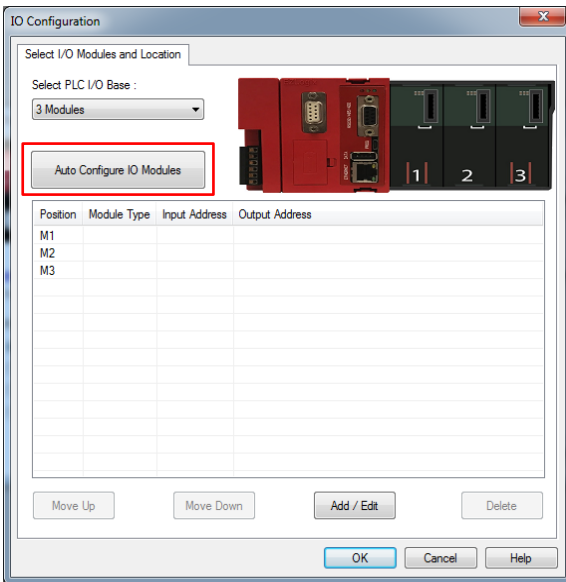
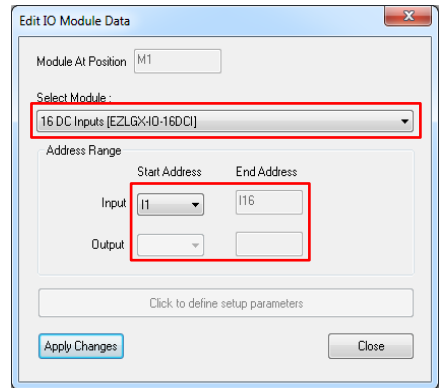
To **Manually Configure** a module on a position, double click the row corresponding to the position number (say M1) or click the **Add/Edit** button.

OR

You can also **Auto Configure** the IO Modules by clicking the “Auto Configure IO Modules” button.

Manual Configuration

After clicking the Add/Edit button you will get the Edit IO Module Data Dialogue Screen. Select the module type from the available modules and it's I and/or O (IR and OR for analog) addresses from respective drop downs. You select the start address of the module, and the software computes and fills up the end address of the module automatically.



Auto Configure

For Auto Configure please make sure that the PC and PLC are connected (either serially, over micro-usb, or over Ethernet).

After clicking the “Auto Configure IO Modules” button you will see dialog on the next page.

The EZRack PLC Designer Pro will read back the PLCs configuration and the table on the next page will display the results. It will also display the rack size detected in the configuration.

Current Configuration -- This the currently selected configuration in the EZRack software.

Detected Configuration -- This the detected configuration of the EZRack PLC.

Select Configuration -- Please select whether to use the current configuration in the software or the detected configuration. You can also select All in the section right under the title.

Final Configuration -- This will display the configuration which will be now configured in the software.

Auto Configure PLC Hardware

Slot#/ Size	Current Configuration	Detected Configuration	Select Configuration	Final Configuration
Rack	3 Modules	3 Modules	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	3 Modules
M1	8 Analog In, 4 Analog Out Voltage EZPL-IO-8ANI4ANOV Input Address: IR1 - IR8 Output Address: OR1 - OR4	8 Analog In, 4 Analog Out EZPL-IO-8ANI4ANOV	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	8 Analog In, 4 Analog Out Voltage EZPL-IO-8ANI4ANOV Input Address: IR1 - IR8 Output Address: OR1 - OR4
M2	16 DC Inputs EZPL-IO-16DCI Input Address: I1 - I16	16 DC Inputs EZPL-IO-16DCI	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	16 DC Inputs EZPL-IO-16DCI Input Address: I1 - I16
M3	16 DC Outputs Sourcing EZPL-IO-16DCOP Output Address: O1 - O16	16 Discrete Outputs EZPL-IO-16DCOP	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	16 DC Outputs Sourcing EZPL-IO-16DCOP Output Address: O1 - O16
M4	Not Applicable	Not Applicable	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	Not Applicable
M5	Not Applicable	Not Applicable	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	Not Applicable
M6	Not Applicable	Not Applicable	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	Not Applicable
M7	Not Applicable	Not Applicable	<input type="radio"/> Use Current <input checked="" type="radio"/> Use Detected	Not Applicable

OK Cancel

Please note: there are certain module families which include multiple modules of different types like “Sinking” or “Sourcing” for Digital Outputs and “Voltage” or “Current” for Analog modules cannot be differentiated. Therefore please make sure the correct module is selected under the detected configuration.

Step 4: Configure PC to PLC Communication

(You can do this later too, when you start programming)

This allows you to select the communication port on your PC that would be used to transfer developed ladder logic to the EZRack PLC. You can make this selection later in the COM Configuration options of main design screen.

Serial Communication

For serial communication you can use a serial cable (EZ-PGMCBL see Appendix 1). If your computer does not have a serial port then we recommend using a USB to Serial Converter. Serial Communication is also the option used when using a Micro-USB cable to program you PLC. The EZRack PLC Designer Pro will auto detect the Com Port that are currently being used. Please select the correct Com Port which corresponds to the communication cable. The EZRack PLC utilize fixed serial com parameters (set at 38.4 Baud, 8, N, 1).

PC to PLC Connection

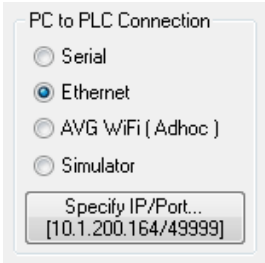
Serial COM2

Ethernet

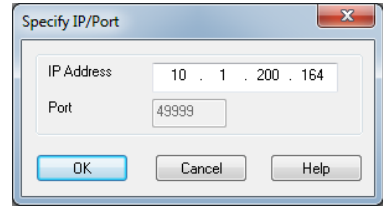
AVG WiFi (Adhoc)

Simulator

Ethernet Communication

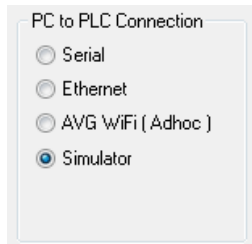
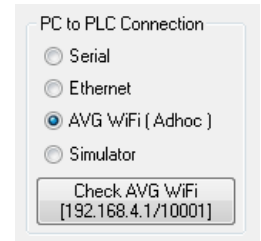


Ethernet communication can be used once the EZRack PLC IP address is configured. To use Ethernet Communication please specify the IP address of the PLC you wish to communicate with. Please note that if your computer has trouble communicating please check that your PC subnet and gateway match the PLCs.



AVG WiFi

You can also purchase an EZ-WiFi module to use to communicate with the EZRack PLC, to do so select the AVG WiFi option. The AVG WiFi option will automatically detect any EZ-WiFi module and then it can be used exactly like the Ethernet and Serial communication options.



Simulator

At any time you can simulate your logic by loading it to the simulator. This allows you to see how your created logic functions without needing any external equipment. *Note: This does not download the project to a PLC.*

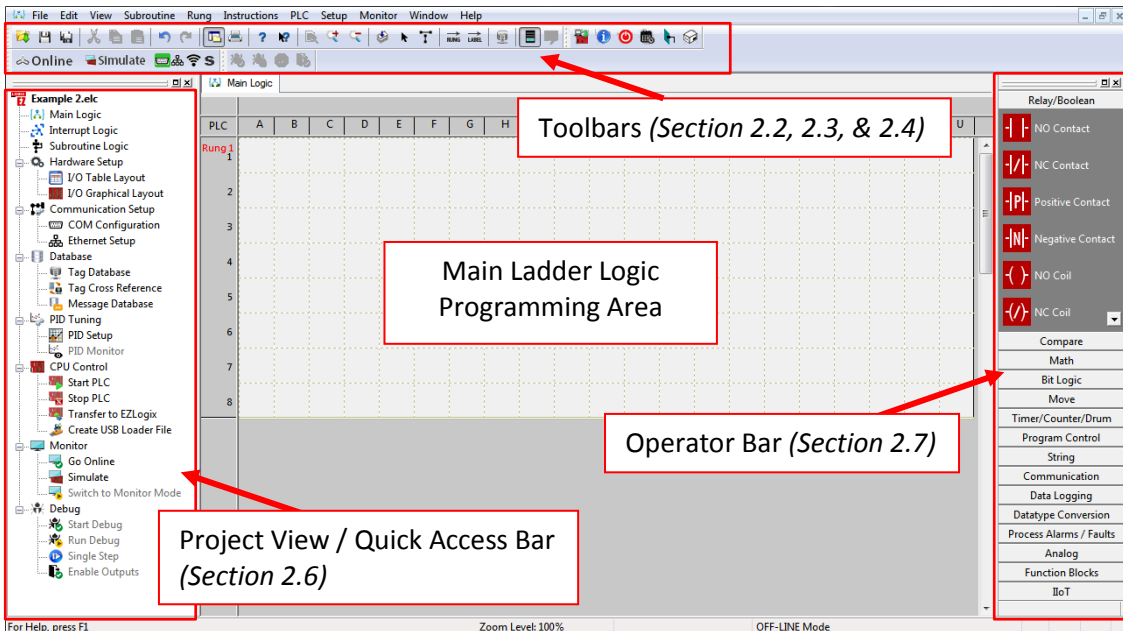
Communication Options Table

Communication Type	PC to PLC Connection Option	Functionality				
		Transfer to PLC	Read from PLC	Monitor PLC (Online)	Debug PLC (Online)	Upgrade Firmware
Serial Cable (EZ-PGMCBL)	Serial (Select Com Port)	✓	✓	✓	✓	✓
Micro-USB Cable	Serial (Select Com Port)	✓	✓	✓	✓	✓*
Ethernet	Ethernet (Input IP Address)	✓	✓	✓	✓	✓*
EZ-WiFi	AVG WiFi	✓	✓	✓	✓	✓*
USB Drive	USB Drive (Create USB file)	✓	✗	✗	✗	✓*

✓* Note: Upgrading using these methods requires the EZRack Editor v2.1 or higher and the firmware on the EZRack PLC to be vA.0.297 or higher. If try to upgrade from vA.0.256 or lower these methods will not work.

1.2.1 Programming Ladder Logic

Once you make your selections in the first dialog box and click **OK**, you will come to the main programming screen as shown below:



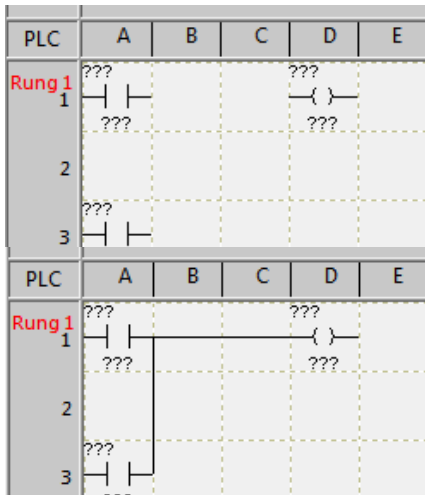
To program a rung, perform the following steps:

1. Select **instruction**. You can select instructions using a menu, or tool bar, or the operator bar on the right. The operator bar provides all instructions symbols organized by types. Once you select an instruction, the cursor changes shape. Click on the location in the rung area where you want to place the instruction.
2. Connect all placed instructions by using the **Line** tool.
3. Double click on any instruction to program its parameters.
4. At any time you may Syntax check the logic by selecting **View>Syntax Check - Current Logic**.
5. Once you are satisfied with the Ladder Logic, you can transfer the developed project to the EZRack PLC by selecting **File > Transfer to PLC** menu.

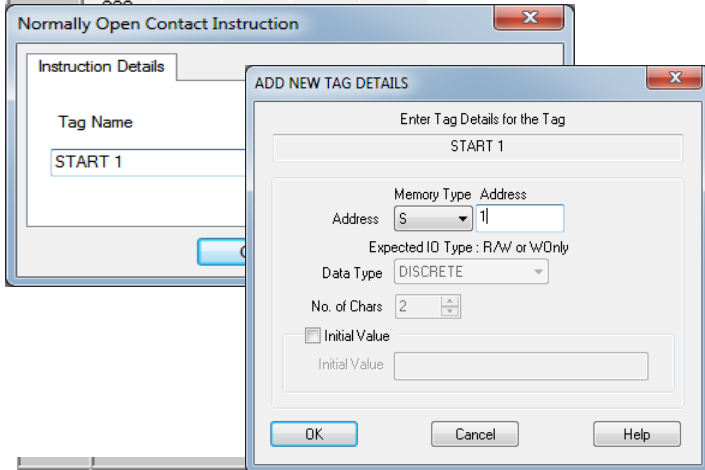
That's all you need to do to program ladder logic for EZRack PLC. The following chapters describe the EZRack PLC Designer Pro User Interface, Ladder Logic Programming, and Instructions in detail.

1.2.2 Creating a Complete Rung

This short example is provided to show you just how easy it is to create a completed rung using EZ Rack PLC Designer Pro. To complete a rung, perform the following steps:

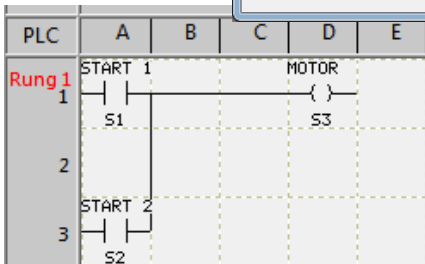


1. Place an instruction onto the Main Logic window. In this example, we've used the **Normally Open Contact** instruction.
2. Place the other instructions you'd like to include in your rung onto the **Main Logic** window. In this example, we've used another **Normally Open Contact** and a **Normally Open Coil** instruction.
3. Use the Line Tool to draw a horizontal line connecting the first **Normally Open Contact** instruction to the **Normally Open Coil** instruction.
4. Use the Line Tool to draw a vertical line connecting the first **Normally Open Contact** instruction to the second **Normally Open Contact** instruction and you're finished completing a rung.



5. Now you just need to add addresses by double clicking on the instructions.

6. In the instruction dialogue input a Tag Name and click OK or press enter. The instruction will prompt you for a Tag Address. For this example use Tag Name: Start 1, Start 2, and Motor. For Tag address use S1, S2, and S3.



7. This is what your Rung should look like when you're finished. It's just that easy!

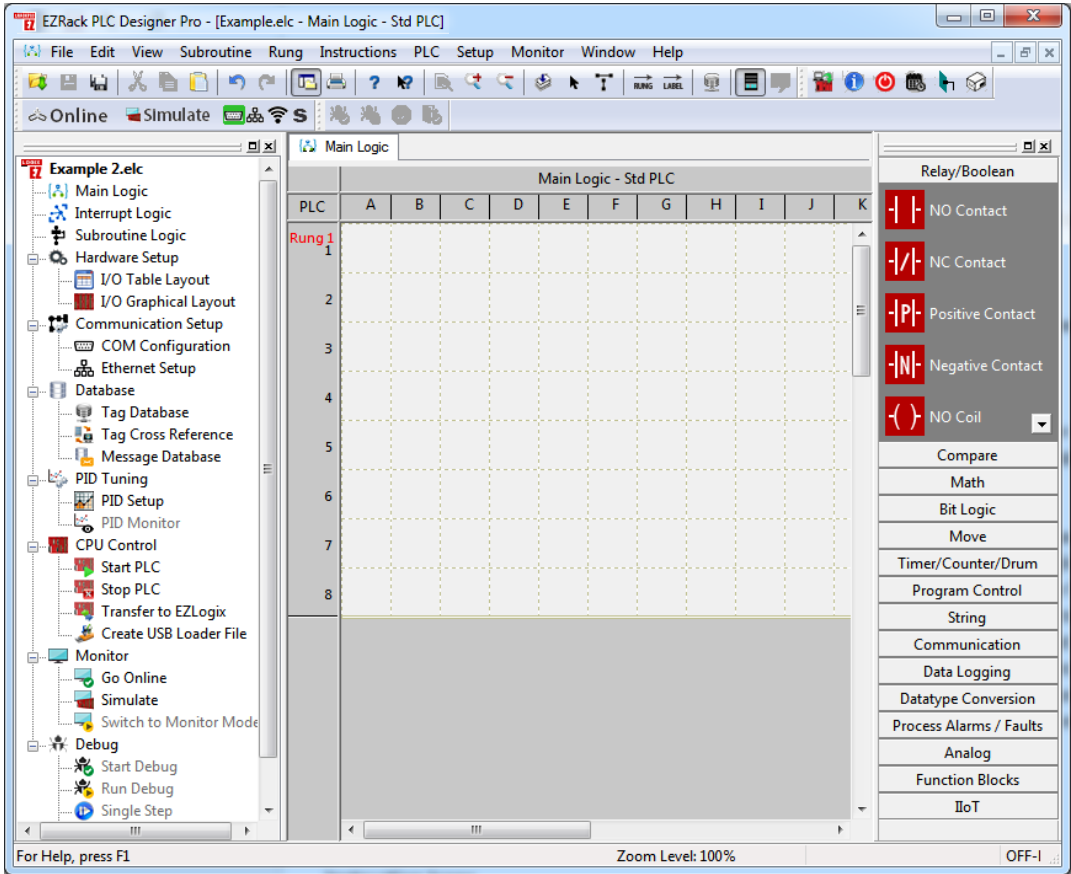
Chapter 2: EZRack PLC Designer Pro User Interface

In this Chapter...

2.1 Main Programming Screen	23
2.2 Standard Toolbar	25
2.3 Instructions Toolbars	26
2.3.1 Relay/Boolean Operations	26
2.3.2 Compare Operations	26
2.3.3 Math Operations	27
2.3.4 Bitwise Operations	27
2.3.5 Move Operations	28
2.3.6 Timer/Counter/Drum	28
2.3.7 Program Control Operations	28
2.3.8 String Operations	28
2.3.9 Communications Operations	29
2.3.10 Data Log Operations	29
2.3.11 Datatype Conversion Operations	29
2.3.12 Process Alarms / Faults Operations	29
2.3.13 Analog Operations	30
2.3.14 Function Blocks Operations	30
2.3.15 IIoT Operations	30
2.4 PLC TOOLBAR	31
2.4.1 Project Operations	31
2.4.2 Online / Simulate Operations	31
2.4.3 Debug Operations	31
2.5 MENUS	32
2.5.1 File Menu	32
2.5.2 Edit Menu	38
2.5.3 View Menu	41

- 2.5.4 Subroutine Menu 43
- 2.5.5 Rung Menu..... 44
- 2.5.6 Instructions Menu..... 46
- 2.5.7 PLC Menu 48
- 2.5.8 Setup Menu..... 51
- 2.5.9 Monitor Menu..... 75
- 2.5.10 Window Menu 77
- 2.5.11 Help Menu 78
- 2.5.12 Right-Click Menus 79
- 2.6 Project View / Quick Access Bar 80
- 2.7 Operator Bar 82
- 2.8 I/O Graphical View 83

2.1 Main Programming Screen



Title Bar

EZRack PLC Designer Pro - [Example.elc - Main Logic - Std PLC]

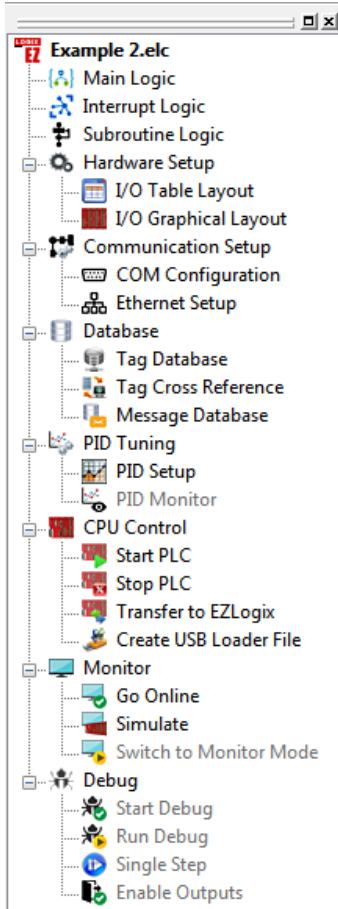
The title bar displays the software name and logo as well as the name of the project currently open.

Main Menu Bar

File Edit View Subroutine Rung Instructions PLC Setup Monitor Window Help

This contains all of the drop menus available in EZRack PLC Designer Pro. Some of the menus are context sensitive, so they are hidden or displayed based on context. See *Section 2.5* for more information.

Project View / Quick Access Bar



This displays different elements being used in your current project. The Project View can also be used as a navigational tool. If you click onto an element in Project View, it will be displayed in the Logic Display Window. It also has quick access to most used functions like Transfer to PLC. Please see *Section 2.6* for more information.

Status Bar



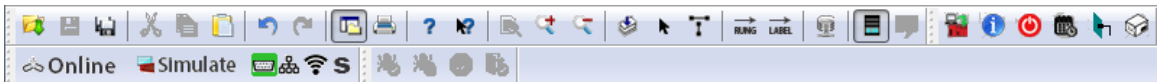
This line at the bottom of the screen displays the status of the current project. And current PLC Status if Online with PLC.

Instruction Icons / Operator Bar



This area contains all of the RLL Instruction icons you will use in your project. Please see *Section 2.7* for more information.

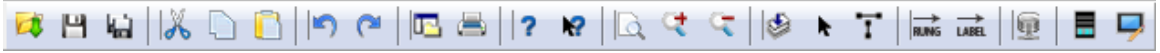
Toolbars


























This is another way for you to access the RLL instructions and many other functions in EZRack PLC Designer Pro. Please see *Section 2.2, 2.3 and 2.4* for more information. Most Toolbars are hidden away as default but can be accessed in **Edit > Toolbars**.

2.2 Standard Toolbar

The EZRack PLC Designer Pro offers multiple toolbars for convenient access to many functions and instructions. These toolbars can be displayed or hidden using menu **Edit > Toolbars**. This section describes various tool bars available in EZRack PLC Designer Pro. For more information about these options see the MENU *Section 2.5*.



- | | |
|---|---|
|  Open Project |  Zoom In |
|  Save Ladder |  Zoom Out |
|  Save Project |  Syntax Check |
|  Cut |  Select Tool |
|  Copy |  Line Tool (Alt-L or double click in Ladder Logic) |
|  Paste |  Go To Rung |
|  Undo |  Go To Label |
|  Redo |  Tag Database |
|  Toggle Project View |  Toggle Operator Bar for Instruction Display |
|  Print |  Monitor Mode / Edit Mode |
|  About | |
|  Help | |
|  Zoom Default | |


2.3 Instructions Toolbars


2.3.1 Relay/Boolean Operations


The Instructions Toolbar consists of icons for all the instructions available for Relay type instructions. These commands are also found in, and accessible from, the **Main Menu > Instructions**.


All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.





 Normally Open Contact

 Normally Closed Contact

 Positive Contact


 Negative Contact

 Normally Open Coil

 Normally Closed Coil


 Set Coil

 Reset Coil

 Normally Open Contact – Immediate Input

 Normally Closed Contact – Immediate Input

 Normally Open Coil - Immediate Output


 Normally Closed Coil - Immediate Output


2.3.2 Compare Operations


The Compare Operations Toolbar consists of icons for all the instructions available for Compare Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.





 Equal To

 Not Equal To

 Greater Than

 Less Than

 Greater Than or Equal To

 Less Than or Equal To

 Limit

 Compare Values

2.3.3 Math Operations

The Math Operations Toolbar consists of icons for all the instructions available for Math Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



+ Add

- Subtract

× Multiply

÷ Divide

⋅ Modulo

ABS Absolute Value

X=Y X=Y Conversion

**BIN
BCD
GRY** Format Conversion

⊗ Advanced Math

2.3.4 Bitwise Operations

The Bitwise Operations Toolbar consists of icons for all the instructions available for Bitwise Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



AND And

OR Or

XOR XOR

NOT Not

101 ← Shift Left

101 → Shift Right

101 ↺ Rotate Left

101 ↻ Rotate Right

2.3.5 Move Operations

The Move Operations Toolbar consists of icons for all the instructions available for Move Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



Move Data



Move Block



Block Fill



Move Table of Constants



Bit Move

2.3.6 Timer/Counter/Drum

The Timer/Counter/Drum Operations Toolbar consists of icons for all the instructions available for Timed Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



Timer



Counter

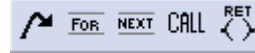


Drum

2.3.7 Program Control Operations

The Program Control Operations Toolbar consists of icons for all the instructions available for Program Control Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



Jump



For Loop



Next



Call Subroutine

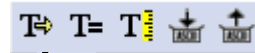


Return (from a subroutine)

2.3.8 String Operations

The String Operations Toolbar consists of icons for all the instructions available for String Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



String Move



String Compare



String Length



Pack String

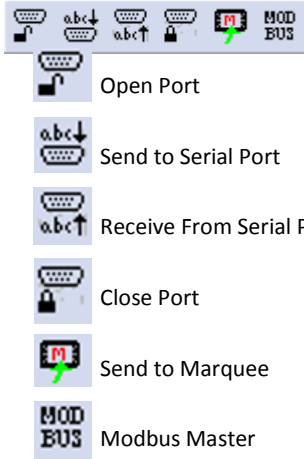


Unpack String

2.3.9 Communications Operations

The Communications Operations Toolbar consists of icons for all the instructions available for Communications Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

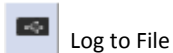
All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



2.3.10 Data Log Operations

The Data Logging Operations Toolbar consists of icons for all the instructions available for Data Log Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.

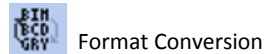
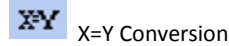
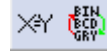


2.3.11 Datatype Conversion Operations

The Datatype Conversion Operations Toolbar consists of icons for all the instructions available for Datatype Conversion Operations. These commands are also

found in, and accessible from, the **Main Menu > Instructions**.

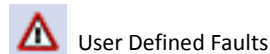
All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



2.3.12 Process Alarms / Faults Operations

The Process Alarms / Faults Operations Toolbar consists of icons for all the instructions available for Process Alarms / Faults Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

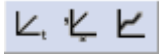
All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.




2.3.13 Analog Operations


The Analog Operations Toolbar consists of icons for all the instructions available for Analog Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



 Ramp Generator

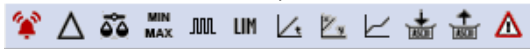
 Scale (Linear)

 Scale (Non-Linear)


2.3.14 Function Blocks Operations


The Function Operations Toolbar consists of icons for all the instructions available for Function Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.



 Alarm

 Change of Values

 Compare Values


 Find Min & Max Value

 Flasher

 Limit


 Ramp Generator

 Scale (Linear)

 Scale (Non-Linear)

 String Pack


 String Unpack

 User Defined Faults

2.3.15 IIoT Operations

The IIoT Operations Toolbar consists of icons for all the instructions available for IIoT Operations. These commands are also found in, and accessible from, the **Main Menu > Instructions**.

All the icons for instructions shown in this section will be described in detail in *Chapter 3 – RLL Instructions*.

 IIoT (MQTT) Publish







2.4 PLC TOOLBAR

The PLC Toolbar consists of icons for all the PLC hardware related functions.

2.4.1 Project Operations

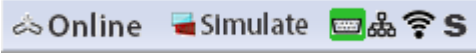
The Project Toolbar consists of icons for all PC to PLC commands. These commands are also found in either **Main Menu > File** or **Main Menu > PLC**. Please see description under Menus for details of these functions.






-  Write to PLC
-  PLC Information
-  Reboot PLC
-  PLC Time and Date
-  COM Configuration
-  OEM Packager

2.4.2 Online / Simulate Operations

The Online / Simulate Toolbar consists of icons for going online with the PLC. You can also simulate your project with a virtual PLC. These commands are also found in **Main Menu > Monitor**. This functionality will be described in detail in *Chapter 4 – Simulating / Monitoring / Debugging PLC Logic*.





-  Online
-  Simulate
-  COM Configuration

2.4.3 Debug Operations


The Debug Toolbar consists of icons for going debugging your project. These commands are also found in **Main Menu > Monitor**.



-  Start Debug
-  Run Debug
-  Single Step
-  Enable Outputs

2.5 MENUS

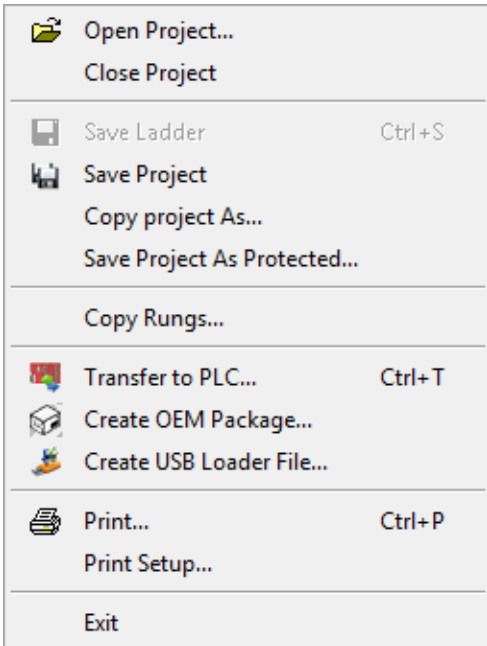
The Main Menu bar consists of the following menus:

 File Edit View Subroutine Rung Instructions PLC Setup Monitor Window Help

Each of the above menus has a pull-down menu with further available options which will be described in this section.

2.5.1 File Menu

When you click onto the File Menu, you can access the following functions:



Open Project

To open an existing project or to create a new project while in a programming window, click on **File > Open Project**. The Step 1, Project Information dialog box will appear. Click on one of the SELECT ACTION buttons. Choose from the available project files or enter a new Project Name. Click on OK to open the project, or Exit to quit without opening.

Close Project

Click on **File > Close Project** to quit the current project.

Save Ladder

Click on **File > Save Ladder** to save the current ladder logic only.

Save Project

Click on **File > Save Project** to save the current project. Ladder, Project Attributes and databases will all be saved.

Copy Project As...

Click on **File > Copy Project As...** to create a copy of your project under another name. *Note: Current save file will remain open.*

Save Project As Protected...

Click on **File > Save Project As Protected** to save the current project as a password protected file. The Protection Password dialog box will appear, as shown on the following page:

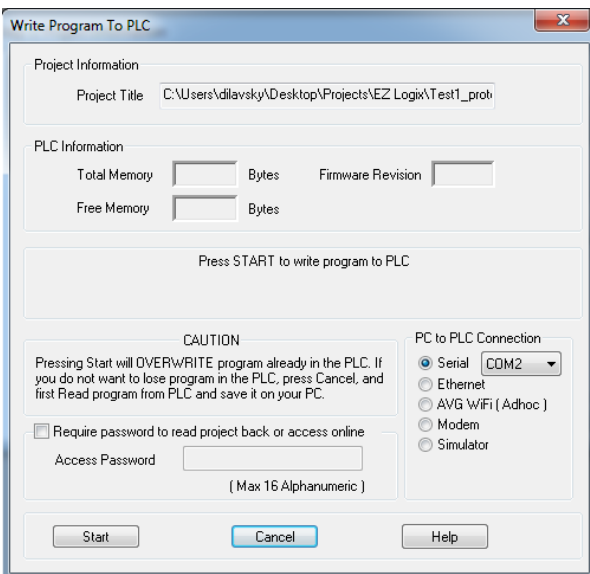
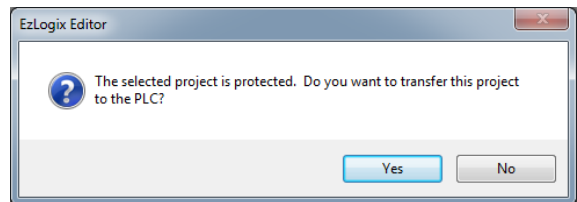


Once saved, if you attempt to open this project again or read this project from an EZRack PLC, you will then be prompted to enter your password, as shown below:



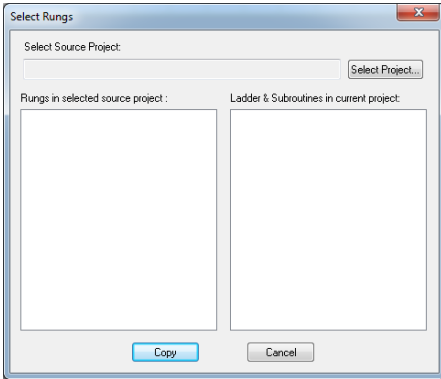
The password protection feature will prevent unauthorized users from viewing/editing the project, but will still allow a user to read from or write to an EZRack PLC.

In the event that a user should not have access to edit a project, but have the ability to write to an EZRack PLC, click Cancel in the window above. This will provide the option to transfer the project to the EZRack PLC, as shown.



Click Yes, and you will be prompted to transfer your project, as shown here.

To Transfer your project first make sure the PC to PLC Connection selection is set to the configuration you are using. For serial cables or Micro-USB cables use the correct COM port. For Ethernet make sure the correct IP address is entered.

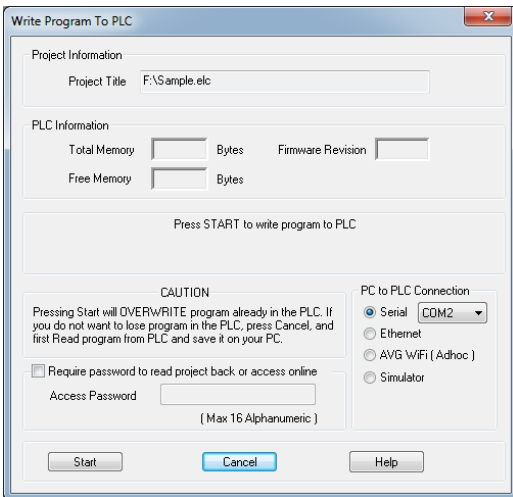


Copy Rungs...

When you click on **File > Copy Rungs...** following window will appear.

Using the Copy Rungs options you can copy rungs from another project into your existing project. Please select the rungs you would like to copy from the source project. Then select the place you would like to copy them to. If copying between Main Logic just select main logic. Otherwise select the subroutine you wish to copy to.

Transfer to PLC...



Transfer to PLC allows you to transfer the current (open) project to the PLC connected to your computer.

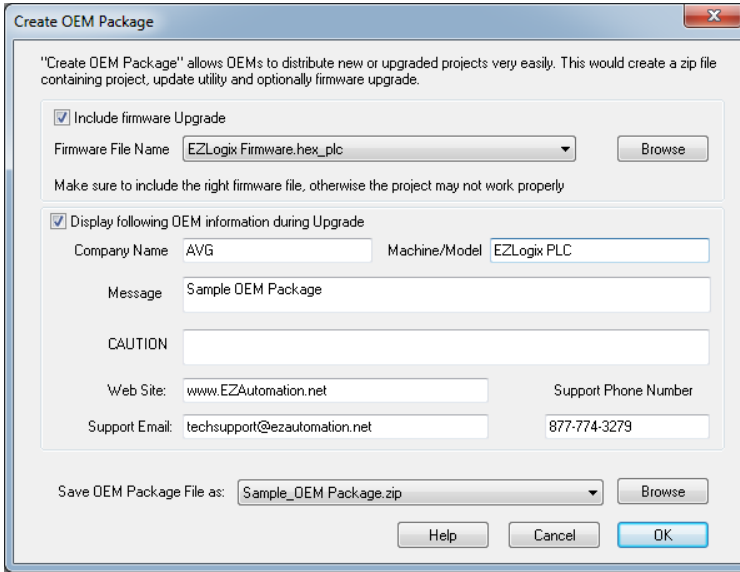
If you would like to password protect this project on the PLC, select and input your password here. *Note: There is no admin password therefore if you forget the password you cannot read back this project.*

Please make sure to configure your PC to PLC communication per what the connection is. See the communication options table below for more information.

Communication Options Table

Communication Type	PC to PLC Connection Option	Functionality				
		Transfer to PLC	Read from PLC	Monitor PLC (Online)	Debug PLC (Online)	Upgrade Firmware
Serial Cable (EZ-PGMCBL)	Serial (Select Com Port)	✓	✓	✓	✓	✓
Micro-USB Cable	Serial (Select Com Port)	✓	✓	✓	✓	✓*
Ethernet	Ethernet (Input IP Address)	✓	✓	✓	✓	✓*
EZ-WiFi	AVG WiFi	✓	✓	✓	✓	✓*
USB Drive	USB Drive (Create USB file)	✓	✗	✗	✗	✓*

✓* **Note:** Upgrading using these methods requires the EZRack Editor v2.1 or higher and the firmware on the EZRack PLC to be vA.0.297 or higher. If try to upgrade from vA.0.256 or lower these methods will not work.



Create OEM Package...

Create OEM Package allows you to create an executable file which can be used without EZRack PLC Designer Pro to transfer both the PLC project and PLC firmware (if need be) to the EZRack PLC.

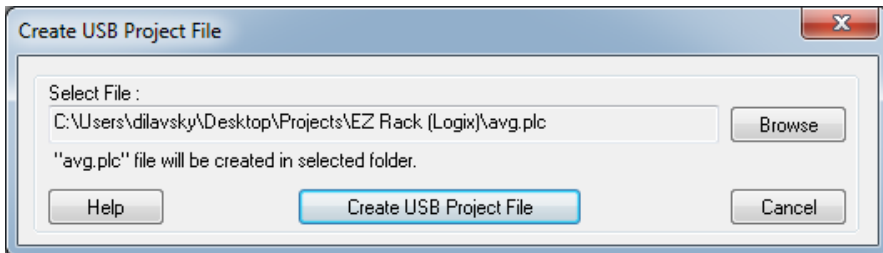
This screen allows you to configure display settings for this OEM exec file. Also here you attach the firmware if needed. After clicking OK the packager will create the file which can then be sent to your user.

Create USB Loader File...

The create USB Loader File option can be used to load a PLC program to the EZRack PLC over an USB drive. To do so please follow the directions below.

Load PLC project over USB Drive (Creating USB Loader File & Loading file to EZRack PLC):

1. To create USB Loader File click on **File > Create USB Loader File**. The following dialog will appear.



2. Use the Browse option to navigate to navigate to the USB drive.
3. Click the Create USB Project File to create the **avg.plc** file(file must be named avg.plc to function).
4. The USB is now ready to be used.

How to use USB to Load project onto EZRack PLC

5. Please turn off the EZRack PLC.
6. Insert the USB into EZRack PLC.
7. Apply power to the EZRack PLC. Upon power up the PLC project will be transferred to the PLC.

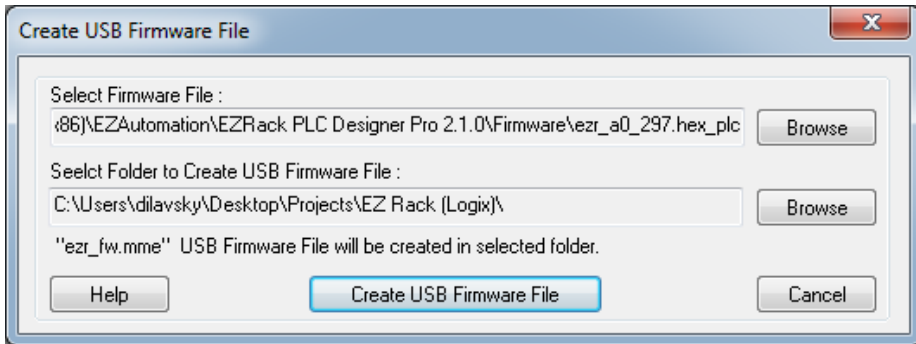
Note: Only 1 file can exist on the USB at a time to be loaded to the PLC.

Create USB Firmware File

There may be occasional upgrades to your EZRack PLC internal software, also referred to as the Exec or Firmware. This can be done over USB drive. Firmware can also be updated over other communication, please see Upgrade Firmware section. To do so please follow the direction below.

To Upgrade Firmware (Creating USB Firmware File and Upgrading EZRack PLC):

1. Back up the user program currently stored in the PLC and save to disk. Firmware upgrade will clear the project on the PLC.
2. Go to File > Create USB Firmware File. After clicking you get the following dialog.



3. Use the first Browse to navigate (click the on Browse button) to the new firmware file (.hex_plc file).
4. Use the second Browse to navigate to the USB drive. *Note: This file should go into the main folder of the USB drive, please do not hide it within sub-folders.*
5. Once both Firmware file and Create location have been selected please click the Create USB Firmware File.
6. The USB should now have a file ezr_fw.mme file which is your firmware file. *Note: The USB should not have a USB Loader file (avg.plc file). If it does please delete this file.*

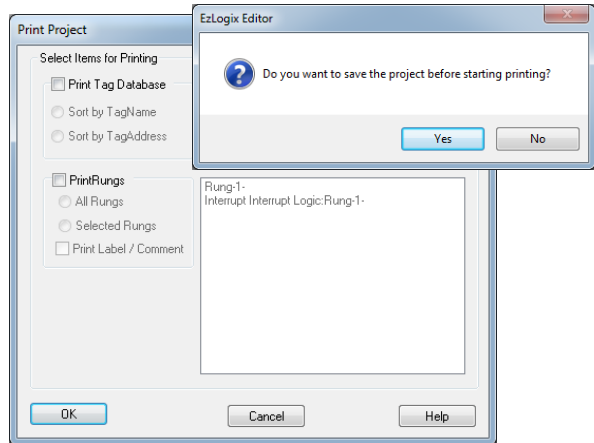
How to use USB to Upgrade EZRack PLC

7. Insert the USB with the file `eze_fw.mme` on it into the EZRack PLC. *Note: This will clear the PLC project.*
8. Wait, the EZRack PLC will start alternatively flashing the red and yellow LED while it is upgrading the firmware.
9. Once the firmware upgrade is finished the PLC will automatically restart.
10. You will now have to download the project. Or you can use a different USB to load the project back on the PLC.

Print

When you click on the Print menu item, you will be asked if you want to save the project. Click on Yes or No. Once you click Yes or No it will take you to another screen as follows:

Using this screen, you can choose if you want to print the Tag Database or the Rungs in the ladder logic program



Print Setup








Choose or change your print settings here.

Exit

Click on Exit to quit the program.

2.5.2 Edit Menu

When you click onto the Edit Menu, you can access the following functions:

	Undo	Ctrl+Z
	Redo	Ctrl+Y
	Cut	Ctrl+X
	Copy	Ctrl+C
	Paste	Ctrl+V
	Select All	Ctrl+A
	Delete	Del
	Edit	
	Toolbars	
	Default Tag Datatype...	
	Go to Rung...	Ctrl+R
	Go to Label...	Ctrl+L
	Show Abbreviated Tag Names	
<input checked="" type="checkbox"/>	Show Address on Instruction	
	Address Suggestion Preference...	

Undo / Redo

The Undo command is used to reverse the previous action. This function must be performed next in order for the action to be undone. The undo command goes back sixteen levels of undo. Redo will “redo” the previously undone action.

Cut

This allows you to Cut (remove) a selected item(s) to the clipboard.

Copy

This allows you to Copy (without removing) a selected item(s) to the clipboard.

Paste

This allows you to Paste a selected item from the clipboard onto the displayed screen.

Select All

Click on Select All to select all items on the displayed screen.

Delete

Click on Delete to remove a selected item without placing it on the clipboard.

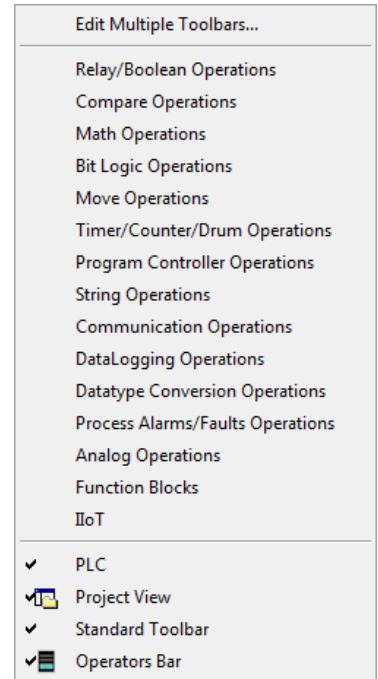
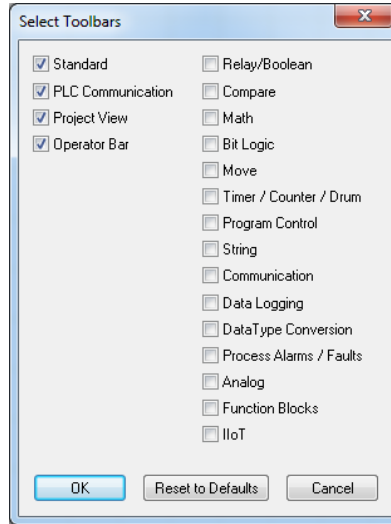
Edit

Select an object and then click on the Edit command to make changes to an object's / instruction's characteristics.

Toolbars

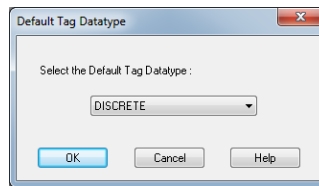
Click on Toolbars to see the available menu where you can click on the desired toolbars to be displayed on the toolbar section of the main screen.

When **Edit Multiple Toolbars...** is selected, it further takes you to the following screen which allows you to hide/select multiple toolbars from one dialog box. You can also restore the default toolbar configuration [here](#).



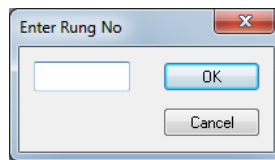
Default Tag Data Type

Default Tag Data Type allows you select the default Tag (memory location of PLC) type. Every time a new Tag is added after this, it will have the default type as chosen by this command through the following screen:



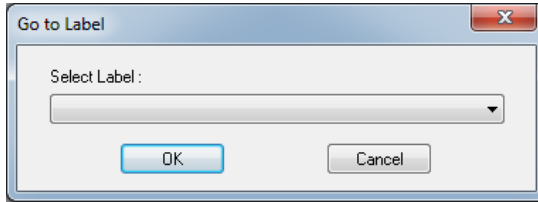
Go to Rung...

Use this option for a convenient way to quickly navigate to the desired rung. The menu will open the following dialog box:



Go to Label...

Use this option to go the specified label as shown in the following screen:



Show Address on Instruction

The EZRack PLC can show the address of the Tag or not show based on this selection. If this is selected the tag address is displayed otherwise it is hidden.

For Example:



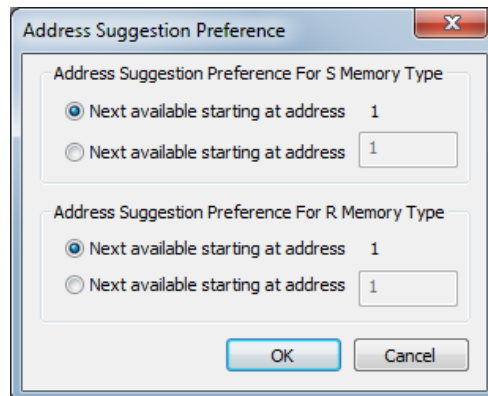
Tag Name and Address



Tag Name Only

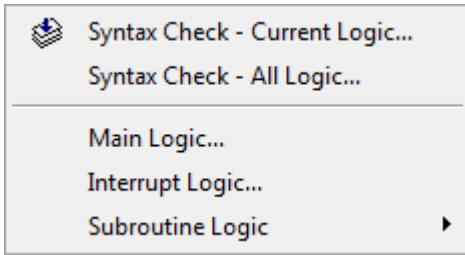
Address Suggestion Preferences

The EZRack PLC will suggest addresses based on the select starting point here. Initially it will start from Address 1 and suggest the next available address. The user can at any point tell the software to start suggesting from a different starting point. For example if starting address is set to 200 then if 200 is available that will be the next suggested address, if it is not then it will go to 201 and so and so forth.



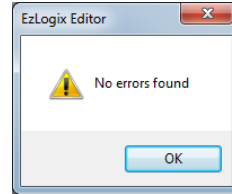
2.5.3 View Menu

When you click onto the View Menu, you can access the following functions:

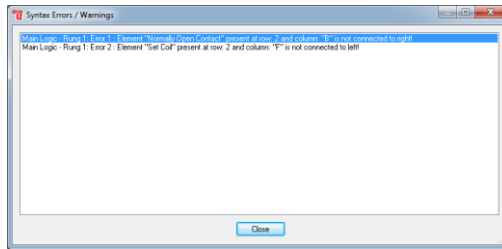


Syntax Check – Current Logic...

Use this option to display any errors present in the logic currently being displayed on the screen. Once selected, it displays the following message if NO errors are found.



If there are errors in the current logic, then it displays a similar screen as below with all the errors present along with their position (e.g. rung etc.):



Syntax Check – All Logic...

When using Syntax Check – All Logic, EZRack PLC Designer Pro checks the entire ladder logic program and displays the errors if found as shown above for Syntax Check – Current Logic.

Main Logic...

Click on this option to display the Main Logic in the Main Window of EZRack PLC Designer Pro when Interrupt or Subroutine logic is present in the Main Window. Main Logic, as the name suggests, is the main logic of your control program.

You can place some of the functions as Subroutine Logic, which is then called from main logic. You may want to use Subroutine to write some logic once and use at many places in your main logic (by calling it), or just to organize your main logic in modules.

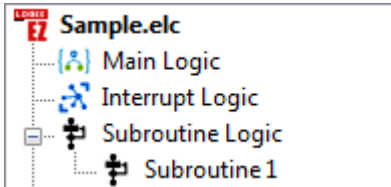
The interrupt logic is a special logic section, which is executed when an external interrupt occurs. The purpose of interrupt logic is to provide a fast response to some time-critical events. You will need to use the Interrupt input module to trigger execution of Interrupt logic.

Interrupt Logic...

Click on this option to display Interrupt Logic in the Main Window of EZRack PLC Designer Pro.

Subroutine Logic

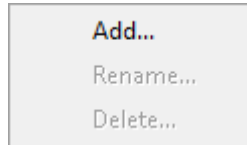
Click on this to display the Subroutine Logic in the Main Window of the EZRack PLC Designer Pro.



Note: To display and switch between Main Logic, Interrupt Logic, and Subroutine Logic, you can also use the Project Window to display the appropriate logic in the Main Window.

2.5.4 Subroutine Menu

Subroutines have two main uses. One, you can write some commonly used functions once, and use those multiple times within the main logic by calling the subroutine. Second, you can use subroutines to write modular logic. You can have multiple subroutines within a project. You can call a subroutine from another subroutine. Such nested calls cannot exceed 8 levels deep. Total maximum number of subroutines is 64. When you click onto the Subroutine Menu, you can access the following functions:



Add

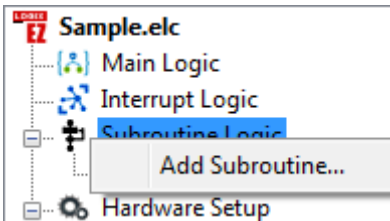
Use this function to add a Subroutine.

Rename

This function can be used to rename a Subroutine.

Delete

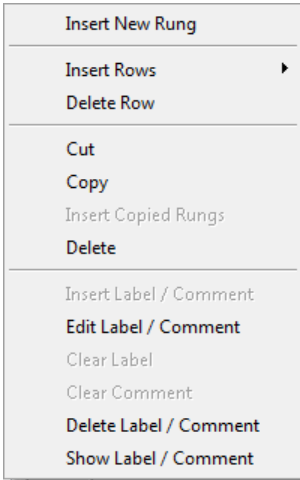
This function can be used to delete an existing subroutine.



Note: Subroutines can also be added by right clicking on Subroutine in the Project Window.

2.5.5 Rung Menu

When you click onto the Rung Menu, you can access the following functions:

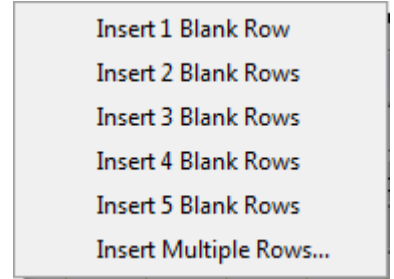


Insert New Rung

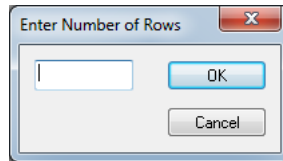
Click on this to add a new rung to the main ladder logic program.

Insert Rows

This function is used to add single or multiple rows within a rung. In order to use this function, first select the RUNG in which you wish to add single or multiple rows. Then click on the sub menu as follows to add the appropriate number of lines within a RUNG.

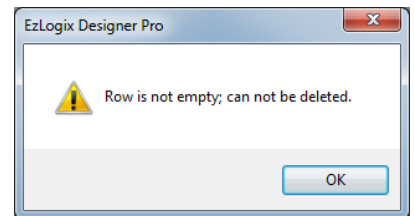


If you click on **Insert Multiple Rows...**, the following screen will appear and will require the number the rows to be added within a RUNG.



Delete Row

Use this function to delete excessive rows from a RUNG. You must select a RUNG from where a row is to be deleted. If Logic exists on the row being deleted, it will prompt you with the following message:



Cut

Use this function for cut and paste functions for RUNGS present in ladder logic. Before you apply this function you must select the desired RUNG which is to be Cut. Once RUNGS are cut using this function, they can be pasted into the desired location using the Insert Copied Rungs function.

Copy

This function is used to Copy the selected Rungs present in ladder logic. Once copied they can be pasted using the Insert Copied Rungs function.

Insert Copied Rungs

This function is used to paste RUNGS that have been Cut or Copied using the Cut and Copy functions in the Rung menu.

Delete

Use this function to delete the selected rungs from ladder logic.

Insert Label / Comment

Use this function to insert Label/Comment for a RUNG whose label and or comment were deleted using the Delete Label / Comment function.

Edit Label / Comment

Use this function to add Labels and Comments for individual rungs. Labels are useful when using the Jump instruction, which allows you to skip RUNGS and go to the one specified in Jump instruction.

Clear Label

Can be used to clear label of an individual RUNG.

Clear Comment

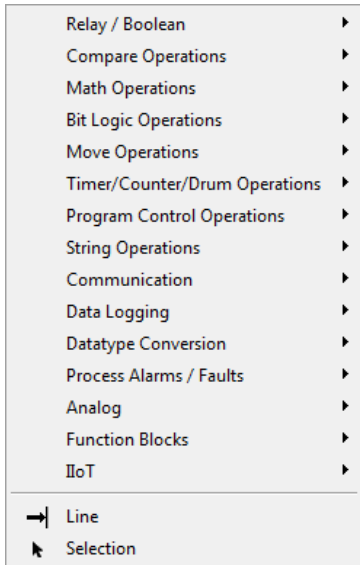
Can be used to clear the comments of an individual RUNG.

Delete Label / Comment

This function can be used to delete a Label and Comment for an individual RUNG. When deleted, a rung can be re-labeled / commented by using the Insert Label / Comment function in Rung Menu > Insert Label/Comment.

Show Label / Comment

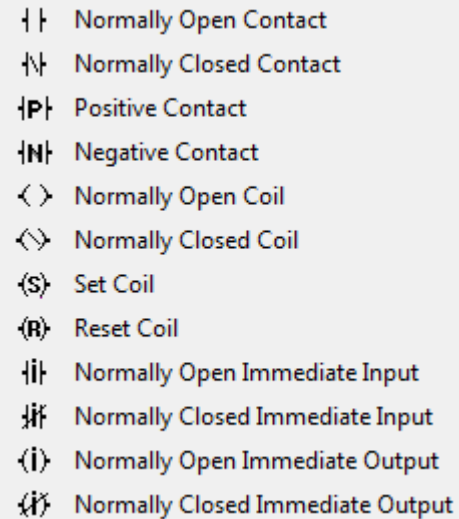
Use this function to Hide or Display the Labels and Comments for all the RUNGS present in the ladder logic program.



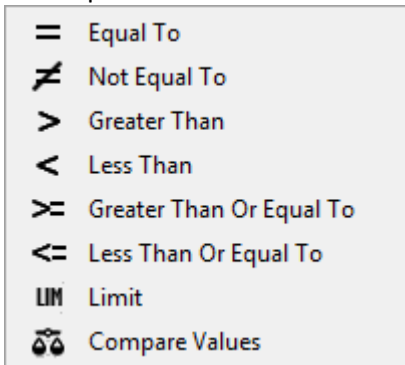
2.5.6 Instructions Menu

All the Instructions used for Relay Ladder Logic are explained in detail in *Chapter 3 - RLL Instructions*. When clicked on appropriate Instruction, the EZ Rack PLC Designer Pro allows you to place that instruction in ladder logic by clicking in the Main Programming window.

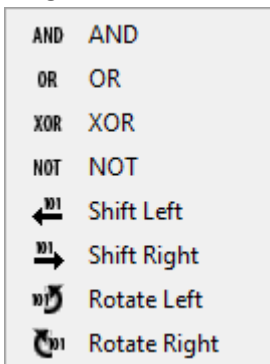
Relay/Boolean Instructions Menu



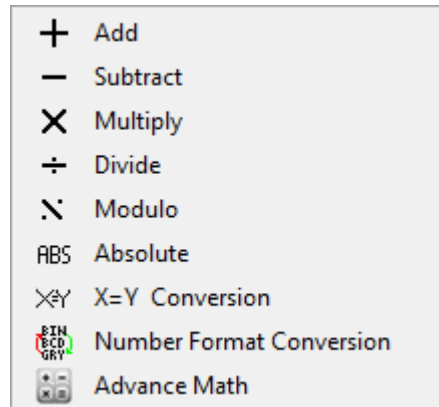
Compare Instructions Menu



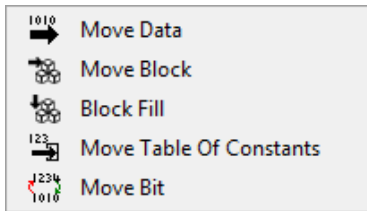
Bit Logic Instructions Menu



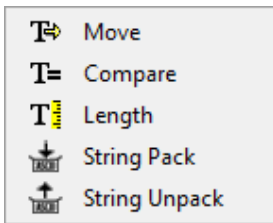
Math Instructions Menu



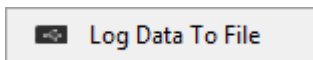
Move Instructions Menu



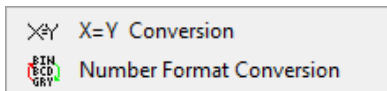
String Operations Menu



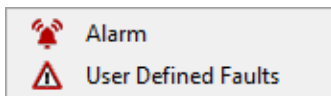
Data Logging Instructions Menu



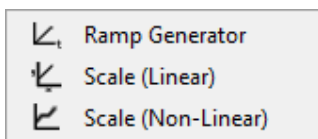
Datatype Conversion Instructions Menu



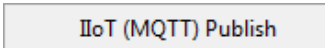
Process Alarms/Faults Instructions Menu



Analog Instructions Menu



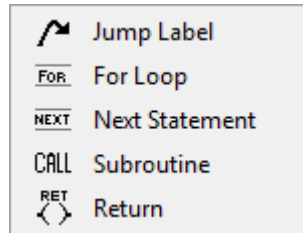
IIoT Instructions Menu



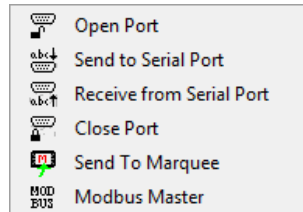
Timer/Counter/Drum Instructions Menu



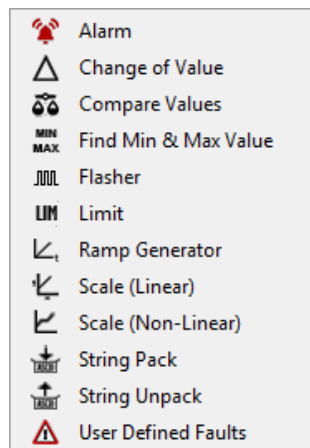
Program Control Instructions Menu



Communication Instructions Menu



Function Block Instructions Menu

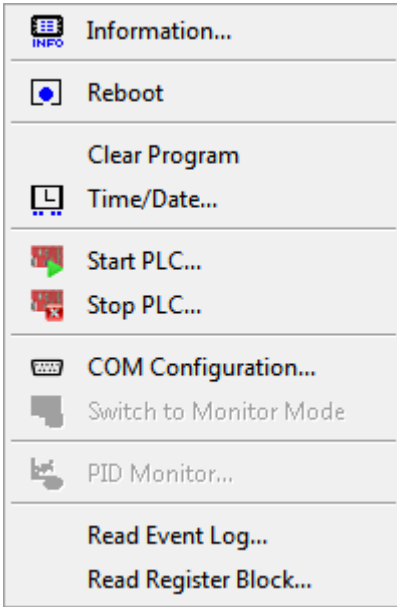


Line

The Line Tool allows you to connect instructions and objects in the ladder logic.

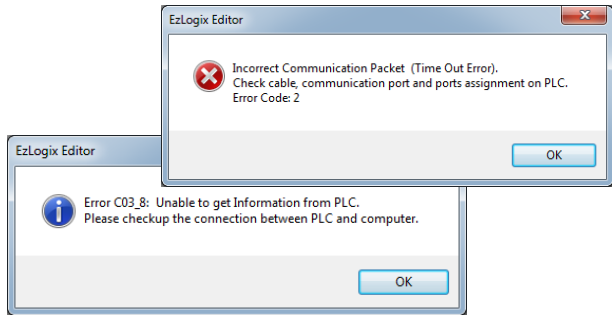
Selection

Click on this function to switch back to the Selection Tool from the Line Tool



2.5.7 PLC Menu

The PLC menu allows you to access and control the EZRack PLC. In order to utilize most of the functions present in this menu, EZAutomation's EZRack PLC must be connected to the programming PC. If the corresponding PLC is not connected to the programming PC, the following error messages will appear:

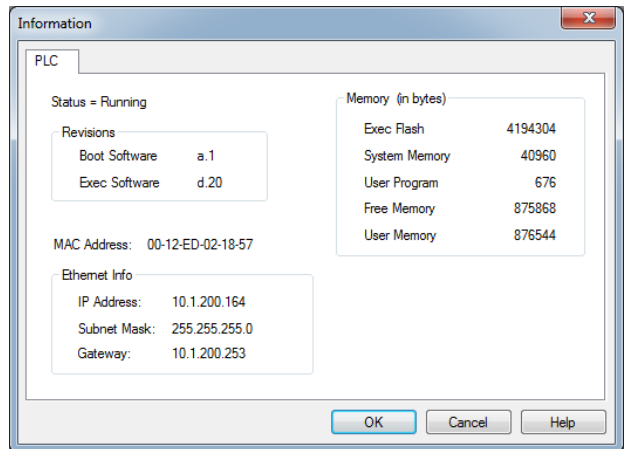


Note: For all functions in the PLC Menu, upon communication timeout EZRack PLC Designer Pro will inform you that the function could not be completed due to a communication issue.

Information

Click on this function to acquire information for the PLC connected to the programming PC.

When connected, it displays information regarding Status, Firmware Revisions, MAC Address, IP address and Memory used as shown in the following screen:

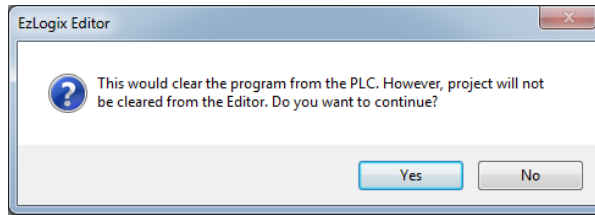


Reboot

Use this function to perform a Warm Reboot of the corresponding PLC while connected to the programming PC as shown in the following screen:

Clear Program

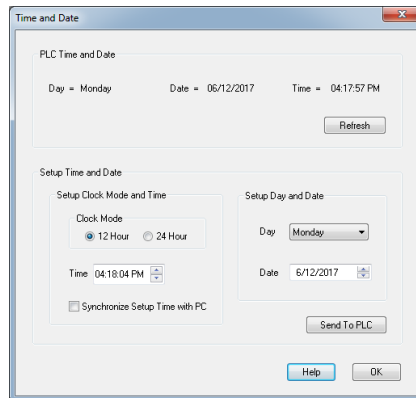
This function is used to clear the existing program present in the memory of the PLC. When used, it will prompt for confirmation as follows:



Warning: If you select Yes, the corresponding PLC's program will be cleared!

Time/Date

This function can be used to Monitor and Change the current Time and Date settings on a PLC. The PLC's clock can be set for either 24-hour or 12-hour along with the option to synchronize the PLC's clock with the clock of the programming PC as shown below:



Start PLC

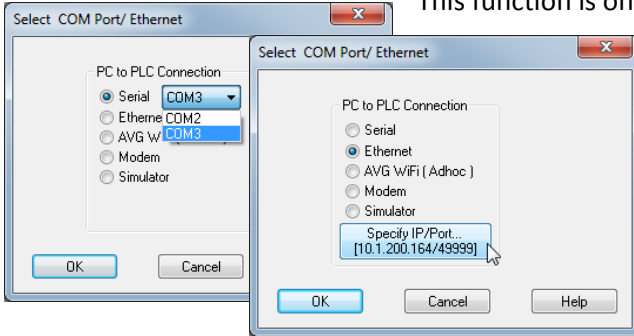
Use this function to Start the corresponding PLC into RUN mode.

Stop PLC

Use this function to Stop the corresponding PLC when present in RUN/Program mode.

COM Configuration...

This function is only available when programming in Offline mode. Use this function to select the COM port / Ethernet's IP address of your PLC based on how you are connecting i.e. via the COM port or Ethernet port. Please see the Communications Options Table for more information. The following screen appears when this function is selected:



- Serial Communication refers to communication through serial cable (EZ-PGMCBL see Appendix 1) and through Micro-USB cable.
- For Ethernet communication, please enter the IP address of your corresponding PLC.
- The AVG WiFi option will automatically detect any EZ-WiFi module and then it can be used exactly like the Ethernet and Serial communication options.

Communication Options Table

Communication Type	PC to PLC Connection Option	Functionality				
		Transfer to PLC	Read from PLC	Monitor PLC (Online)	Debug PLC (Online)	Upgrade Firmware
Serial Cable (EZ-PGMCBL)	Serial (Select Com Port)	✓	✓	✓	✓	✓
Micro-USB Cable	Serial (Select Com Port)	✓	✓	✓	✓	✓*
Ethernet	Ethernet (Input IP Address)	✓	✓	✓	✓	✓*
EZ-WiFi	AVG WiFi	✓	✓	✓	✓	✓*
USB Drive	USB Drive (Create USB file)	✓	✗	✗	✗	✓*

✓* Note: Upgrading using these methods requires the EZRack Editor v2.1 or higher and the firmware on the EZRack PLC to be vA.0.297 or higher. If try to upgrade from vA.0.256 or lower these methods will not work.

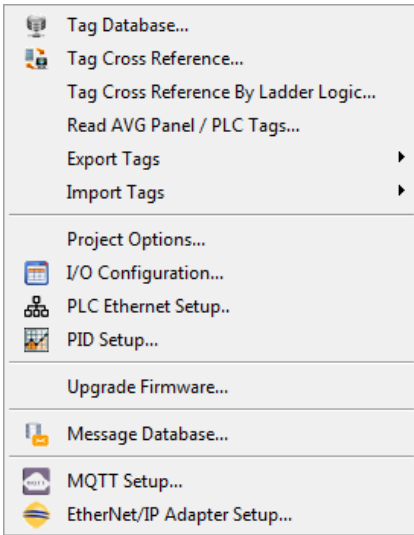
Switch Monitor Mode / Switch to Edit Mode

Click on this function to switch between Monitor and Edit mode when connected to the corresponding PLC. Please see more information on Online Edit Mode and Monitor Mode in Chapter 4.

Note: Monitor mode will not allow any editing of the ladder logic program present in the PLC.

PID Monitor...

Click on this to open the dialog box for the PID Monitor function. Please see more information in Chapter 6 – PID.



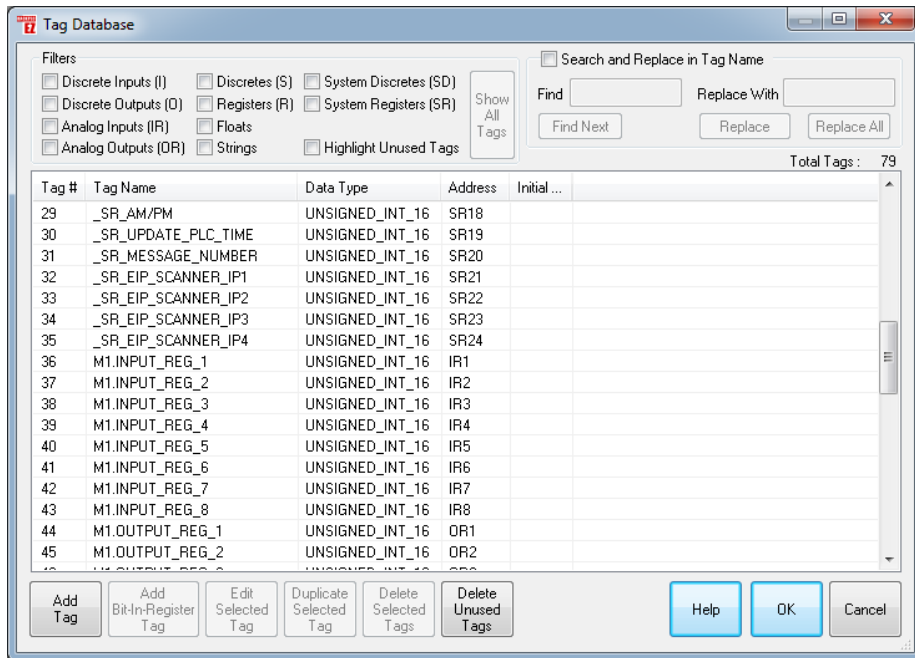
2.5.8 Setup Menu

When you click onto the Setup Menu, you can access the following functions:

Tag Database...

The EZ Rack PLC Designer Pro will auto fill tag addresses for you. This functionality is explored more in section 3.3.2 Auto Generated / Auto Fill Tags.

Tag Database allows you to view and add/edit current and new Tags (memory addresses) of your corresponding PLC. Click on this function to display the following screen:



As shown the Tag Database will display all the Tags that have already been entered for your PLC. The tag database will be prepopulated with the PLC system tags and any output / input module tags for any module that has been configured in your I/O setup. For more information please see the I/O Configuration section.

Add Tag

Clicking onto the **Add Tag button** will display this screen:

As shown in the Add New Tag Details screen, you can add the Tag Name and change the Tag Address. Based on the tag datatype the Memory type will change, also the Tag address will be preselected for you but you can change it. For more information please see the section 3.3.2 Auto Generated / Auto Fill Tags. You can also specify the Initial Value of the Tag that is being added, where the Initial Value is the value that the PLC assumes for this particular Tag when using it the first time.

Tag Name: Every memory address (Tag) can also be assigned a Tag Name which is used while programming the PLC. E.g. "O1" is the memory location of physical output 1 present on an IO module. The tag name of prepopulated tags can also be changed. For I/O module tags the tag name has a specific format of M#.Tag_Name where # refers to the module position. The tag name is then used in all instructions to select this tag. Other than in specified cases "." cannot be used. Refer chart below for information on prepopulated tags and their format.

Address Suggestion Preference

The EZRack PLC will suggest addresses based on the select starting point here. Initially it will start from Address 1 and suggest the next available address. The user can at any point tell the software to start suggesting from a different starting point. For example if starting address is set to 200 then if 200 is available that will be the next suggested address, if it is not then it will go to 201 and so and so forth.

Tag DataType: The tag datatype will determine the Memory Type and how many address it will consume in memory. Discrete datatype will use S memory type, everything else will use R memory type. I/O tags are automatically generated based on the select I/O modules in the I/O Configuration.

Address: Address is used to specify the actual Tag (memory address) of the corresponding PLC. For example, O1 refers to Output 1, R1 refers to internal register 1, SR1 refers to system register 1 etc. Please see section 3.3.1 Memory Map for more information on address ranges and specifications.

Initial Value: When initial value is enabled you can enter the value that the tag will start with upon program load. All tags are retentive so upon power cycle the tag will retain the last value it was set to. *Note: Retentive tags are battery backed therefore please make sure your battery has power.*

Example:

As described above, the following screen is an example of adding a valid Tag whose Address is S1, Data Type is Discrete, and the Tag Name is Light1. Also, the Initial Value has been assigned to be ON or "1".

When you have entered your Tag information, click on Apply Changes to add the new Tag created by Edit Tag Details. Click Close to return back to the main Tag Database screen. The # of Characters can only be specified

when using an ASCII type Data Type for a word register and the maximum number of characters is 126. The main Tag Database screen also offers features for easy handling of Tags entered in the database. The section below describes all the functions available in this screen.

Selection Menu



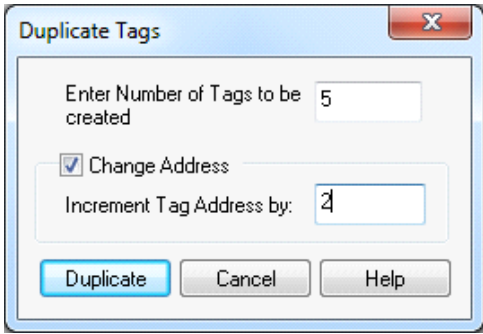
You can either use the buttons in the bottom row or you can right click on the tag to access the selection menu.

Add Tag: Use this to add new tags to the database.

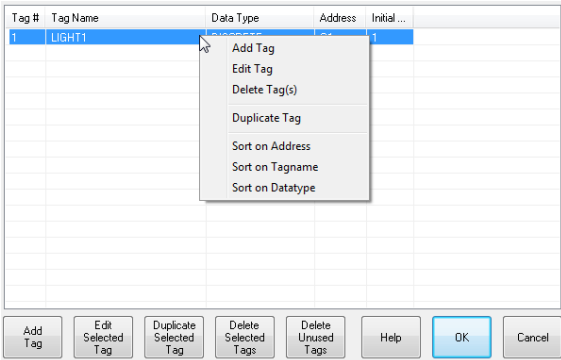
Add Bit-In-Register Tag: This will add bit within words tag to the tag database. Will only function for the first 16 bits of a word, please select the needed word before pressing the tag.

Edit Selected Tag: Will bring up Edit Tag Details screen and you can change the Tag Name, Tag Data Type, Address and Initial Value for the Tag.

Duplicate Selected Tag: This option will bring up the duplicate tags dialog. Using this the tag can be duplicated either with the same address or incrementing the address by a set amount.



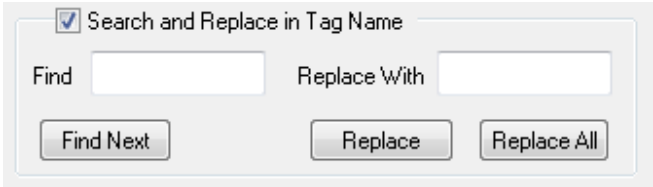
Delete Selected Tag: This will delete the selected tag.



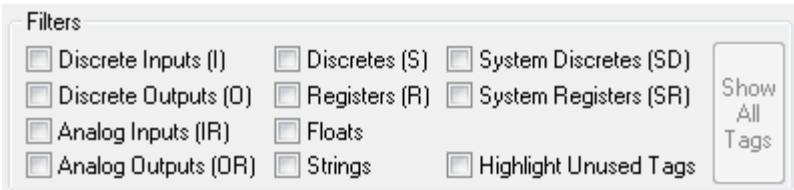
When you select a tag in the tag database you will have the following options: Edit Selected Tag, Duplicate Selected Tag, and Delete Selected Tag.

Search and Replace

The **Search and Replace** function can be used to find and replace Tags present in the Tag Database. This allows you to easily make changes to Tags previously entered in the database.



Filters



Filters allow you to only display certain tags in the tag database. Use the **Show All Tags** to go back to default view with all tags visible.

Discrete Inputs: Will only show (I) discrete inputs.

Discrete Outputs: Will only show (O) discrete outputs.

Analog Inputs: Will only show (IR) analog inputs.

Analog Outputs: Will only show (OR) analog outputs.

Discretes: Will only show (S) internal discretes.

Registers: Will only show (R) internal registers. Includes bit within words for internal registers.

Floats: Will only show floating point tags.

Strings: Will only show string tags.

System Discretes: Will show all System Discretes.

System Registers: Will show all System Registers.

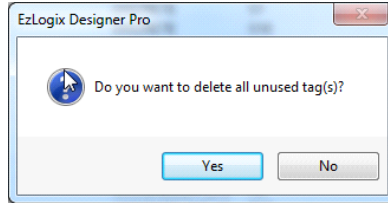
Highlight Unused Tags

When this check box is checked, it will highlight all the tags in the PLC database which are not being utilized by an instruction. When checked it will highlight unused Tags as follows:

Tag #	Tag Name	Data Type	Address	Initial ...
1	LIGHT1	DISCRETE	O1	1
2	ON	DISCRETE	S1	
3	CLOSE	DISCRETE	S10	
4	TAG_1	UNSIGNED_INT_16	R2	
5	TAG_2	UNSIGNED_INT_16	R3	
6	TAG_3	UNSIGNED_INT_16	R4	
7	TAG_4	UNSIGNED_INT_16	R5	
8	TAG_5	UNSIGNED_INT_16	R6	
9	TAG_6	UNSIGNED_INT_16	R7	
10	TAG_7	UNSIGNED_INT_16	R8	
11	TAG_8	UNSIGNED_INT_16	R9	
12	TAG_9	UNSIGNED_INT_16	R10	
13	TAG_10	UNSIGNED_INT_16	R11	
14	STATUS	UNSIGNED_INT_16	R20	
15	SIZE	UNSIGNED_INT_32	R40	
16	SR7	UNSIGNED_INT_16	SR7	

As shown in the above screen, all the Tags not being utilized anywhere in the PLC program will be highlighted.

You can also delete these Tags which might be present in access by clicking on **Delete Unused Tags** which will prompt the following message:



If YES is selected, the corresponding Unused Tags in the database will be deleted.

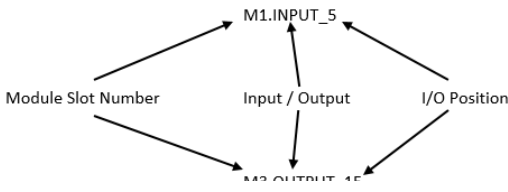
Table of Tag Formats

This table describes all the different possible tag formats that exist. While it may look complicated most of these are auto created for you so the user only needs to create the basic tags. *Note: All tags can have their name modified except for system and reserved tags.*

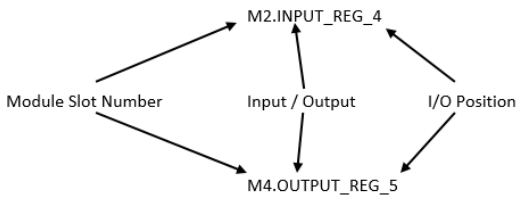
Tag Name Structure	Possible Data Types	Description	How Created	Example
TAG_NAME	Discrete or Register	This is the basic tag the user can create and modify the address of.	User Created.	LIGHT, VALUE1
TAG_NAME.#	Discrete (Bit within word)	Bit access to the register. Works with Registers only.	User Created.	VALUE1.0
TAG_NAME.RES#	Register	Reserved tags for internal calculations.	Auto Created for certain function blocks.	VALUE1.RES1
M#.INPUT_#	Discrete	Module # Inputs. User can changed name.	Created during I/O Config.	M1.INPUT_1
M#.OUTPUT_#	Discrete	Module # Outputs. User can changed name.	Created during I/O Config.	M3.OUTPUT_7
M#.INPUT_REG_#	Discrete	Module # Input Registers. User can changed name.	Created during I/O Config.	M7.INPUT_REG_3
M#.OUTPUT_REG_#	Discrete	Module # Output Registers. User can changed name.	Created during I/O Config.	M5.OUTPUT_REG_5
M#.TAG_NAME	Discrete or Register	Specialty module # tags. User can changed name.	Created during I/O Config.	M4.CNTR1_COUNTS
_SD_TAG_NAME	Discrete	System discrete.	Always in project.	_SD_FIRST_SCAN
_SR_TAG_NAME	Register	System register.	Always in project.	_SR_MINUTES

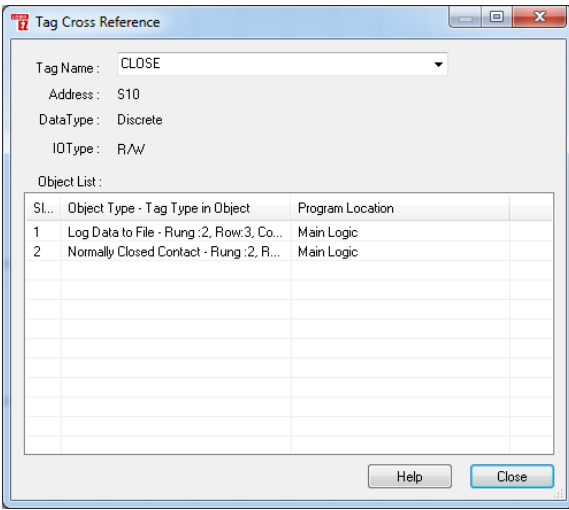
I/O Address Format for EZRack PLC

Bit



Register



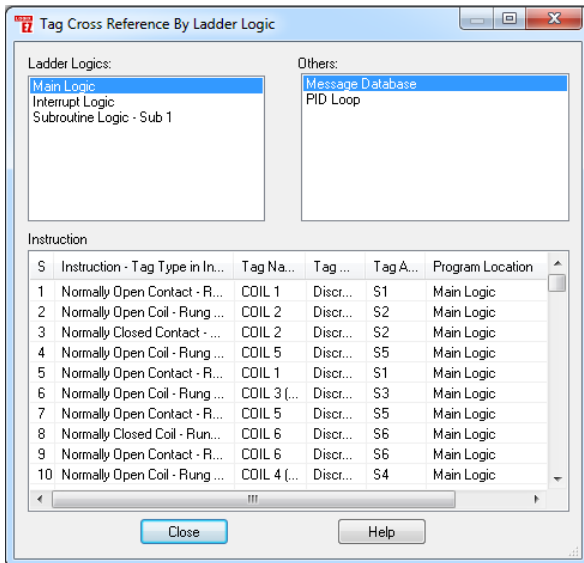


Tag Cross Reference...

This function is extremely useful for identifying the objects and instructions utilizing a certain PLC Tag in the ladder logic program as shown in the following screen:

As shown, Tag Cross Reference provides all the details where and how many times a certain Tag is used in the ladder logic program. In the example shown, the “CLOSE” discrete is being used in two instructions: The Log Data to File instruction is present in Rung 2, Row 3, Column 4 and the Normally Closed Contact instruction is present in Rung 2, Row 3, Column 1.

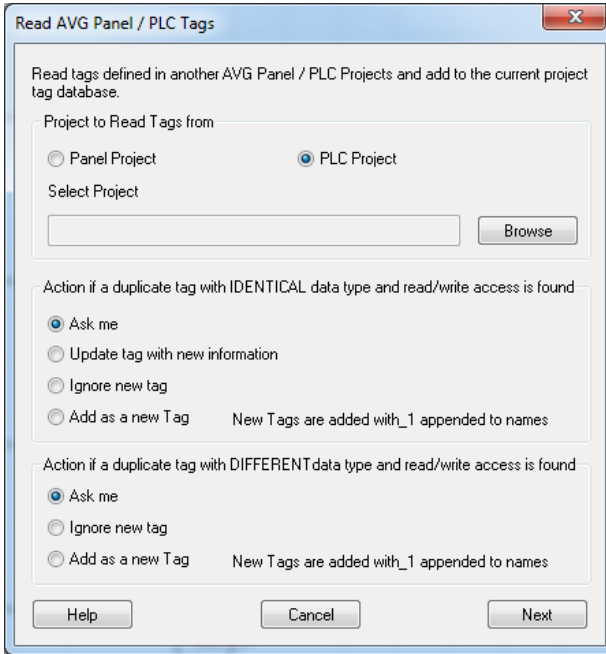
Tip: Use the Cross Reference function before changing the functionality of any Tag. This will allow you to figure out very quickly where and how many times that register is utilized in the ladder logic program.



Tag Cross Reference By Ladder Logic...

This function allows you to see all the tags used in specified locations. You can therefore cross reference the tags used in the message database or PID Loop with the ones used in your main logic, interrupt logic and subroutines.

Read AVG Panel/PLC Tags...



This function is particularly useful when using EZ Rack PLC with AVG Panels. Using this function, the Tag Database of a project can be very easily populated by automatically reading from a pre-existing Tag Database of an AVG Panel or another EZ Rack PLC project. Click on this function to display this screen.

As shown in the example screen, you can decide which pre-existing project is to be used for copying and adding the Tag information to the existing Tag Database. Click on the Browse button to select the directory and the name of the project to be utilized for copying the Tag information. As shown in the Read AVG Panel/PLC Tags... screen, you also have the option to select the action in case the Tag being

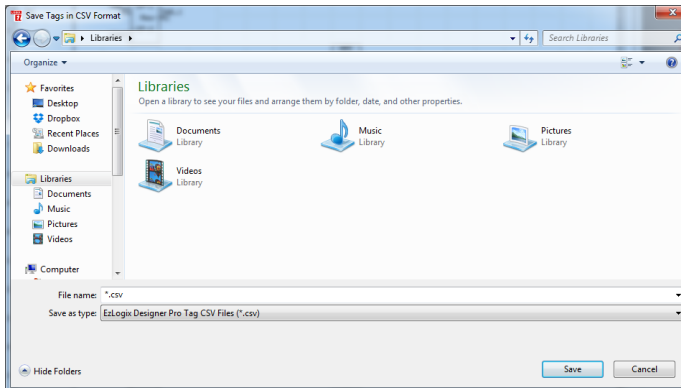
copied to the Tag Database already exists.

Export Tags

Click on this function to expand this menu as shown:

Comma delimited...
To Excel...

When **Comma delimited** is selected, this function exports all the Tags in the Tag Database into a CSV file as specified by the user in the following screen:



When **To Excel...** is selected, this function automatically opens Excel software on the programming computer and exports all the Tags in the Tag Database as shown below:

TAG NAME	TAG DATA ADDRESS	# OF CHAN	INITIAL V	RETENTIVE	PLC/INT	WRITE TAG
LIGHT1	DISCRETE O1		0	NO	PLC1	
ON	DISCRETE S1				PLC1	
CLOSE	DISCRETE S10				PLC1	
TAG_1	UNSIGNECR2				PLC1	
TAG_2	UNSIGNECR3				PLC1	
TAG_3	UNSIGNECR4				PLC1	
TAG_4	UNSIGNECR5				PLC1	
TAG_5	UNSIGNECR6				PLC1	
TAG_6	UNSIGNECR7				PLC1	
TAG_7	UNSIGNECR8				PLC1	
TAG_8	UNSIGNECR9				PLC1	
TAG_9	UNSIGNECR10				PLC1	
TAG_10	UNSIGNECR11				PLC1	
STATUS	UNSIGNECR20				PLC1	
SIZE	UNSIGNECR40				PLC1	
SR7	UNSIGNELSR7				PLC1	

Note: When using the “To Excel...” function, the EZRack PLC Designer Pro software automatically opens a new “Book1.xls” file and exports the Tag information. Also the file is NOT saved on the hard drive unless you manually save it. You must have Microsoft Excel software installed on your computer to utilize this function.

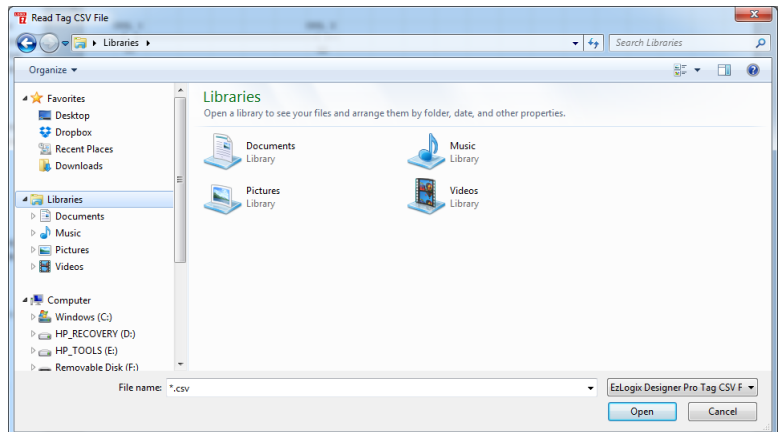
Import Tags

Click on this function to expand this menu as shown:

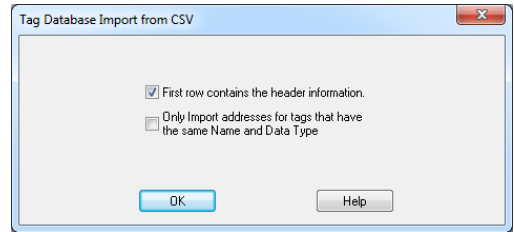
Comma delimited...

Excel format...

When **Comma delimited** is selected, this function imports all the Tags present in a CSV file as specified by the user in the following screen:



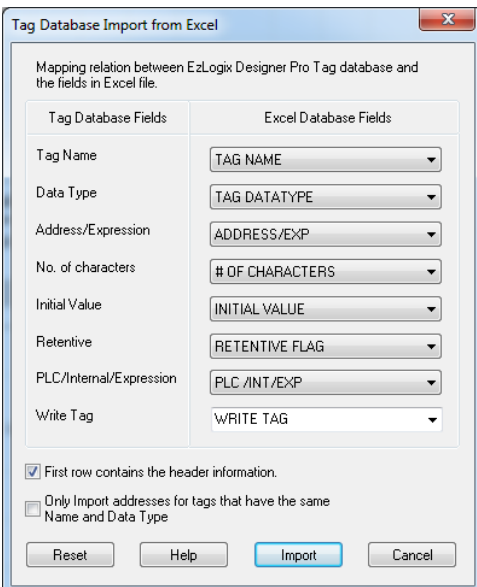
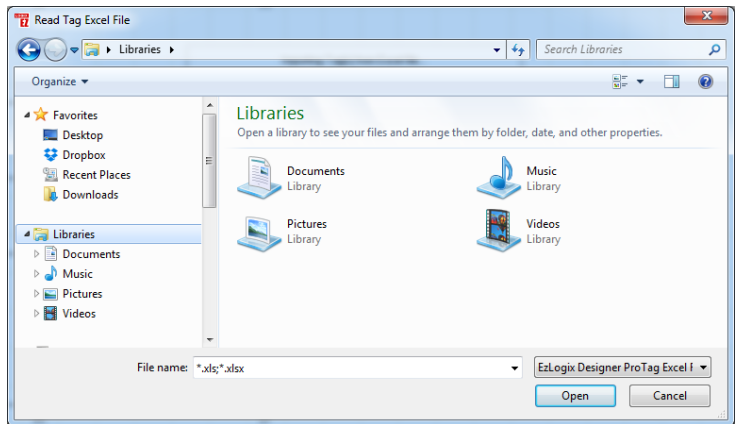
Once the user has selected a CSV file which is to be imported, the following screen will appear and prompt user as follows for more information:



Select first row contains header information if the csv file has header information like Tag Name, Tag Address, etc.

Select only Import address for tags that have the same Name and Data Type if you want to overwrite the tag address information of tags already in the project.

When **Excel format...** is selected, this function imports all the Tags present in an Excel file as specified by the user in the following screen:



Once the user has selected an Excel file which is to be imported, the following screen will appear and prompt user as follows for more information:

Please make sure to map the correct headers to the corresponding Tag database header.

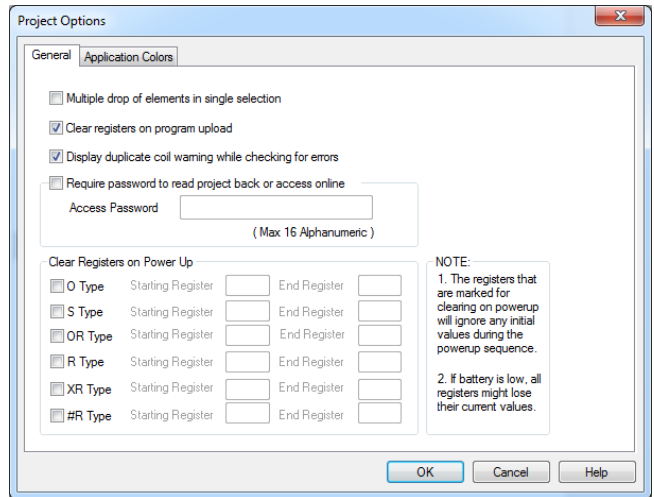
Select first row contains header information if the csv file has header information like Tag Name, Tag Address, etc.

Select only Import address for tags that have the same Name and Data Type if you want to overwrite the tag address information of tags already in the project.

Project Options...

Click on the Miscellaneous button to select options for Multiple Drop, Clearing registers on program upload, and Displaying information for warnings as shown in the following screen:

Multiple Drop: This option allows you to add multiple elements after selecting the element e.g. select NO Contact and put 5 of them in your ladder logic.

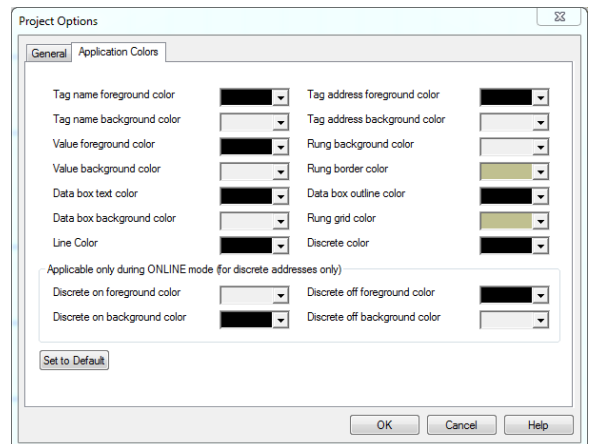


Clear Register on program upload: When the project is transferred to the PLC, this options means all the registers will be cleared and set to initial value of the transferred project (no initial value equals 0).

Display duplicate warning: EZRack PLC Designer Pro allows you to use coils with the same address in multiple places but unless done correctly this type of coding can cause problems. Therefore this option will warn you about these instances and if there is no need please use coils such that there are no such instances.

Password: To always require a password for project read back from PLC select this option and enter the password. *Note: There is no admin password therefore if you forget the password you cannot read back this project.*

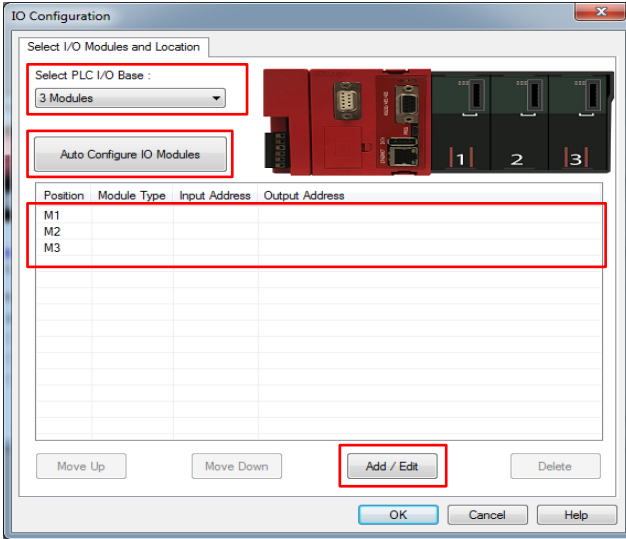
Clear registers: The EZRack PLC has retentive registers therefore upon power up the registers are set at the last known values. If you would like to clear registers upon power up you can use these options to select the specific range of registers to clear. *Note: Retentive tags are battery backed therefore please make sure your battery has power.*



The Application Color tab allows the user to edit the color and configuration for the Ladder Logic programming window as shown in the screen above.

I/O Configurations...

Click on this function to select the I/O configuration for your corresponding PLC as shown in the screen below:



The Module slot positions are identified as M1, M2, M3 etc., on the I/O base. The dialog box shows only the available module positions for the selected I/O base. For example, a 3-module base will show only M1-M3 positions, while a 7-slot base will display rows M1-M7.

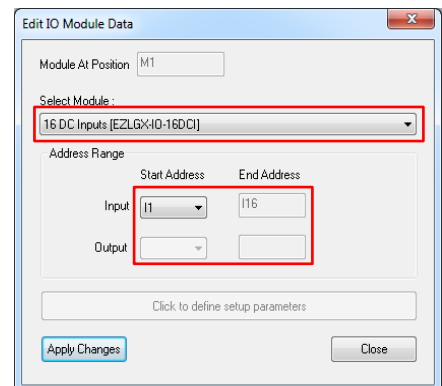
To **Manually Configure** a module on a position, double click the row corresponding to the position number (say M1) or click the **Add/Edit** button.

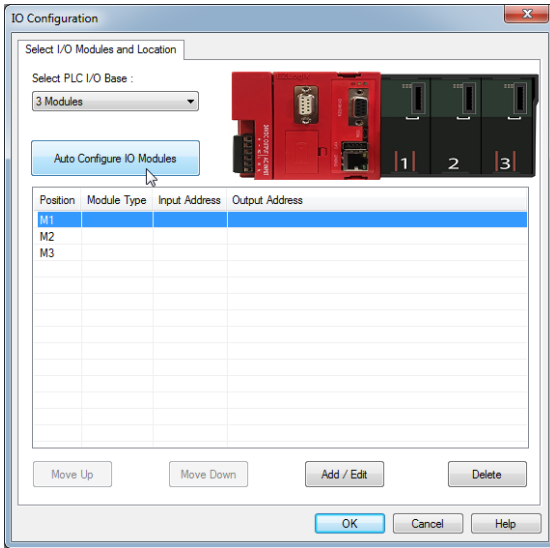
OR

You can also **Auto Configure** the IO Modules by clicking the “Auto Configure IO Modules” button.

Manual Configuration

After clicking the Add/Edit button you will get the Edit IO Module Data Dialogue Screen. Select the module type from the available modules and its I and/or O (IR or OR for analog) addresses from respective drop downs. You select the start address of the module, and the software computes and fills up the end address of the module automatically.

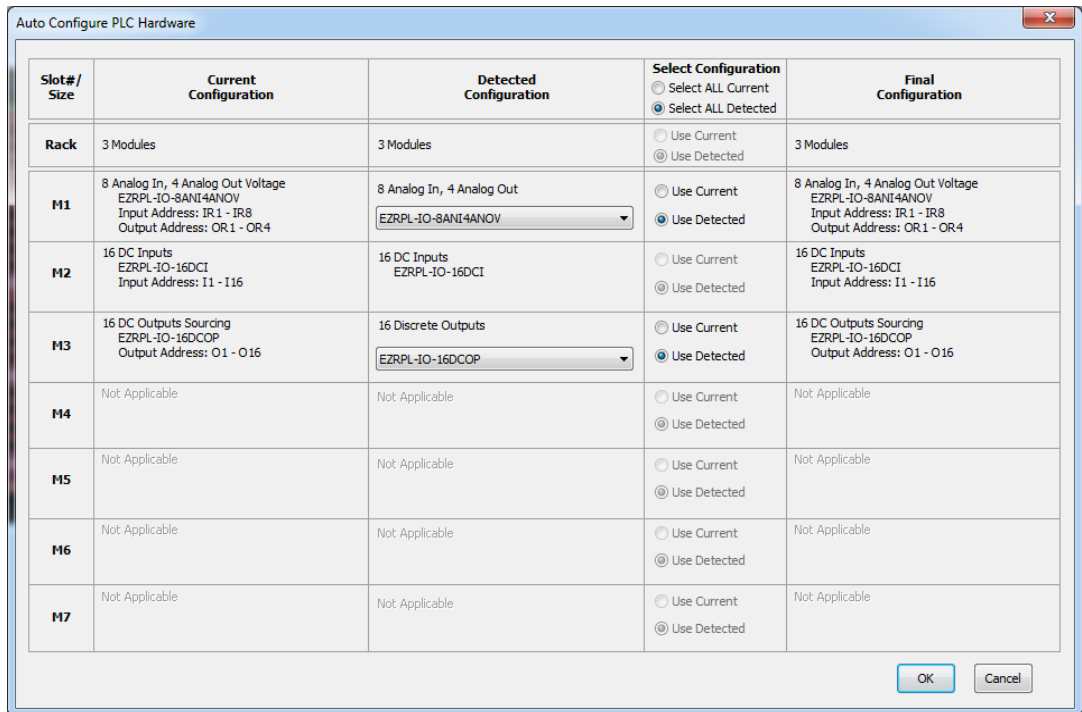




Auto Configure

For Auto Configure please make sure that the PC and PLC are connected (either serially, over micro-USB, or over Ethernet).

After clicking the “Auto Configure IO Modules” button you will get the following dialog.



The EZRack PLC Designer Pro will read back the PLCs configuration and the table above will display the results. It will also display the rack size detected in the configuration.

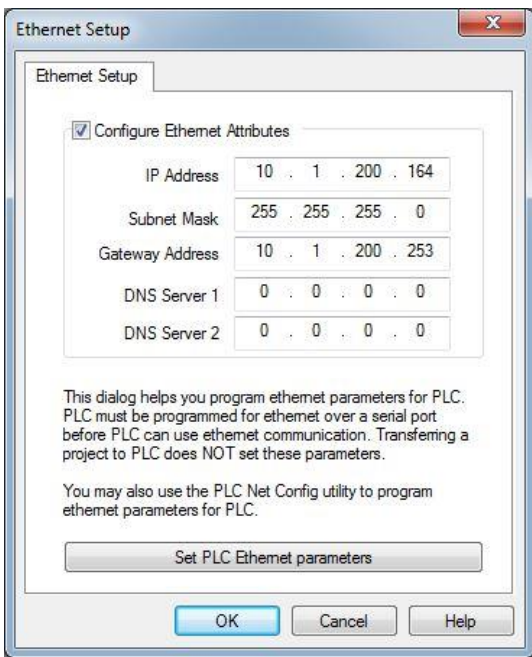
Current Configuration -- This the currently selected configuration in the EZRack software.

Detected Configuration -- This the detected configuration of the EZRack PLC.

Select Configuration -- Please select whether to use the current configuration in the software or the detected configuration. You can also select All in the section right under the title.

Final Configuration -- This will display the configuration which will be now configured in the software.

Please note: there are certain module families which include multiple modules of different types like "Sinking" or "Sourcing" for Digital Outputs and "Voltage" or "Current" for Analog modules cannot be differentiated. Therefore please make sure the correct module is selected under the detected configuration.



Ethernet Setup...

Select this function for defining the Ethernet settings for your corresponding PLC. When selected, the following screen will be shown.

As shown, you can specify the IP Address, Subnet Mask, and Gateway for the corresponding PLC. To set the PLC Ethernet parameters please use either serial cable, Micro-USB cable, or AVG WiFi.

You can also configure the EZRack PLC to use a DNS server. Please enter the IP address of the DNS servers here. If you do not want to use this option please leave them as 0.0.0.0.

Note: You cannot set Ethernet parameters over Ethernet.

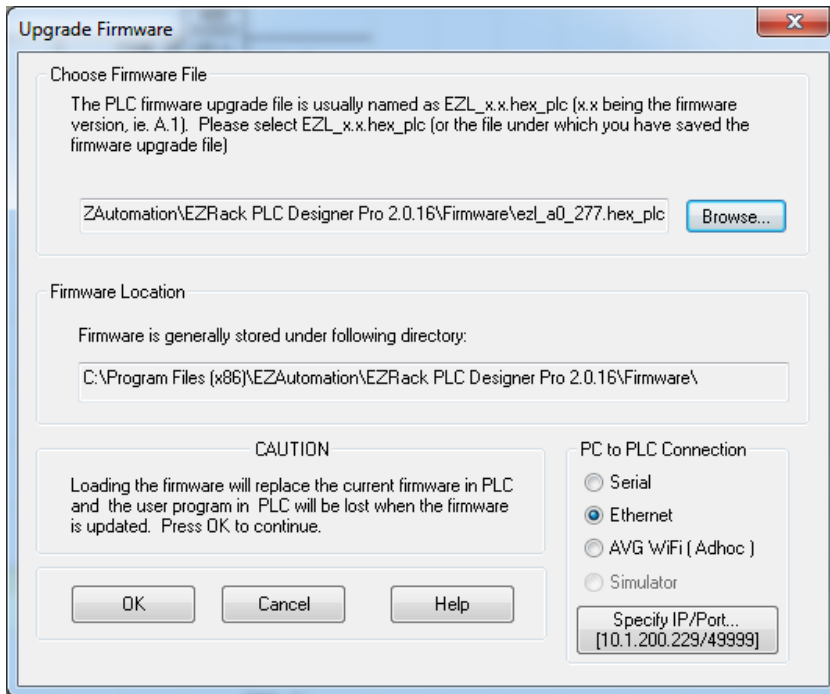
Note: Transferring a project to the PLC does not set these parameters.

PID...

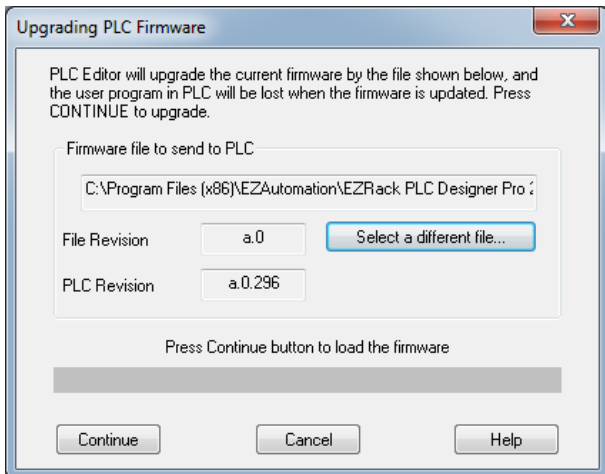
Click on this to open the PID Setup dialog box (explained in greater detail in *Chapter 6*).

Upgrade Firmware

There may be occasional upgrades to your EZRack PLC internal software, also referred to as the Exec or Firmware. (Check the EZAutomation website periodically for information about software and firmware upgrades.)



To Upgrade Firmware (Serial, Ethernet and Micro-USB):



1. Back up the user program currently stored in the PLC and save to disk. Firmware upgrade will clear the project on the PLC.
2. Click on Upgrade Firmware and navigate (click the on Browse button) to the new firmware file (.hex_plc file).
3. Select the appropriate communication option (please see table below) under PC to PLC Connection and click on the OK button to begin the upgrade.
4. If the PLC is running you will be asked to stop the PLC. If the PLC is not stopped the firmware upgrade will not happen.

5. On the next screen click continue to start the upgrade. The status bar will let you know when the upgrade is complete.

Note: For Ethernet and Micro-USB the firmware will first download then an upgrade will happen. You can cancel the upgrade only during download. When upgrade is happening on the PLC the red and yellow LED will alternate flashing.

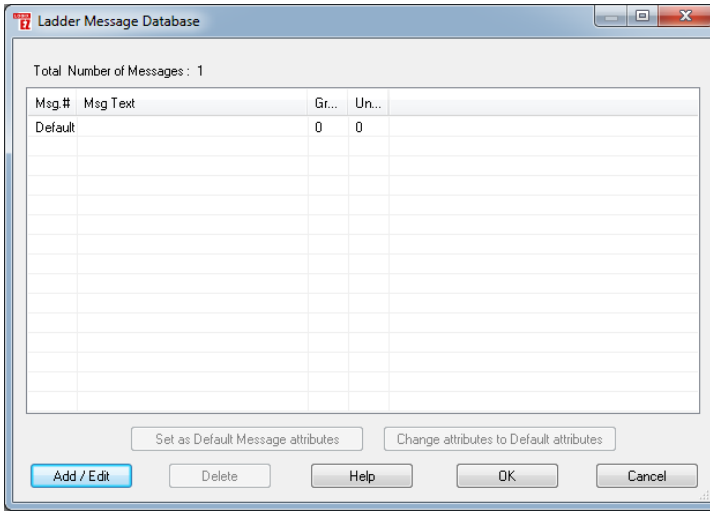
Note 2: If Ethernet or Micro-USB firmware upgrade fails then the firmware upgrade over a serial cable will always work to fix any corrupted firmware.

To Upgrade Firmware (USB): Please see the Create USB Firmware File section.

Communication Options Table

Communication Type	PC to PLC Connection Option	Functionality				
		Transfer to PLC	Read from PLC	Monitor PLC (Online)	Debug PLC (Online)	Upgrade Firmware
Serial Cable (EZ-PGM CBL)	Serial (Select Com Port)	✓	✓	✓	✓	✓
Micro-USB Cable	Serial (Select Com Port)	✓	✓	✓	✓	✓*
Ethernet	Ethernet (Input IP Address)	✓	✓	✓	✓	✓*
EZ-WiFi	AVG WiFi	✓	✓	✓	✓	✓*
USB Drive	USB Drive (Create USB file)	✓	✗	✗	✗	✓*

✓* Note: Upgrading using these methods requires the EZRack Editor v2.1 or higher and the firmware on the EZRack PLC to be vA.0.297 or higher. If try to upgrade from vA.0.256 or lower these methods will not work.



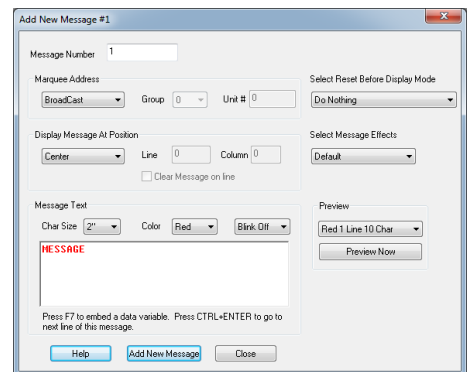
Message Database

The Message Database for EZRack PLC Designer Pro is used for populating a messages which can be utilized with the Send to Marquee instruction. Click on this function to display the Message Database Edit screen.

Note: For more information on how to use the Send to Marquee instruction please see Chapter 5.

Click onto the Add/Edit button to display the following screen:

In the Add New Message window, add the details of the message as shown. You can select the message number, marquee address, message positioning, and message text along with options for previewing the messages exactly how they would appear on an EZMarquee when sent. Click on Add New Message to add the message to the database and continue the same operation for all the messages that need to be populated.



MQTT Setup...

The MQTT Setup allows you to configure the options for the Industrial Internet of Things (IIoT). For more information about IIoT and how it works please see *Chapter 8*. To use the setup follow the directions below:

The needed information for this setup is:

Information Type	Description	Example
Domain Name	This is the broker URL. Used to find your broker that you have configured.	m12.cloudmqtt.com
Port Number	Port number that your broker uses.	16581
Client ID	Individual connection ID. Needs to be different for every client otherwise will encounter problems. Can be random.	ee097f5c-fa36-4929-9414-fad17b3df3bd
User Name	Your configured username for EZRack PLC connection to broker. Should be different for every client.	
Password	Your configured password for EZRack PLC connection to broker. Should be different for every client.	

Instruction to setup MQTT:

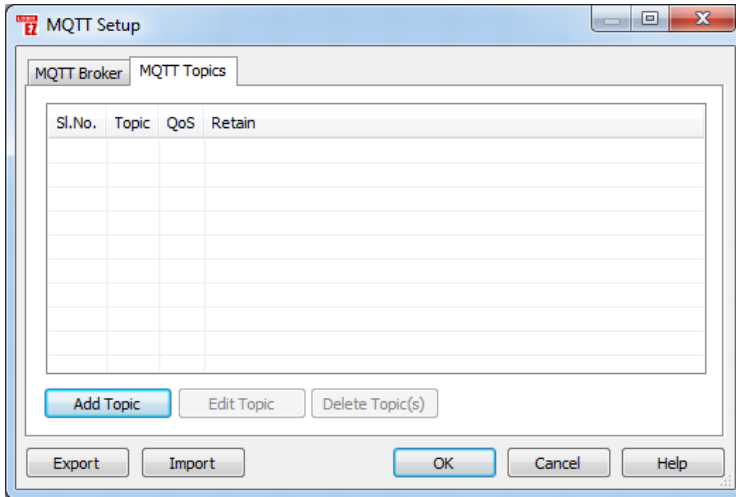
1. Go to **Setup > MQTT Setup....** You will see the following dialog box appear.

2. Use the Domain Name Lookup with the Domain Name from the broker to find the Broker IP Address.

3. Enter the port number from the broker.

4. Select your keep alive interval if wanted. See *section 8.6* for more information.

5. Enter a unique client ID or generate one using the Generate Unique Id button.
6. Enter the user name and password for your broker.
7. Go to the MQTT topics.



8. In the MQTT Topics use the Add Topic button to create the prefixes for your tags. The publish instruction will publish the tagname as a topic but if you want to have more topic information create the prefix here. For example:

Note: After this topic an "/" is appended

Topic: EZRack PLCPLC/Machine1

TagName: Speed

Published Topic: EZRack PLCPLC/Machine1/Speed

9. Now in your ladder logic add the IIoT (MQTT) Publish instruction and configure it. For configuration options please see [Section 3.3.16](#).

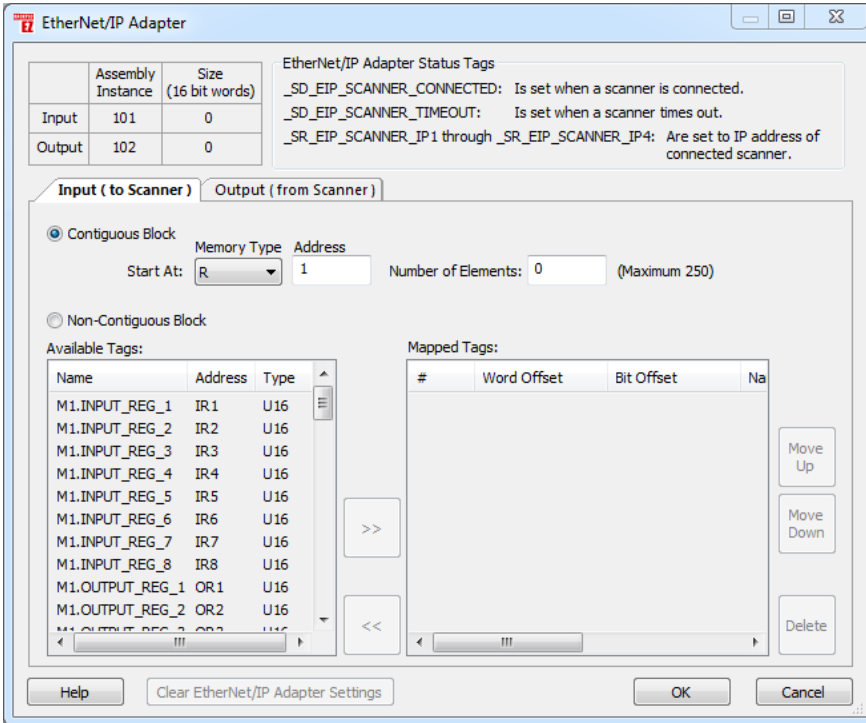
EtherNet/IP Adapter Setup...

The EZRack PLC supports Ethernet/IP Adapter communication. This communication is for the EZRack to act as Adapter to external device Scanner. This section will define how to setup the EZRack PLC and how to setup an Allen-Bradley PLC to act as Scanner to the EZRack PLC.

EZRack PLC Setup

To setup the EZRack to communicate as an Ethernet/IP Adapter follow the direction below:

1. Go to **Setup > Ethernet/IP Adapter Setup...** You will see the following screen appear:



2. This screen allows you to setup which tags are to be used for the Adapter. You can either use a contiguous block of registers (example R1-R250) or you can select the registers you would like to send. The maximum number is restricted to 250 Input and 250 Output.
3. Important information for the setup of Scanner is the Connection Parameters of the Input / Output Assembly Instances and the Input / Output Word size. Make sure to setup this information in your Allen-Bradley PLC after you have selected all tags that you will send and receive.

System Discretes:

_SD_EIP_SCANNER_CONNECTED	SD26	Read Only	Indicates when the EtherNet/IP Adapter is connected with an EtherNet/IP Scanner.
_SD_EIP_SCANNER_TIMEOUT	SD27	Read Only	Indicates when the EtherNet/IP Adapter connection times out (after 3000 mSec)

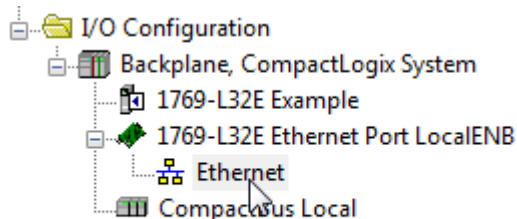
System Registers:

_SR_EIP_SCANNER_IP1	SR21	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP2	SR22	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP3	SR23	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP4	SR24	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.

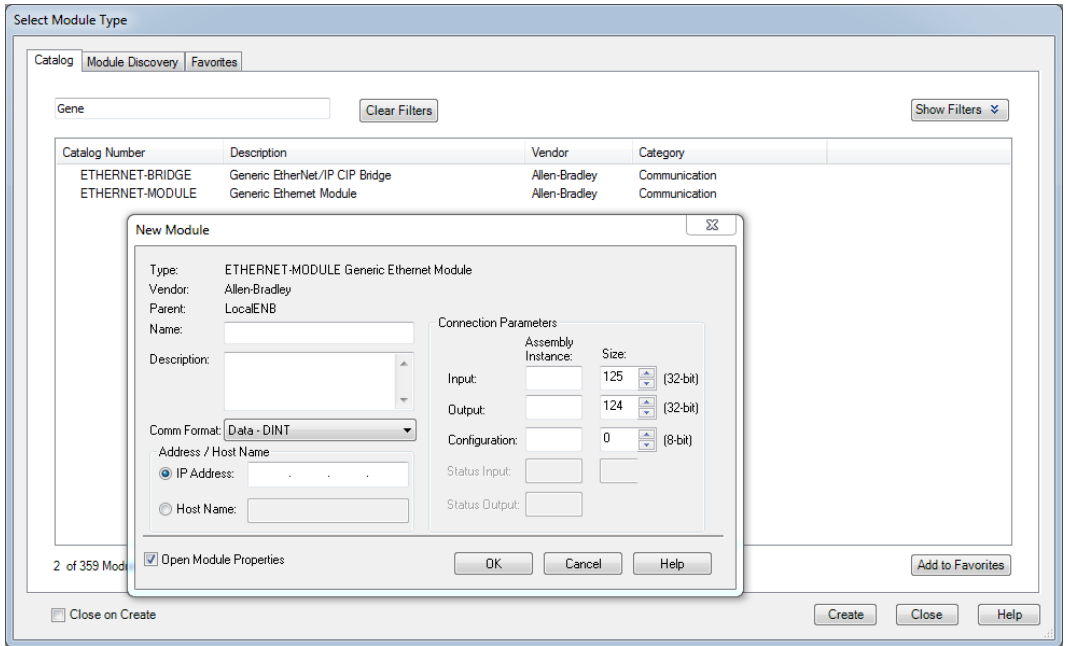
Allen-Bradley Setup

To setup an Allen-Bradley PLC to communicate to the EZRack PLC please use RSLogix 5000.

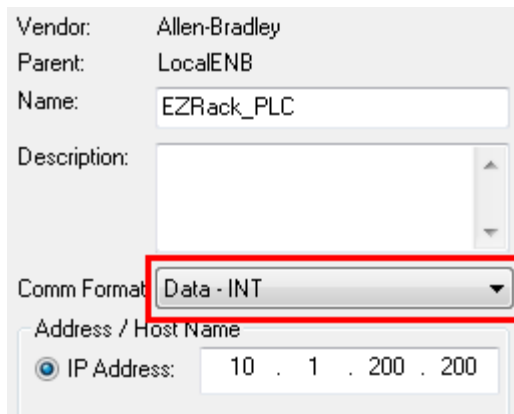
1. In the selected project under your Ethernet option click to Add New Module.



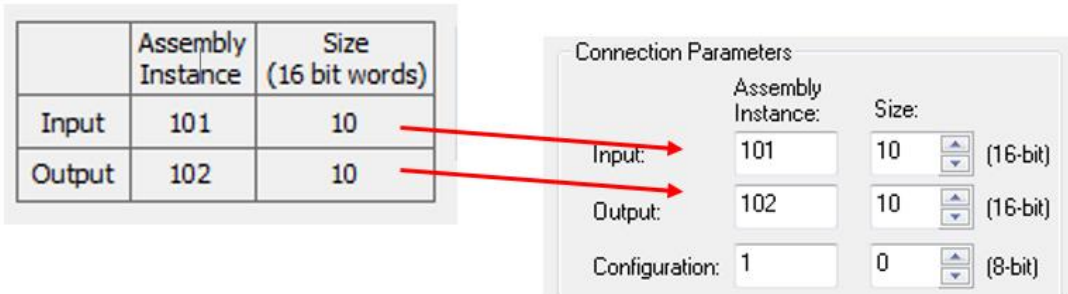
2. In the Select Module Type find the Generic Ethernet Module. Select this module and the following dialog will show up:



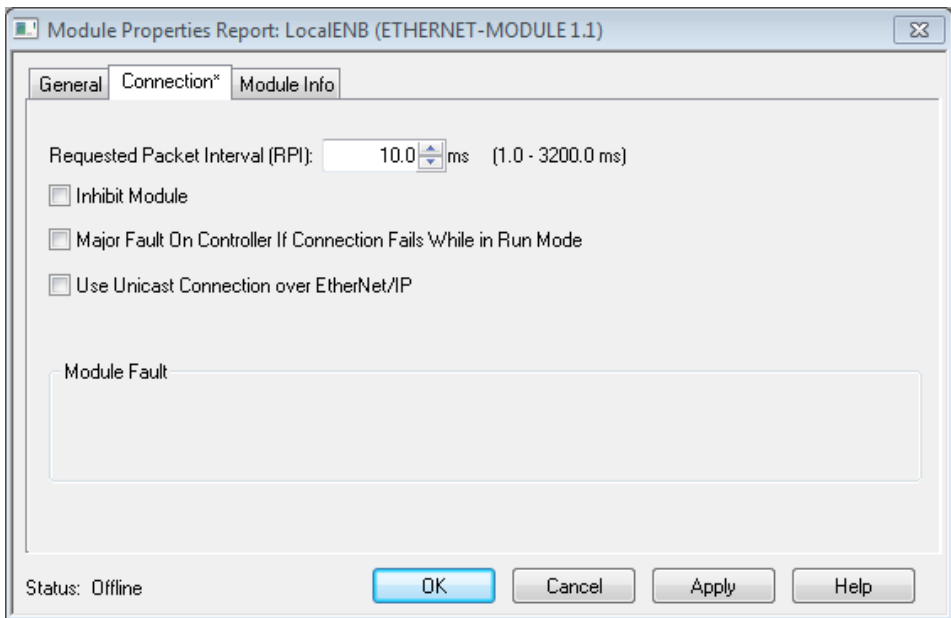
3. Add a name for your module and put in the Comm Format as Data INT. Also put in the IP Address of your EZRack PLC.



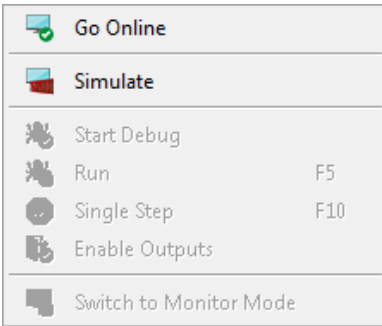
4. Copy the Connection Parameters from the EZRack PLC to the Connection Parameters of the New Module. Set the Assembly Instance Configuration to 1.



5. Click OK to create the module. In the Connection settings please use an RPI of 10 ms or more. A faster RPI does work but is not as reliable. You can also choose to use any of the other settings all function with the EZRack PLC.



6. After hitting apply you have created a connection between the EZRack PLC and the Allen-Bradley PLC. As soon as both projects are transferred to their respective PLCs they will communicate together and exchange the selected tag information. *Note: Allen-Bradley tags will be in the Controller Tags area under the name of the Generic Ethernet Module.*



2.5.9 Monitor Menu

When you click onto the Monitor Menu, you can access the following functions. These functions and their uses are explored in more detail in *Chapter 4*. *Note: Most functions are not available unless you are Online or Simulating your project.*

Go Online

Click on this function to go online with an EZRack PLC. The EZRack PLC Designer Pro will communicate with the PLC using the COM parameters setup in **PLC>COM Configuration....** The EZRack PLC Designer Pro will either transfer and Go Online with the currently open project or go online with the project that is in the PLC. If you are transferring the currently open project you will get the option to save the project that is on the PLC. Use this function before entering Monitor Mode or Debug Mode. More information is in *Chapter 4*.

Simulate

Click on this function to simulate your currently open project without need for any external hardware. After clicking this the EZRack PLC Designer Pro will transfer this project to the simulator and then behave like you are Online with a PLC (No need for any PLC and does not use configure COM parameters). When simulating you can also use Monitor Mode and Debug Mode. More information is in *Chapter 4*.

Start Debug

Click on this function to start Debug mode where you can add break points to your logic to examine what is happening. To start Debug Mode please be either Online with your PLC or Simulating your project. More information is in *Chapter 4*.

Run

When in Debug Mode (need to Start Debug before this can be used) click on this function to Run your logic. With no break point Run will start normal logic running. If there is a breakpoint then Run will cause your logic to do 1 scan till your breakpoint where it will stop. Clicking Run again will execute the rest of your logic till the next breakpoint. More information is in *Chapter 4*.

Single Step

When in Debug Mode (need to Start Debug before this can be used) click this function to single step through your logic. Single step only works after you have reached a breakpoint. Before this function can be used, Run needs to be clicked with a breakpoint added to your logic. This function does not single step through rows, it will single step by rungs. More information is in *Chapter 4*.

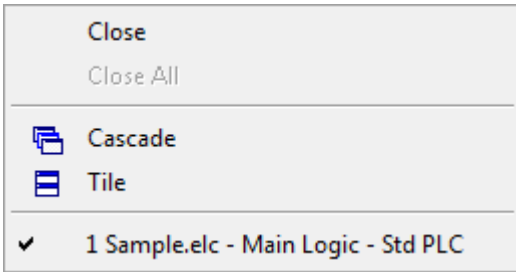
Enable / Disable Outputs

When in Debug Mode (need to Start Debug before this can be used) click on this function to enable or disable your outputs. When debugging it might be useful to not have the PLC activate your outputs therefore use this function to disable them. As soon as Debug Mode is left then outputs will be re-enabled. More information is in *Chapter 4*.

Switch Monitor Mode / Switch to Edit Mode

Click on this function to switch between Monitor and Edit mode when connected to the corresponding PLC. Please see more information on Online Edit Mode and Monitor Mode in *Chapter 4*.

Note: Monitor mode will not allow any editing of the ladder logic program present in the PLC.



2.5.10 Window Menu

When you click onto the Window Menu, you can access the following functions. This menu will also list all the currently open windows (e.g. Main Logic, Subroutine Logic, and etc.).

Close

Use this function to close the current ladder logic

window open in the main programming window. Double click on the ladder logic window in the Project View window to show them again.

Close All

Use this function to close all the ladder logic windows open in the main programming window. Double click on the ladder logic window in the Project View window to show them again.

Cascade

Click here to view open screen files in the window. Screens will cascade down the window, overlapping each other, but with their title bars in view. This is helpful when you are making changes to two or more screens at the same time. Click on the title bar of one of the screens to bring it to the front. The title bar is grayed out in screens that are not currently active.

Tile

Click here to view open screen files in the window. Screens will be arranged within the window. This is helpful if you want to copy or cut and paste objects or drawings between screens. The title bar is grayed out in screens that are not currently active.

Help Topics...



About EzLogix Designer Pro...

2.5.11 Help Menu

When you click onto the Setup Menu, you can access the following functions:

Help Topics...

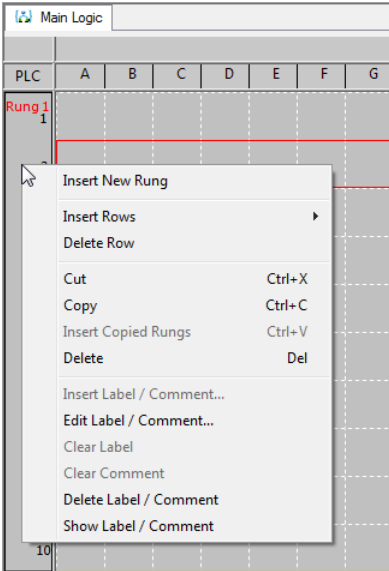
Click on Help Topics to view the help topics for the EZRack PLC Designer Pro Software. The help window is in Compiled HTML format. Use the Contents tab to view help topics by category. Click on the Index tab to view an alphabetical list of all help topics. Click on the Search tab and enter a word or words to search the help topics for.

About EZRack PLC Designer Pro...

Click on About EZRack PLC Designer Pro for copyright, manufacturer, and version number of the EZRack PLC Designer Pro Software.

2.5.12 Right-Click Menus

In addition to the drop-down menus mentioned earlier in this section, there are two more menus available to give you more options while working with your EZRack PLC Designer Pro. They can be accessed by right-clicking in two different areas in the Main Programming Screen. These menus change based on what the current EZRack PLC Designer Mode is (Offline or Online). The Online menus will be discussed in *Chapter 4*, the menu settings discussed here are for Offline Mode only.



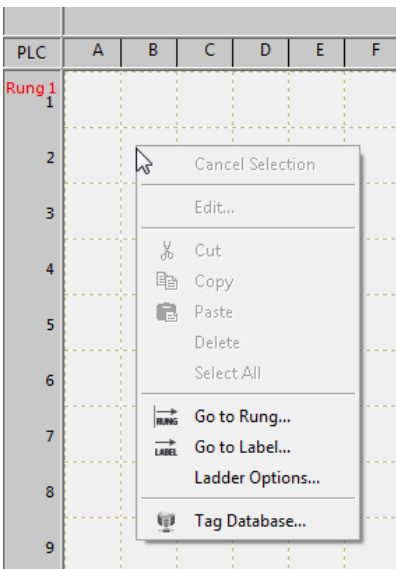
Rung Edit Right Click Menu

The first menu can be accessed by right clicking in the rungs area of the Main Programming Screen.

When you right click in the Rung area (in the square above) the following menu will appear:

Through this menu, you can access the following functions: Insert New Rungs, Insert Rows, Delete Rows, Cut, Copy, Insert Copied Rungs, Delete, Insert Label/Comment, Edit Label/Comment, Clear Label, Clear Comment, Delete Label/Comment, and Show Label/Comment.

These functions are the function that are available in the Rung Menu of the Main Menu. Please see *Section 2.5.5* of this chapter.



Instruction Edit Right Click Menu

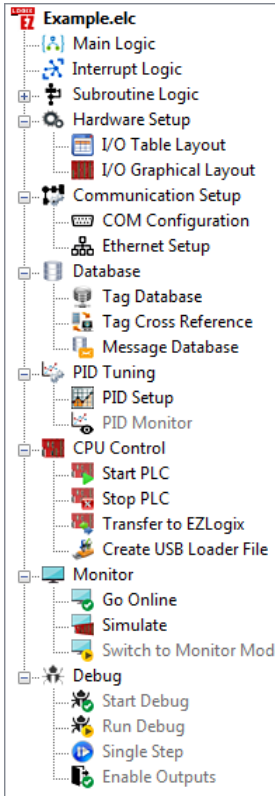
The second menu can be accessed by right clicking in the Main Logic area of the Main Programming Screen.

When you right click in the Main Logic area (in the square above) the following menu will appear:

Through this menu, you can access the following functions: Cancel Selection, Edit, Cut, Copy, Paste, Delete, Select All, Go to Rung, Go To Label, Ladder Options, and Tag Database.

These functions are the function that are available in the Edit Menu and in the Setup Menu of the Main Menu. Please see *Section 2.5.2 and 2.5.8* of this chapter.

2.6 Project View / Quick Access Bar



This side menu is a project view as well as a quick access bar to some often used functionalities. This is used to navigate through your created project logic including Main Logic, Interrupt Logic, and all Subroutine Logics. Then for quick access of menu items you have your different subsections, please see below for information on what you are accessing.

Main Logic:

As the name suggests, is the main logic of your control program. You can place some of the functions as Subroutine Logic, which is then called from main logic. You can have only 1 Main Logic.

Interrupt Logic:

The interrupt logic is a special logic section, which is executed when an external interrupt occurs. The purpose of interrupt logic is to provide a fast response to sometime critical events. You will need to use the Interrupt input module to trigger execution of Interrupt logic. You can have only 1 Interrupt Logic.

Subroutine Logic:

You may want to use Subroutine to write some logic once and use at many places in your main logic (by calling it), or just to organize your main logic in modules. You can have up to 64 subroutines. You use the Program Control Instructions to call subroutines. To add subroutine right click on this selection or use menu option Subroutine. See *Chapter 3 – RLL Instruction* for more information.

Hardware Setup:

- I/O Table Layout – Opens the I/O Configuration table. Please see *Section 2.5.8* for more information.
- I/O Graphical Layout – Opens the I/O Graphical Layout. Please see *Section 2.8* for more information.

Communication Setup:

- COM Configuration – Opens the COM Configuration. Please see *Section 2.5.7* for more information.
- Ethernet Setup – Opens the Ethernet Setup. Please see *Section 2.5.8* for more information.

Database:

- Tag Database – Opens the Tag Database. You can see all the tags for this project. Please see *Section 2.5.8* for more information.
- Tag Cross Reference – Opens the Tag Cross Reference. This allows for locating all references to a tag. Please see *Section 2.5.8* for more information.
- Message Database – Opens the Message Database. You can see all messages for this project. Please see *Section 2.5.8* for more information.

PID Tuning:

- PID Setup – Opens the PID Setup. This dialogue is used for creating up to 8 PID loops. Please see *Chapter 6* for more information.
- PID Monitor – Opens the PID Monitor. This is used for fine tuning PID loops when online with PLC. Please see *Chapter 6* for more information.

CPU Control:

- Start PLC – Starts the PLC. PLC logic will start to execute.
- Stops PLC – Stops the PLC. PLC logic will not execute if stopped.
- Transfer to PLC – Transfers current open project to PLC. Please see *Section 2.5.1* for more information.
- Create USB Loader file – Creates USB Loader file that can be used to transfer project to EZRack PLC. Please see *Section 2.5.1* for more information.

Monitor:

- Go Online – EZRack PLC Designer Pro goes online with PLC. Please see *Chapter 4* for more information.
- Simulate – Simulates the current open project. Please see *Chapter 4* for more information.
- Switch to Monitor Mode / Edit Mode – Switches current mode from Monitor Mode to Edit Mode and vice versa. . Please see *Chapter 4* for more information.

Debug:

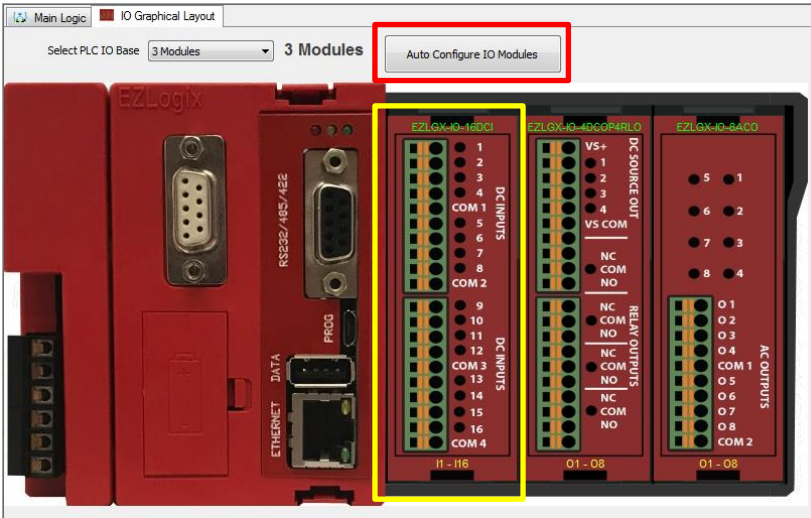
- Start Debug – Starts Debug Mode. This available after EZRack PLC Online with PLC or when Simulating Project. Please see *Chapter 4* for more information.
- Run Debug – Run Debug Mode. This available after in Debug Mode. Please see *Chapter 4* for more information.
- Single Step – Single Step through PLC Project. This available after you have Run Debug. Please see *Chapter 4* for more information.
- Enable Outputs – Enable Outputs or Disable Outputs when in Debug Mode. Please see *Chapter 4* for more information.

2.7 Operator Bar

The operator bar contains all the different instructions elements that can be used in the ladder logic. There are three ways to add instructions to ladder logic. First you can use the Instructions Menu (*Section 2.5.6*). Second you can use the Toolbar (*Section 2.3.#*). And finally you can use the operator sidebar right next to the Ladder Logic. The toolbar and sidebar can be hidden in the Edit>Toolbars Menu (*Section 2.5.2*). Below you can see all the possible instructions, please see *Chapter 3 – RLL Logic* for more information.

<p>Relay/Boolean</p> <ul style="list-style-type: none"> NO Contact NC Contact Positive Contact Negative Contact NO Coil NC Coil Set Coil Reset Coil NO Immediate Input NC Immediate Input NO Immediate Output NC Immediate Output 	<p>Compare</p> <ul style="list-style-type: none"> Equal to Not Equal to Greater than Less than Greater than or equal to Less than or equal to Limit Compare Values 	<p>Math</p> <ul style="list-style-type: none"> Add Subtract Multiply Divide Modulo Absolute X=Y Conversion Format Conversion Advance Math 	<p>Bit Logic</p> <ul style="list-style-type: none"> AND OR XOR NOT Shift Left Shift Right Rotate Left Rotate Right
<p>Timer/Counter/Drum</p> <ul style="list-style-type: none"> Timer Counter Drum 	<p>Datatype Conversion</p> <ul style="list-style-type: none"> X=Y Conversion Format Conversion 	<p>Move</p> <ul style="list-style-type: none"> Move Data Move Block Block Fill Move Table of Constants Move Bit 	<p>Function Blocks</p> <ul style="list-style-type: none"> Alarm Change of Value Compare Values Find Min & Max Value Flasher Limit Ramp Generator Scale (Linear) Scale (Non-Linear) String Pack String Unpack User Defined Faults
<p>Data Logging</p> <ul style="list-style-type: none"> Log Data To File 	<p>Communication</p> <ul style="list-style-type: none"> Open Port Send To Serial Port Receive From Serial Port Close Port Send To Marquee Modbus Master 	<p>String</p> <ul style="list-style-type: none"> String Move String Compare String Length String Pack String Unpack 	
<p>Analog</p> <ul style="list-style-type: none"> Ramp Generator Scale (Linear) Scale (Non-Linear) 	<p>Program Control</p> <ul style="list-style-type: none"> Jump For Loop Next Statement Call Subroutine Return 	<p>Process Alarms / Faults</p> <ul style="list-style-type: none"> Alarm User Defined Faults 	

2.8 I/O Graphical View

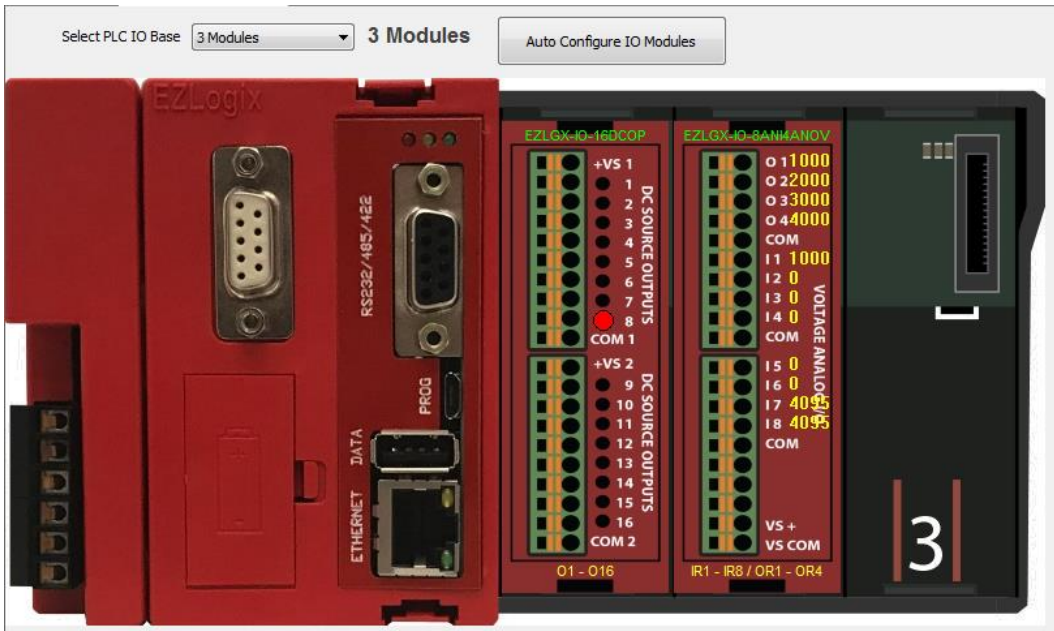


The I/O Graphical view allows for both viewing the current status of the outputs and configuring the I/O the same way the I/O Table can.

To configure please double click on module or click Auto Configure IO Modules.

View Current Output Status

When in Monitor Mode (Online with PLC or Simulating project) the I/O Graphical View will show you which outputs are ON and OFF. For Analog outputs it will indicate the value of the Analog output.



Manual Configure

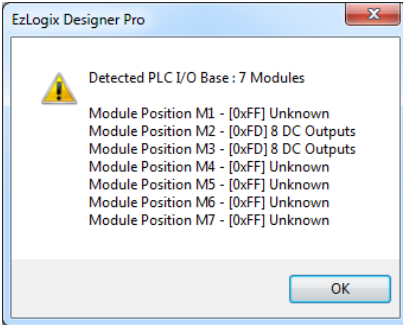
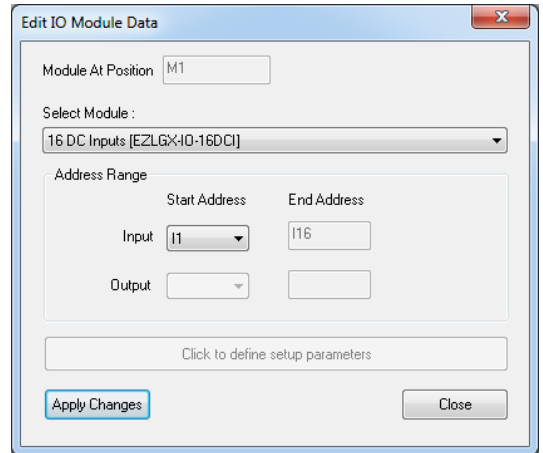
To manual configure the I/O in a specific slot double click on full or empty slot.

For example if you double click on Slot M1 you will get the following dialogue allowing you to change the module in the slot.

OR

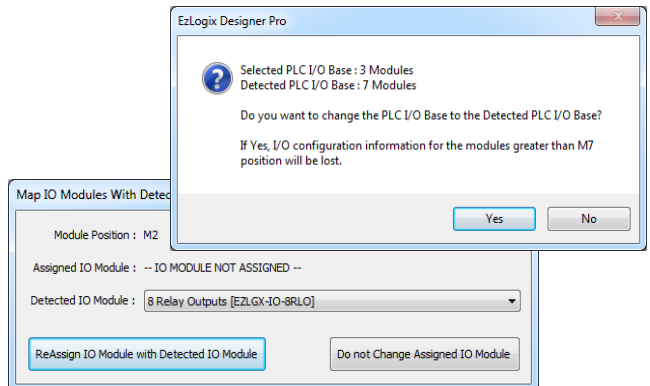
Auto Configure

To auto configure click on the auto configure button. For Auto Configure please make sure that the PC and PLC are connected (either serially, over micro-usb, or over Ethernet).



After clicking the “Auto Configure IO Modules” button you will get the following dialogue.

The EZRack PLC Designer Pro will read back and let you know which modules are currently plugged into your PLC. If the PLC base is different size, EZRack PLC Designer Pro will also ask you if you would like to change your PLC base. For each module the system will then ask you to confirm the type module and whether you would like to change that position to this modules.



Please note: certain modules such as “Sinking” or “Sourcing” for Digital Outputs and “Voltage” or “Current” for Analog modules cannot be differentiated so make sure the correct module is being selected.

Chapter 3: Instructions for Programming EZRack PLC

In this Chapter...

3.1 Ladder Logic Programming in EZRack PLC	86
3.2 Memory Map	87
3.2.1 System Discretes	88
3.2.2 System Registers	89
3.3 RLL Instructions in EZRack PLC.....	91
3.3.1 Available Data Types (Creating and Using)	96
3.3.2 Auto Generated Tags / Auto Fill Tag Address	99
3.3.3 Relay/Boolean Instructions.....	107
3.3.4 Compare Instructions.....	114
3.3.5 Math Instructions.....	121
3.3.6 Bit Logic Instructions.....	132
3.3.7 Move Instructions	137
3.3.8 Timer/Counter/Drum Instructions.....	144
3.3.9 Program Control Instructions	156
3.3.11 Communication Instructions.....	169
3.3.12 Data Logging Instructions	181
3.3.13 Datatype Conversion	188
3.3.14 Process Alarms/Faults.....	191
3.3.15 Analog	197
3.3.16 Function Blocks	204
3.3.17 IIoT	228

3.1 Ladder Logic Programming in EZRack PLC

EZRack PLC Designer Pro is used for developing relay ladder logic (RLL) programs using a Personal Computer (PC) running windows (Windows 7 or Windows 10). A PLC accepts inputs from a variety of devices such as Switches, Sensors, etc.; processes inputs according to user programmed control logic; and controls a variety of devices (e.g. relay, motors, valves etc.) connected to the outputs of the PLC. The Relay Ladder Logic is the user programmed control algorithm.

A ladder program is made up of a set of instructions to achieve the desired control processes. Ladder Logic is built on the basis of electrical relay diagrams. A ladder diagram graphically represents the elements of an electro-mechanical circuit. The user makes rungs of a ladder comprised of series or parallel combinations of the input devices and memory locations, which are usually followed by an output device or memory location. The Output element is usually the last element on the rung. Based on the conditional state of the inputs, Output receives an action signal. When the logical rung continuity is not achieved, the output is not executed.



An example of a rung is shown in the picture on the left.

If you are new to RLL programming, here is a simple sequence you should follow to develop RLL (Relay Ladder Logic) programs:

- Define your machine automation or automated process
- Determine hardware requirements for the control action
- Define a control algorithm
- Assign inputs and output parameters of the process to the control algorithm
- Develop ladder program on a PC using EZRack PLC Designer Pro Software
- Match I/O addresses of the Controller to the correct input/output devices
- Simulate your program on PC using PLC Simulator
- Load the program into the PLC
- Validate / Debug the program
- Run program

3.2 Memory Map

Each instruction is associated with one or more memory locations in the PLC. When tested, the logic instructions test, set, reset bits and/or modify values in associated bits and/or registers. EZRack PLC supports several types of memory elements (please see the hardware manual for a description of these). The tables below summarize various memory types and the ranges for each.

Description	Syntax	Access	Range	Use
Discrete Inputs	I	Read Only	1-128	For physical Discrete inputs (Input Image table, see below)
Discrete Outputs	O	Read/Write	1-128	For physical outputs (Output Image table, see below)
Discrete Internals	S	Read/Write	1-1024	General Purpose bits
System Discretes	SD	See Table	1-16	System bits (some read-only)
Input Registers	IR	Read Only	1-64	For Input modules providing register type information (such as counter module, analog input module)
Output Registers	OR	Read/Write	1-64	For output modules providing register type information (such as counter module, analog output module)
Registers Internals	R	Read/Write	1-16384	General purpose internal registers
System Registers	SR	See Table	1-20	System Register (some read-only) some read-write

What is Image Table?

EZRack PLC first reads Inputs and stores them in its internal Image Table. Then it executes the logic where any reference to Inputs/Outputs made is read from the Image Table only (except for Immediate instructions) and NOT from the actual values of Inputs/Outputs on the EZLGX IO Modules. During execution, any Outputs changed are also written to the Image Table. After completion of the RLL execution, EZRack PLC writes the Outputs from the Image Table to EZLGX IO Modules and reads the Inputs again to the Image Table and the process continues.

3.2.1 System Discretes

The table below describes all of the System Discretes available in EZRack PLC Designer Pro.

System Discretes Name	System Discretes	Description	Read/Write
_SD_FIRST_SCAN	SD1	First Scan Bit: Bit is On ONLY during the first scan of logic	Read Only
_SD_TOGGLE_100MS	SD2	Bit toggles every 100 mSec (the bit is ON for 100 mSec, and then off for 100 mSec)	Read Only
_SD_TOGGLE_1S	SD3	Bit toggles every second (the bit is ON for 1 Sec, and then OFF for 1 sec)	Read Only
_SD_RUN	SD4	Run Bit: Bit is 1 while PLC is executing ladder logic; HMI status indication	Read Only
_SD_OPEN_PORT	SD5	Setting this bit will open the port at the specified baud rate. Next, it will search for the Message Database for the defined message number in SR20.	Read/Write
_SD_SET_BAUD	SD6	If ON the baud rate of the serial port is set to 38400. If OFF the Baud Rate to is set to 9600.	Read/Write
_SD_MESSAGE_ERROR	SD7	This system discrete will be set if the Message Database is not defined or the message number is NOT defined.	Read Only
_SD_SEND_BUSY	SD8	This bit is set when a valid message is unable to be sent and will be retired.	Read Only
_SD_ANALOG_FILTER	SD9	If SD9 is ON, analog module input values will averaged so that value is more consistent and does not fluctuate as much.	Read/Write
NA	SD10-13	Reserved – DO NOT USE	
_SD_LOW_BATTERY	SD14	Indicates low battery	Read Only
NA	SD15-24	Reserved – DO NOT USE	
_SD_USB_STATUS	SD25	Indicates the current status of USB. Whether there is a USB in (1) or if there is no USB (0).	Read Only
_SD_EIP_SCANNER_CONNECTED	SD26	Indicates when the EtherNet/IP Adapter is connected with an EtherNet/IP Scanner.	Read Only
_SD_EIP_SCANNER_TIMEOUT	SD27	Turns ON when the EtherNet/IP Adapter connection times out (after 3000 mSec). Note will be always OFF if EtherNet/IP Adapter never connected.	Read Only
_SD_SPARKPLUG_CONFIG_ERROR	SD29	This bit will turn on if the Sparkplug configuration has an error.	Read Only
_SD_SPARKPLUG_PUBLISH_ERROR	SD30	This bit indicates if the EZRack PLC is not able to publish a message to the broker.	Read Only
_SD_SPARKPLUG_RUNNING_STATUS	SD31	This bit is ON if the Sparkplug MQTT is connected to the Broker and publishing correctly.	Read Only

3.2.2 System Registers

The table below describes all of the System Registers available in EZRack PLC Designer Pro. The numbers in these registers are in Binary format.

System Register Names	System Registers	Description	Read/Write	Data Type
SR_MAJOR_REV	SR3	Firmware Major Revision	Read Only	U16
SR_MINOR_REV	SR4	Firmware Minor Revision	Read Only	U16
SR_BUILD_REV	SR5	Firmware Build Number	Read Only	U16
SR_WATCHDOG_TIMER	SR6	Watchdog Timer Register; Increments every 10 ms	Read Only	U16
SR_SCAN_TIME	SR7	Scan Time in ms	Read Only	U16
SR_ERROR_MASK	SR8	Status – used to indicate errors (See below for defined bits)	Read Only	U16
SR_ERROR_NUM	SR9	Error Message Number (see below for defined errors)	Read Only	U16
SR_SECONDS	SR10	Real Time Clock (RTC) Second	Read/Write	U16
SR_MINUTES	SR11	RTC Minute	Read/Write	U16
SR_HOUR	SR12	RTC Hour	Read/Write	U16
SR_DAY	SR13	RTC Day: 1=Sunday, 2=Monday, ... 7=Saturday	Read/Write	U16
SR_DATE	SR14	RTC Date	Read/Write	U16
SR_MONTH	SR15	RTC Month	Read/Write	U16
SR_YEAR	SR16	RTC Year (only 2 digits)	Read/Write	U16
SR_CLOCK_MODE	SR17	Clock Mode: 0=24 Hour, 1=12 Hour	Read/Write	U16
SR_AM/PM	SR18	AM/PM (0=AM, 1=PM)	Read/Write	U16
SR_UPDATE_PLC_TIME	SR19	Update Clock:In Ladder Logic ONLY Set to 1 to update internal clock with the value in these registers. If setting time from a computer or HMI, DON'T write to this bit.	Read/Write	U16
SR_MESSAGE_NUMBER	SR20	The message number to be displayed if valid. A message number not defined in the message database is not a valid message and therefore the default message will be displayed.	Read/Write	U16
SR_EIP_SCANNER_IP1	SR21	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.	Read Only	U16
SR_EIP_SCANNER_IP2	SR22	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.	Read Only	U16
SR_EIP_SCANNER_IP3	SR23	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.	Read Only	U16
SR_EIP_SCANNER_IP4	SR24	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.	Read Only	U16
SR_TIME_ZONE	SR29	This register is used to set the timestamp of messages to correspond to GMT (UTC).	Read/Write	S16
SR_SPARKPLUG_CURRENT_BROKER_INDEX	SR30	If multiple Brokers are defined this register will indicate which broker is connected (0-3)	Read Only	U16
SR_SPARKPLUG_STORE_FORWARD_FIFO_COUNT	SR31	When Store and Forward is enabled this will indicate how many tag updates are stored	Read Only	U16
SR_SPARKPLUG_TASK_COUNTER	SR32	Each time that Sparkplug communication is initiated this counter will increment.	Read Only	U16

The PLC reports its errors in two system registers: SR8 and SR9. SR8 uses bits for indicating errors, while SR9 uses values to indicate the same errors. When these errors occur, the PLC halts the execution of ladder logic, but continues to communicate. So an HMI can be used to detect these errors. When PLC halts execution of ladder logic, the outputs are disabled.

Status Reported in SR8 (PLC stops executing ladder logic if error detected)	
Error Type	Bit Set to 1
Invalid User Program	Bit 0 (lsb)
No Label for Jump	Bit 1
Invalid Move data range	Bit 2
System Error	Bit 3

Error Number Reported in SR9 (PLC stops executing ladder logic if error detected)	
Error Number	Description
0	No Error
1	Invalid User Program
2	No Label for Jump
3	Invalid Move data range
4	System Error
5	Either FOR without NEXT, or NEXT without FOR

3.3 RLL Instructions in EZRack PLC

This section provides you with detailed information about using the RLL (Relay Ladder Logic) instructions in EZRack PLC. These 15 groups (1-8 similar instructions per group) of different instructions is adequate to develop some of the most powerful control programs and at the same time it's concise enough to provide the shortest learning curve.

Each of the following sections is dedicated to a type of instructions. It is organized in such a way that you will find:

- Available Data Types and how to use them
- How to use the instructions
- Descriptions of every individual instruction, including a graphical example and supported data types

The following table is provided as a quick reference to all RLL instructions available in EZRack PLC Designer Pro, as well as a brief description of what each instruction is used for.

Instruction	Description
Relay/Boolean Instructions	
NO Contact	When the corresponding memory bit is a 1 (on) it will allow power flow through this element.
NC Contact	When the corresponding memory bit is a 0 (off) it will allow power flow through this element.
Positive Transition	If the corresponding bit has changed from 0 (off) to 1 (on) in the current scan, power flows through this element.
Negative Transition	If the corresponding bit has changed from 1 (off) to 0 (on) in the current scan, power flows through this element.
NO Coil	As long as the power flows to the instruction, corresponding memory bit is remains 1(on).
NC Coil	As long as the power flows to the instruction, corresponding bit to remains 0 (off).
Set Coil	When power flows to the instructions, corresponding bit is set to 1 (on) and remains 1 (on) even if the rung condition goes to false (use RESET COIL instruction to turn the corresponding bit Off).
Reset Coil	When power flows to the instructions, corresponding bit is set to 0 (off) and remains 0(off) even if the rung condition becomes false (use SET COIL instruction to turn the corresponding bit on).
NO Immediate Input	EZRack PLC reads the addressed bit immediately from the input module (instead of memory). The power flows through the instruction if the read bit is 1 (on). (Please note all the bits corresponding to the input module are updated with the read value).
NC Immediate Input	EZRack PLC reads the addressed bit immediately from the input module (instead of memory). The power flows through the instruction if the read bit is 0 (off). (Please note all the bits corresponding to the input module are updated with the read value).
NO Immediate Output	The bit status is immediately written to corresponding output module. The bit remains 1(on) as long as the power flows to the instruction.
NC Immediate Output	The bit status is immediately written to corresponding output module. The bit remains 0(on) as long as the power flows to the instruction.

RLL Instructions Table (Continued)

Instruction	Description
Compare Instructions	
Equal To	Allows power flow through this element if the data value of Input 1 register is Equal to Input 2 register.
Not Equal To	Allows power flow through this element if the data value of Input 1 register is NOT Equal to Input 2 register.
Greater Than	Allows power flow through this element if the data value of Input 1 register is Greater Than Input 2 register.
Less Than	Allows power flow through this element if the data value of Input 1 register is Less Than Input 2 register.
Greater Than or Equal To	Allows power flow through this element if the data value of Input 1 register is Greater Than or Equal to Input 2 register.
Less Than or Equal To	Allows power flow through this element if the data value of Input 1 register is Less Than or Equal to Input 2 register.
Limit	Allows power flow through this element if the data value of Input register is within the data values of "High Limit" and "Low Limit" registers.
Compare Values	Compares 2 Inputs and evaluates whether Input 1 (>, <, =) Input 2. Turns on 1 of 3 discretes based on results.
Math Instructions	
Add	Adds two data values in Input 1 and Input 2 registers and stores the result in Result register.
Subtract	Subtracts Input 2 register data value from Input 1 register data value and stores the result in Result register.
Multiply	Multiplies two data values in Input 1 and Input 2 registers and stores the result in Result register.
Divide	Divides Input 1 register data value by Input 2 register data value and stores the result in Result register.
Modulo	Divides Input 1 register data value by Input 2 register data value and stores only the remainder in Result register.
Absolute	Converts a negative data value from Input 1 register to a positive value and stores it in Result register.
X=Y Conversion	Copies the data value of Input register, converts it into Result registers data type, and stores the data value in Result register.
Format Conversion	Converts the data value from Source register in Binary, BCD, or GRAY code to the data value of Result register in Binary, BCD or GRAY Code.
Advanced Math	Performs an advanced operation on a Input register value and outputs the resulting value into a Result register.
Bit Logic Instructions	
AND	Performs a bitwise AND operation between the data values of two registers Input 1 and Input 2. The result is stored in Result register.
OR	Performs a bitwise OR operation between the data values of two registers Input 1 and Input 2. The result is stored in Result register.
XOR	Performs a bitwise XOR operation between the data values of two registers Input 1 and Input 2. The result is stored in Result register.
NOT	Performs a bitwise NOT operation on the data value of Source register and stores the result in Destination register.
Shift Left	Performs a logical Shift Left on the data value of Input 1 register by the data value of Input 2 register and stores the result in Result register.
Shift Right	Performs a logical Shift Right on the data value of Input 1 register by the data value of Input 2 register and stores the result in Result register.
Rotate Left	Performs a logical Rotate Left on the data value of Input 1 register by the data value of Input 2 register and stores the result in Result register.
Rotate Right	Performs a logical Rotate Right on the data value of Input 1 register by the value of Input 2 register and stores the result in Result register.

Instruction	Description
Move Instructions	
Move Data	Moves data value of Source register to Destination register.
Bit Move	Moves either words to bits or bits to words with user-specified length for the number of words to move. Maximum of 16 words can be moved at a time.
Move Block	Moves a block of memory area. Source register defines the starting area of memory address/register to Move from and Destination register defines the starting area of memory address/register to move to. The number of elements to move is user defined.
Block Fill	Fills a block of memory area. Source register defines the data value to Fill with and Destination register defines the starting area of memory address/register to Fill to. The number of elements to move is user defined. The number of elements to Fill is user defined.
Move Table of Constants	Loads a table of user defined constants to a consecutive memory/register locations with the starting memory address/register location defined by Destination register.
Timer/Counter/Drum Instructions	
Timer	This instruction starts timing when called and once it reaches the preset value as defined by the data value of Timer Preset Value register, it will stop timing and will allow power flow through the element.
Counter	This instruction starts counting either Up or Down by the increments of one until the counter reaches the data value of Counter Preset Value register. The Counter will then allow power flow through the element.
Drum	Time and/or Event driven drum type sequencer with up to 16 steps and 16 discrete outputs per step. The outputs are updated during each step. Counts have a specified time base (1MSec to 1 Sec) and every step has its own counter along with an event to trigger the count. After the time expires for one step, it transitions to the next step and completes up to 16 steps total. After the completion of all the steps this element allows power flow through it.
Program Control Instructions	
Jump	Skips the rung containing Jump instruction (after execution of the rung) to a rung with the label specified in the JUMP instruction and continues executing the program thereafter.
For Loop	Executes the logic between the FOR Loop and NEXT instructions by the data value of "Loop Count" register.
Next Statement	Specifies the return/end point for the FOR Loop instruction.
Call Subroutine	Calls a Subroutine specified by the label in CALL Subroutine instruction and is terminated by the RETURN instruction.
Return	Terminates a subroutine and returns back to the main logic.
String Instructions	
String Move	Moves the data value (string type) of Source register to Destination register by the number of characters specified by the user.
String Compare	Allows power flow through this element if the data value (string type) of Source 1 register is Equal to Source 2 register by the number of characters specified.
String Length	Computes the length of a null-terminated String register (string type) and stores the result in Save Length in register.
String Pack	Combines data from 2 or more Numeric and/or String Tags into 1 common Output String Tag.
String Unpack	Extract data from Input String and place into one or more Numeric and/or String Output Tags.

RLL Instructions Table (continued)

Instruction	Description
Communication Instructions	
Open Port	Opens the serial port for communication using the parameters specified by the user.
Send to Serial Port	Send an ASCII string data from Source register to the serial port with control and character count from user defined "Control Address" and "Character Count Address" registers respectively.
Receive From Serial Port	Receives an ASCII string data from serial port to Source register with control and character count from user defined "Control Address" and "Character Count Address" registers respectively.
Close Port	Closes the serial port opened for communication (can be used for downloading projects again).
Send to Marquee	Sends ASCII instructions for marquee communication. The message to be displayed on a marquee is selected by the data value of "Message Number" register which looks up the message number for a corresponding message from the central message database. If message number is not found in the message database, user selected action for unmatched messages is done.
Modbus Master	Sends Modbus Master Read/Write commands to a configured slave unit. Can be done over Modbus RTU or Modbus TCP/IP. Can read or write from any Modbus address up to 100 registers per time. EZRack PLC can be used as Modbus slave without needing this instruction. Just need to open port.
Data Logging Instructions	
Log to Data File	Allows to log to USB up to 10 tags per instruction. Maximum of 4 instructions are supported. Can log at set time interval, log based on event, or log based on event at set time interval. Needs tags for file name (can be constant), file size (will be updated based on current conditions), and status (current log status).
Datatype Conversion Instructions	
X=Y Conversion	Copies the data value of Input register, converts it into Result registers data type, and stores the data value in Result register.
Format Conversion	Converts the data value from Source register in Binary, BCD, or GRAY code to the data value of Result register in Binary, BCD or GRAY Code.
Process Alarms / Faults Instructions	
Alarm	Evaluates the current condition of Input Tag. Will set 4 output discrete tags based on value of Input tag. Compares Input to a tag based (or constant) setpoints (2 setpoints above and 2 setpoints below normal operating range).
User Defined Faults	Evaluates conditions based on multiple inputs and generates faults when conditions are true. For example if Input 1 Register is equal to Input 2 Register, will set fault number as fault code (Bit 0 of fault code = fault 1, etc.) and fault found tags.
Analog Instructions	
Ramp Generator	Increments / decrements an Output Tag based on Ramp Rate and Ramp Time tags (can be constant). Goes from minimum to maximum output, if exceed or equal extremes will set overflow tag.
Scale (Linear)	Scales the Input Tag to Output Tag linearly based on a 2 point reference (each point has input and output value).
Scale (Non-Linear)	Scales the Input Tag to Output Tag non-linearly using up to 10 reference points. Scale is linear between every 2 reference points.

RLL Instructions Table (continued)

Function Block Instructions	
Alarm	Evaluates the current condition of Input Tag. Will set 4 output discrete tags based on value of Input tag. Compares Input to a tag based (or constant) setpoints (2 setpoints above and 2 setpoints below normal operating range).
Change of Value	Finds the change of value of 1 Input Tag between 2 points of time. The Output Tag is the change value of the Input Tag between T1 and T2.
Compare Values	Compares 2 Inputs and evaluates whether Input 1 (>, <, =) Input 2. Turns on 1 of 3 discretes based on results.
Find Min & Max Value	Outputs two tags whose value is the maximum and minimum value the Input Tag has reached since last reset.
Flasher	Turns a discrete ON/OFF at a settable rate of time. The rate time is based on the Flash Rate tag (can be constant).
Limit	Allows power flow through this element if the data value of Input register is within the data values of "High Limit" and "Low Limit" registers.
Ramp Generator	Increments / decrements an Output Tag based on Ramp Rate and Ramp Time tags (can be constant). Goes from minimum to maximum output, if exceed or equal extremes will set overflow tag.
Scale (Linear)	Scales the Input Tag to Output Tag linearly based on a 2 point reference (each point has input and output value).
Scale (Non-Linear)	Scales the Input Tag to Output Tag non-linearly using up to 10 reference points. Scale is linear between every 2 reference points.
String Pack	Combines data from 2 or more Numeric and/or String Tags into 1 common Output String Tag.
String Unpack	Extract data from Input String and place into one or more Numeric and/or String Output Tags.
User Defined Faults	Evaluates conditions based on multiple inputs and generates faults when conditions are true. For example if Input 1 Register is equal to Input 2 Register, will set fault number as fault code (Bit 0 of fault code = fault 1, etc.) and fault found tags.
IIoT Instructions (Industrial Internet of Things)	
IIoT (MQTT) Publish	Allows the EZRack PLC to publish up to 10 tags to the Broker. Need to use the Setup > MQTT Setup... option before this instruction can be used. Please see <i>Chapter 8</i> for more information.

3.3.1 Available Data Types (Creating and Using)

These are the available data types which can be used in instructions. This section will define the difference between the different data types. Also this section will also show how to define data types, set default data type, and how to work with auto generated data types.

DATA TYPE vs MEMORY TYPE

Data Type: The data type of a tag determines how the bits representing those values are stored in the PLC's memory. For example for the number 72 here is how they are stored:

Data Type	Value	Representation in Bits
UNSIGNED_INT_16	72	0000 0000 0100 1000
ASCII_STRING	72	0011 0111 0011 0010

Memory Type: The memory type of a tag determines the size and location in the PLC where the tag is stored. For example discrete S Internals have a size of 1 bit than register R Internals which are 16 bits long. Therefore the memory location for them is different.

Memory Type Table

Memory Type	Syntax	Range	Data Types Available
Discrete			
Discrete Inputs	I	1-128	DISCRETE
Discrete Outputs	O	1-128	DISCRETE
Discrete Internals	S	1-1024	DISCRETE
System Discretes	SD	1-16	DISCRETE
Bit Access Registers*	R	1-16384 / 0-15	DISCRETE
Registers			
Register Inputs	IR	1-64	SIGNED_INT_16, SIGNED_INT_32, UNSIGNED_INT_16, UNSIGNED_INT_32, BCD_INT_16, FLOATING_PT_32, ASCII_STRING
Register Outputs	OR	1-64	SIGNED_INT_16, SIGNED_INT_32, UNSIGNED_INT_16, UNSIGNED_INT_32, BCD_INT_16, FLOATING_PT_32, ASCII_STRING
Register Internals	R	1-16384	SIGNED_INT_16, SIGNED_INT_32, UNSIGNED_INT_16, UNSIGNED_INT_32, BCD_INT_16, FLOATING_PT_32, ASCII_STRING
System Registers	SR	1-20	SIGNED_INT_16, SIGNED_INT_32, UNSIGNED_INT_16, UNSIGNED_INT_32, BCD_INT_16, FLOATING_PT_32, ASCII_STRING
Index Registers	XR	1-4	UNSIGNED_INT_16
Value Registers	#R	1-4	UNSIGNED_INT_16
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. For all 16,384 registers you can individually access all 16 bits of them.			

Data Type Table

Data Type	Min Value	Max Value	Number of Registers used ²	Memory Types where used
DISCRETE	OFF (0)	ON (1)	1	I, O, S, SD, R ¹
SIGNED_INT_16	-32,768	32,767	1	IR, OR, R, SR,
SIGNED_INT_32	-2,147,483,648	2,147,483,647	2	IR, OR, R, SR,
UNSIGNED_INT_16	0	65,535	1	IR, OR, R, SR,
UNSIGNED_INT_32	0	4,292,967,295	2	IR, OR, R, SR,
BCD_INT_16	0	9,999	1	IR, OR, R, SR,
FLOATING_PT_32	-10,000,000,000.0000	10,000,000,000.0000	2	IR, OR, R, SR,
ASCII_STRING	2 Char	126 Char	Variable (1-63)	IR, OR, R, SR,
¹ Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using Tag_Name.Bit_0. For more information see section 3.3.2.				
² If 2 or more registers are used than for example if R1 is the Tag Address then R2 will also be used. If you use R2 in another register, results will be unpredictable.				

DISCRETE

The discrete data type is the single bit data type. It is used to indicate an ON (1) or OFF (0) state controlling the operations of logic. It is used in instructions like Normally Open Contact, Normal Open Coils, and etc. Please see tables for more information.

Note: For bit access discretets you will use the format of Tag_Name.Bit_Number. For example with tag Start (R1) to access bit 0 you will type Start.0 (R1/0). You can also add bits within a word in the tag database using the Add Bit-In-Register Tag button.

UNSIGNED_INT_16

The basic single word register data type. The most used register type when negative numbers are not needed. Almost all instructions can use this data type. Please see tables for more information.

UNSIGNED_INT_32

The basic double word register data type. Used when you need to store larger values and negative numbers are not needed. Almost all instructions can use this data type. Please see tables for more information.

Note: This data type uses two register address for example if R1 is the Tag Address then R2 will also be used. If you use R2 in another register, results will be unpredictable.

SIGNED_INT_16

The basic signed single word register data type. The most used register type when need to have negative number. Almost all instructions can use this data type. Please see tables for more information.

SIGNED_INT_32

The basic signed double word register data type. Used when you need to store larger values and need negative numbers. Almost all instructions can use this data type. Please see tables for more information.

Note: This data type uses two register address for example if R1 is the Tag Address then R2 will also be used. If you use R2 in another register, results will be unpredictable.

BCD_INT_16

This is the binary coded decimal single word data type. Can only be used in specific instructions but is great for displaying decimal values and for ease of conversion. Please see tables for more information.

FLOATING_PT_32

This is double word data type used for working with really large or really small numbers. Using single-precision you can work with exponentially large numbers or decimal numbers. When displaying can be in either decimal notation or scientific notation. Almost all instructions can use this data type. Please see tables for more information.

Note: This data type uses two register address for example if R1 is the Tag Address then R2 will also be used. If you use R2 in another register, results will be unpredictable.

ASCII_STRING

This is the data type that is used with string instructions. This data type stores character information as ASCII data. Each register address will store 2 characters and the maximum character length is 126. Therefore please note this data type can use from 1 to 63 registers.

Note: This data type uses two or more register addresses for example if R1 is the Tag Address then R2, R3, R4, etc. can also be used. If you use R2 in another register, results will be unpredictable.

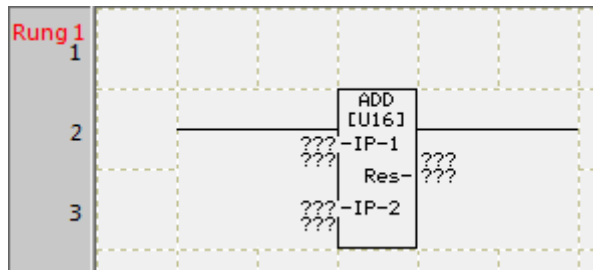
3.3.2 Auto Generated Tags / Auto Fill Tag Address

The EZRack PLC Designer Pro will automatically fill in the next available address when a new tag is created. For I/O tags please configure the PLC with the modules you will be using. This will create the needed I/O tags for those modules. Please see below for other Auto Populate tags.

To create a new tag please follow the process below:

Ladder Logic Creation

1. Add a new instruction to the ladder logic. For example an Add Instruction.



2. Double click on the instruction and enter a tag name for your Input 1. Example Value 1

The 'Add Instruction' dialog box is shown. The 'Instruction Details' tab is active, displaying the formula 'Result = Input 1 + Input 2'. The 'Input 1' section has 'Tag Name' selected, with 'Value 1' entered in the dropdown. The 'Data Type' is set to 'UNSIGNED_INT_16'. The 'Input 2' section has 'Tag Name' selected, with an empty dropdown. The 'Save Result in (Res)' dropdown is empty. The 'Display all values in' dropdown is set to 'Decimal'. The dialog has 'OK', 'Cancel', and 'Help' buttons at the bottom.

3. Hit enter or right click in the Tag Name area. The Add New Tag Details box will pop up. The box will be prepopulated with the next available Memory Address. The starting point from which these addresses are taken can be changed in the Address Suggestion section. The only change needed is to select the data type. The memory address can be changed but it should only be done if needed for instructions like Block Move also you might need to this for the Modbus slave Memory map (for more info see Section 7.2.4 EZRack as Modbus Slave).
 - a. If creating a Discrete the Tag Details box will not appear since the data type need not be modified.
 - b. To use Input and Outputs please set select the used I/O modules in the I/O Table. For more information please see section 2.5.8 about IO Configuration.

The screenshot shows a dialog box titled "ADD NEW TAG DETAILS" with a close button (X) in the top right corner. The main title inside the dialog is "Enter Tag Details for the Tag". Below this, there is a text field containing "VALUE 1".

The dialog contains several input fields and controls:

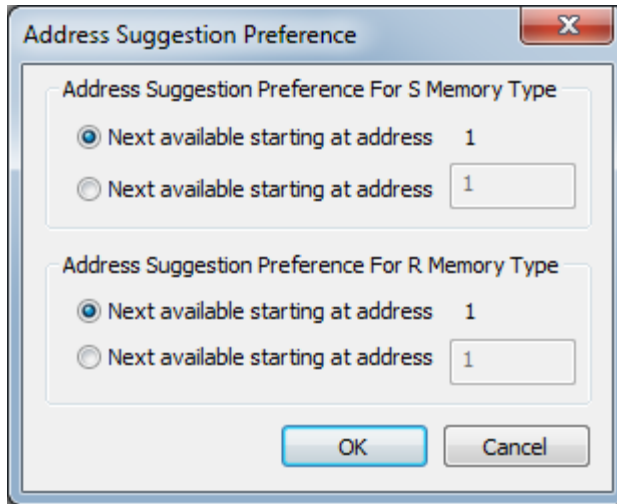
- Address:** A dropdown menu set to "R" and a text field containing "1".
- Expected IO Type:** A label indicating "Expected IO Type : R/W or ROnly".
- Data Type:** A dropdown menu currently showing "UNSIGNED_INT_32". A mouse cursor is hovering over the "UNSIGNED_INT_16" option, which is highlighted in blue. Other visible options in the list include "SIGNED_INT_16", "SIGNED_INT_32", "UNSIGNED_INT_32", and "FLOATING_PT_32".
- No. of Chars:** A label next to the Data Type dropdown.
- Initial Value:** A checkbox labeled "Initial Value" which is currently unchecked, followed by an empty text field for the value.
- Address Suggestion Preference:** A button located to the right of the Address field.

At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Help".

4. You can now add an Initial Value if you would like. Next just click OK and the tag will be created.

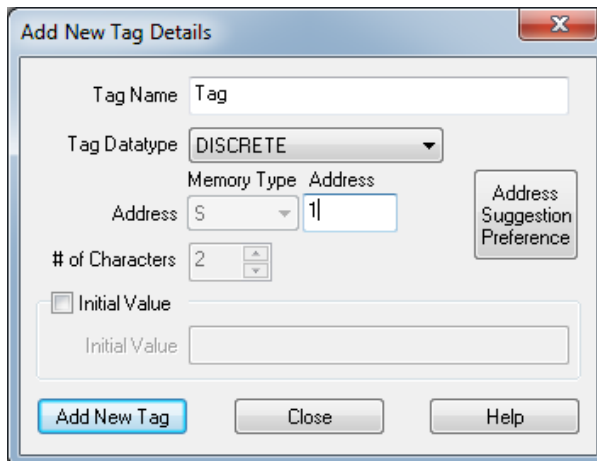
Address Suggestion Preference

The EZRack PLC will suggest addresses based on the select starting point here. Initially it will start from Address 1 and suggest the next available address. The user can at any point tell the software to start suggesting from a different starting point. For example if starting address is set to 200 then if 200 is available that will be the next suggested address, if it is not then it will go to 201 and so and so forth.



Tag Database Creation

1. In the tag database click on the Add Tag option. The Add New Tag Details box will appear.
2. Enter the wanted Tag Name and then select the Tag Datatype.



3. Based on the tag Datatype you will either use the S Memory Type or the R Memory Type. The box will be prepopulated with the next available Memory Address. The starting point from which these addresses are taken can be changed in the Address Suggestion Preference. The memory address can be changed but it should only be done if needed for instructions like Block Move also you might need to this for the Modbus slave Memory map (for more info see Section 7.2.4 EZRack as Modbus Slave).
4. You can now add an Initial Value if you would like. Next just click OK and the tag will be created.

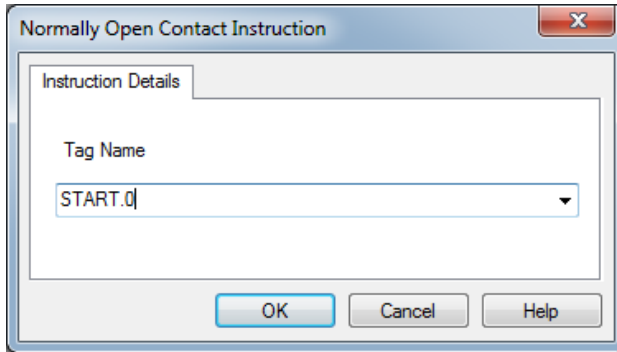
Note: The automatically filled in tag address can be modified at any point by the user. This means that while the auto fill will check and make sure that the used range does not overlap with any other tag, the user can make changes which mean that tags can overlap again. Please see example below.

Example of user error:

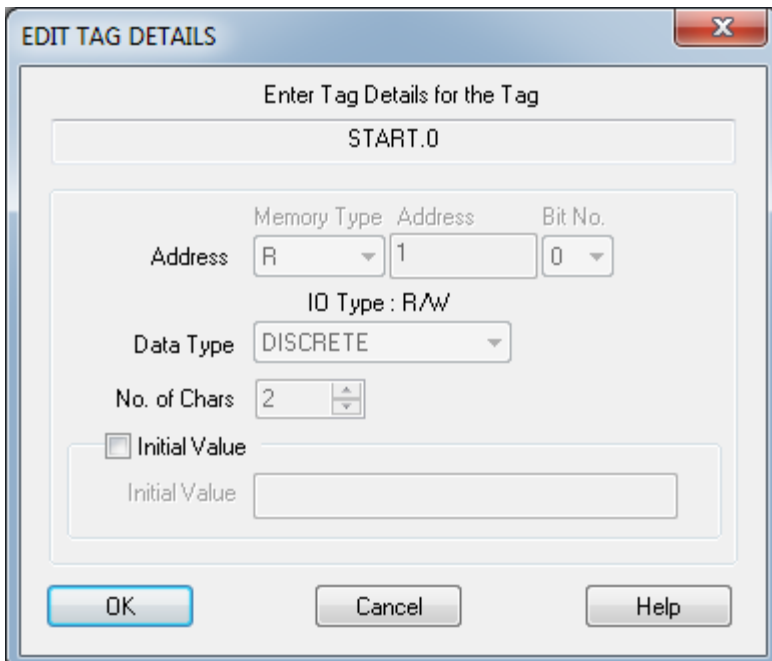
1. Project currently uses R2 and R3 for something.
2. User creates a tag which is `SIGNED_INT_32`, the auto fill will put this tag at register address R4.
3. The user can change the address to R1 but at that point this tag will use R1 and R2 -> this will cause overlap with the already created tag at R2.

Bits within a Word (Bit Access)

To access the bits with a word you will use the format of Tag_Name.Bit_Number. For example with tag Start (R1) to access bit 0 you will type Start.0 (R1/0). You can also add bits within a word in the tag database using the Add Bit-In-Register Tag button.



Note: The Register tag needs to exist before you can add bit access to that tag.



Auto Populated Tags

There also exist some tags which will automatically be populated:

I/O Modules tags

Any IO module added to the project will populate the tag database with corresponding IO tags. Please see section 2.5.8 for more information.

System Tags

All systems tags are now automatically included in the project. For more information on system tags please see section 3.2.

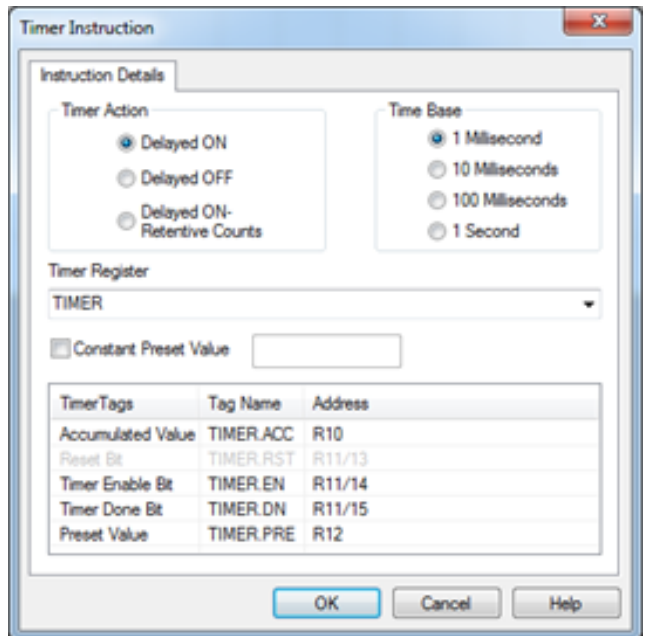
PID tags

PID tags will be automatically created after a new PID Loop is created. You will have the choice of changing the memory location of those tags if need be. For more information on PID tags please see Chapter 6.

Instruction Auto Generated Tags

Certain instructions will auto generate some tags. These tags will have the same name as the base tag used, with a "." and some more information afterwards. These tags cannot be deleted unless the base tag is deleted. The base tag used to generate these tags can be either the input or the output of the instruction. The instructions will inform you about the tags that were generated and their Tag addresses.

Note: Please be careful that the tag address used by these Auto Generated tags are not used elsewhere unintentionally. This may cause unpredictable behavior.



For example the Timer Instruction will create tags: TIMER.ACC, TIMER.RST, TIMER.EN, TIMER.DN, and TIMER.PRE with the base tag of TIMER. Base tag address is R10 and the rest of auto generated tags use R11 and R12.

These tags will appear in the tag database but they cannot be edited or deleted unless the base tag and its associated object

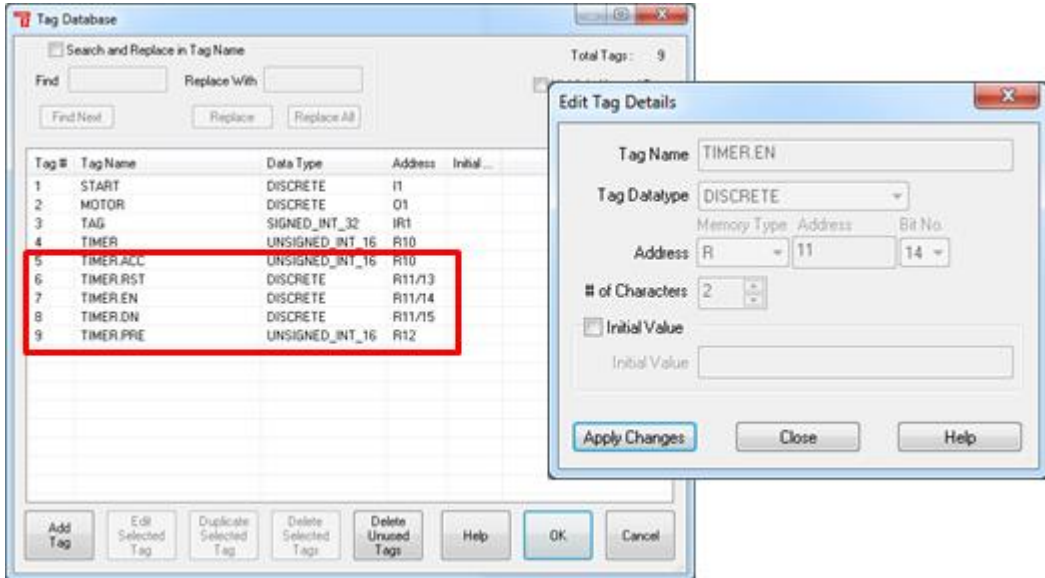


Table of Tag Formats

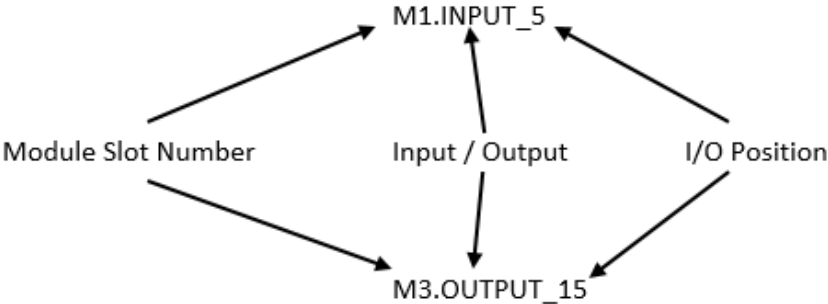
This table describes all the different possible tag formats that exist. While it may look complicated most of these are auto created for you so the user only needs to create the basic tags. *Note: All tags can have their name modified except for system and reserved tags.*

Tag Name Structure	Possible Data Types	Description	How Created	Example
TAG_NAME	Discrete or Register	This is the basic tag the user can create and modify the address of.	User Created.	LIGHT, VALUE1
TAG_NAME.#	Discrete (Bit within word)	Bit access to the register. Works with Registers only.	User Created.	VALUE1.0
TAG_NAME.RES#	Register	Reserved tags for internal calculations.	Auto Created for certain function blocks.	VALUE1.RES1
M#.INPUT_#	Discrete	Module # Inputs. User can changed name.	Created during I/O Config.	M1.INPUT_1
M#.OUTPUT_#	Discrete	Module # Outputs. User can changed name.	Created during I/O Config.	M3.OUTPUT_7
M#.INPUT_REG_#	Discrete	Module # Input Registers. User can changed name.	Created during I/O Config.	M7.INPUT_REG_3
M#.OUTPUT_REG_#	Discrete	Module # Output Registers. User can changed name.	Created during I/O Config.	M5.OUTPUT_REG_5

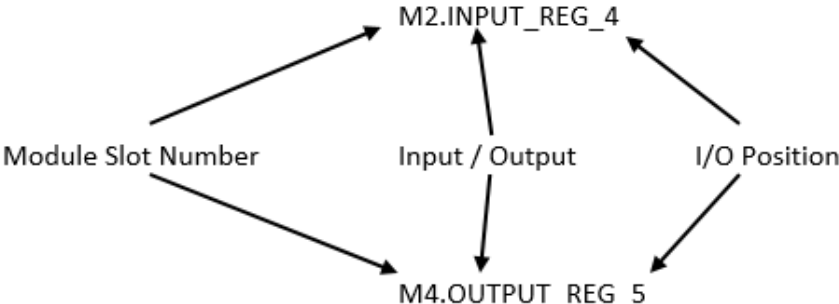
M#.TAG_NAME	Discrete or Register	Specialty module # tags. User can changed name.	Created during I/O Config.	M4.CNTR1_COUNTERS
_SD_TAG_NAME	Discrete	System discrete.	Always in project.	_SD_FIRST_SCAN
_SR_TAG_NAME	Register	System register.	Always in project.	_SR_MINUTES

I/O Address Format for EZRack PLC

Bit



Register



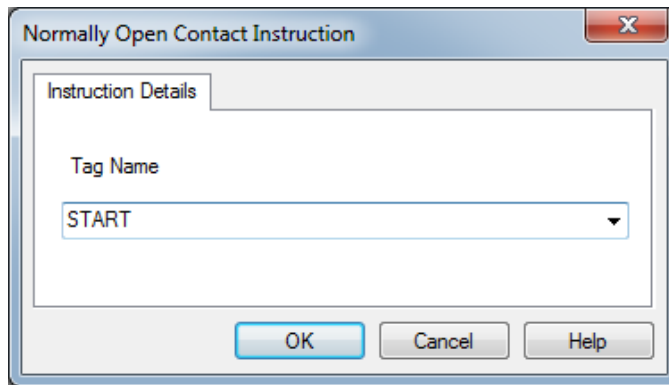
3.3.3 Relay/Boolean Instructions

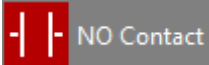
Use discrete instructions to monitor and control the status of bits in the PLC. The bits that can be monitored / controlled by using relay instructions are inputs, outputs, internal bits and system bits.

Adding Relay/Boolean Instructions

To configure all of the various Relay/Boolean instructions, perform the following steps:

1. Click on any Boolean instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click to place the instruction.
3. To enter the Tag name/address, double click the instruction to open its Dialog box.
4. Select a proper Tag name/address from the drop down list called Tag Name.
 - a. If need be you can add a new tag by entering a new Tag Name
 - b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
 - c. Enter the Tag Address in this screen.

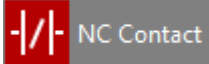




NO Contact

Normally Open Contact:

The Normally Open Contact instruction reads/examines an input or storage bit at memory location Aaaaa. If the corresponding memory bit is ON (1), power will flow through this element.



NC Contact

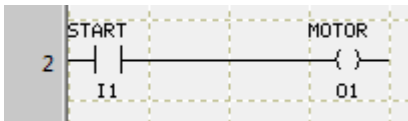
Normally Closed Contact:

The Normally Closed Contact instruction reads/examines an input or storage bit at memory location Aaaaa. If the corresponding memory bit is OFF (0), power will flow through this element.

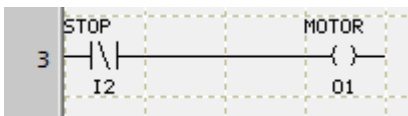


Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Inputs	I	1-128
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretes	SD	1-16
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete and Bit Access to Registers

In the example above, when input I1 is ON, output O1 will energize.



In the example above, when input I2 is OFF, output O1 will energize.

-P- Positive Contact

Positive Contact:

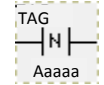
The Positive Contact instruction reads/examines an input or storage bit at memory location Aaaaa. If the addressed bit has transitioned from the OFF (0) to the ON (1) state in the current scan, power will flow through this element for the rest of that scan.



-N- Negative Contact

Negative Contact:

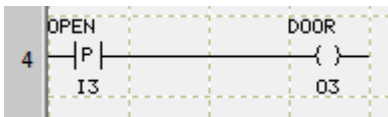
The Negative Contact instruction reads/examines an input or storage bit at memory location Aaaaa. If the addressed bit has transitioned from the ON (1) to the OFF (0) state in the current scan, power will flow through this element for the rest of that scan.



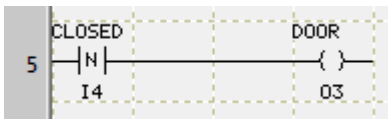
Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Inputs	I	1-128
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretets	SD	1-16
Register Discretets		
Bit Access Registers*	R	1-64 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete and Bit Access to Registers (Only the first 64 registers)



In the example above, every time I3 makes an off-to-on transition in current scan, O3 will energize for a single scan.



In the example above, every time I4 makes an on-to-off transition in current scan, O3 will energize for a single scan.

Note: The Positive and Negative Contact instructions test whether a bit has changed from 0 to 1 or 1 to 0 during the current scan of ladder logic respectively. Therefore, to use these instructions, the logic to change the state of the bit MUST be placed before the logic containing this instruction. If the logic for change of state is placed after the instruction, the instruction will never see the transition, and therefore will never be true.



Normally Open Coil:

As long as power flows to this element, the bit Aaaaa associated with the Normally Open Coil instruction remains ON (1).



Normally Closed Coil:

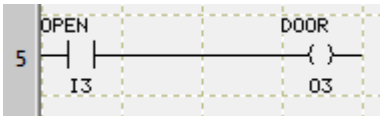
As long as power flows to this element, the bit Aaaaa associated with the Normally Closed Coil instruction remains OFF (0).



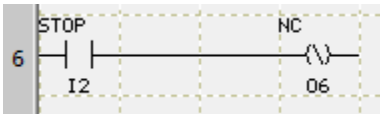
Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete and Bit Access to Registers



In the example above, O3 energizes when I3 transitions from 0 to 1.



In the example above, O6 will be de-energized as long as I2 is ON.



Set Coil:

When power flows to this element, the Set Coil instruction sets/turns ON (1) the specified output or storage bit at memory location Aaaaa. Once the specified output or storage memory bit is turned ON (1), it will remain ON (1) even if the rung conditions change later to stop power flow to the element. The only way to change the status of the specified storage or memory bit set by 'Set Coil' is to use the 'Reset Coil' instruction.



Reset Coil:

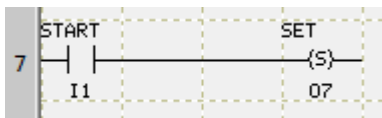
When power flows to this element, the Reset Coil instruction resets/turns OFF (0) the specified output or storage bit at memory location Aaaaa. Once the specified output or storage memory bit is turned OFF (0), it will remain OFF (0) even if the rung conditions change later to stop power flow to the element. The only way to change the status of the specified storage or memory bit reset by 'Reset Coil' is to use the 'Set Coil' instruction to set/turn ON (1).



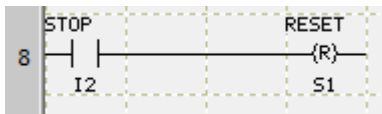
Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

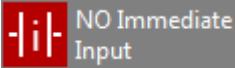
Allowed Data Formats: Discrete and Bit Access to Registers



In the example above, bit 07 is Set when I1 is ON. Bit 07 will remain Set even after I1 becomes FALSE.

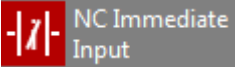


In the example above, if I2 is ON, S1 will be Reset (turned OFF).



Normally Open Immediate Input:

The Normally Open Immediate Input instruction reads/examines the status of the specified Input point at location Aaaaa directly from the EZLGX IO module at the time of execution and NOT from the memory bit present in the Image Table. If the corresponding input state is ON (1), power will flow through this element. All the available inputs on corresponding module are read only. The Image Table is also updated with the read input memory locations.



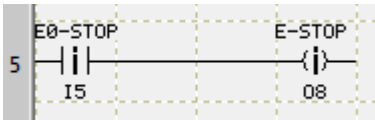
Normally Closed Immediate Input:

The Normally Closed Immediate Input instruction reads/examines the status of the specified Input point at location Aaaaa directly from the EZLGX IO module at the time of execution and NOT from the input memory bit present in the I/O scan image. If the corresponding input state is OFF (0), power will flow through this element. When Aaaaa corresponds to an EZLGX IO input module, all the available inputs on the corresponding module are read only. The Image Table is also updated with the read input memory locations.

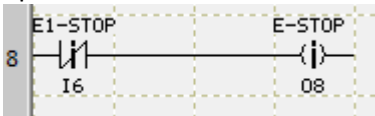


Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Inputs	I	1-128

Allowed Data Formats: Discrete Only



In the example above, when instruction `--|I|--` is executed, the input module addressed I8 is read, and then the rung is solved. All inputs (I1- I8) on that module are read and the memory is updated.



In the example above, when instruction `---|I|---` is executed, the input module addressed I8 is read, and then the rung is solved. All inputs (I1- I8) on that module are read and the memory is updated.

A Normal PLC scan consists of reading inputs (and saving the input status in memory or input image table), solving ladder logic, and writing outputs (from memory or output image table). During a logic scan, if a reference to an input comes up, the value stored in memory is used for that input. Similarly, if logic needs to energize an output, a corresponding memory bit is set, which is later written to the physical output during the I/O scan phase.

The immediate input instructions allow you to read a corresponding input bit at the time of instruction execution, and use the most current bit status (instead of the status stored in memory during input read) in logic solving. Immediate input instructions update all the bits corresponding to an input module, even if only one of the bits is used in an Immediate input instruction. For example, if I1 is used for immediate input, which is on an 8 input card, bits I1 - I8 would be updated immediately.



NO Immediate Output

Normally Open Immediate Output:

When power flows to this element, the Normally Open Immediate Output instruction sets/turns ON (1) the specified output point at memory location Aaaaa directly on the EZLGX IO module and the output memory bit in the Image Table at the time of execution.



NC Immediate Output

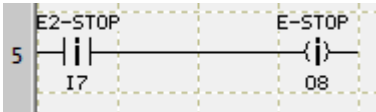
Normally Closed Immediate Output:

When power flows to this element, the Normally Open Immediate Output instruction resets/turns OFF (0) the specified output point at memory location Aaaaa directly on the EZLGX IO module and the output memory bit in the Image Table at the time of execution.

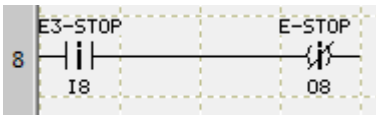


Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Outputs	O	1-128

Allowed Data Formats: Discrete Only



In the example above, if the power flows to the output instruction, O8 will be energized and immediately written to the physical output corresponding to O8.



In the example above, if the power flows to the output instruction, O8 will be de-energized and immediately written to the physical output corresponding to O8.

A Normal PLC scan consists of reading inputs (and saving the input status in memory or input image table), solving ladder logic, and writing outputs (from memory or output image table). During logic scan, if a reference to an input comes up, the value stored in memory is used for that input. Similarly, if logic needs to energize an output, a corresponding memory bit is set, which is later written to physical output during the I/O scan phase.

The immediate Output instructions allow you to write to the corresponding physical output at the time of instruction execution, instead of waiting for the I/O scan to write the output. Only the output referred to by the instruction is updated.

3.3.4 Compare Instructions

Compare instructions allow you to compare values using a specific comparison instruction. When using compare instructions you must compare values of the same data and display type. The parameters you enter are program constants or logical addresses of the values you want to compare.

Compare instructions perform comparisons of two addresses Input 1 and Input 2 defined by the data box selected. When the processor finds the expression is true, the power flows through these instructions.

Adding Compare Instructions

To add Compare Instructions, perform the following steps:

1. Click on any Compare instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

4. Select a proper Tag name/address from the drop down list for Input 1. Or add a new tag.
5. Select a proper Tag name/address from the drop down list for Input 2. Or add a new tag.
6. Choose the correct data format from the last drop down list on the dialog box.
7. Data types for both Input 1 and 2 must be the same.

Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

Note:

1) Data of five different types *SIGNED_INT_16*, *SIGNED_INT_32*, *UNSIGNED_INT_16*, *UNSIGNED_INT_32* or *FLOAT_32* is allowed.

2) Word Data Types default to decimal display type.

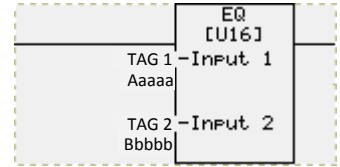
3) *UNSIGNED* Data Types also allow Hex and Octal displays.

4) Display Type allows you to select how the number will be displayed in the program. There are three display options

= Equal to

Equal To:

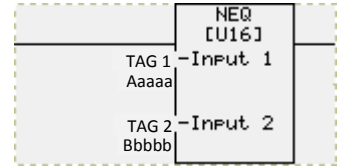
The Equal To instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 = Input 2 then power will flow through this element. Either Inputs can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.



≠ Not Equal to

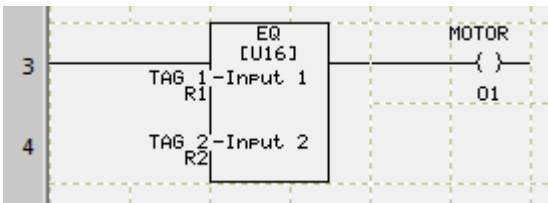
Not Equal To:

The Not Equal To instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 ≠ Input 2 then power will flow through this element. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.

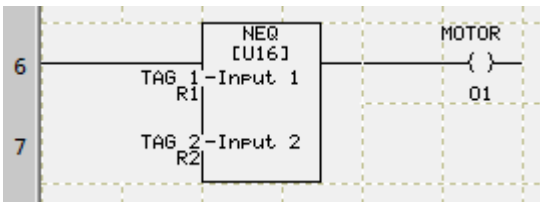


Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, if R1 equals R2, power will flow out of EQ and O1 will be energized.

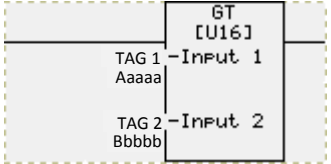


In the example above, if R1 does not equal R2, power will flow out of NEQ and O1 will be energized.

> Greater than

Greater Than:

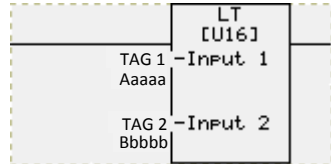
The Greater Than instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 > Input 2 then power will flow through this element. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.



< Less than

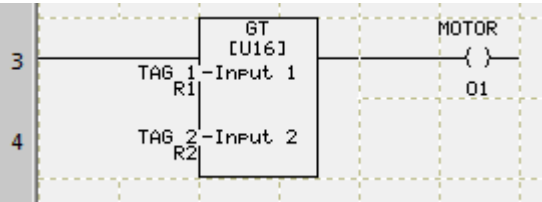
Less Than:

The Less Than instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 < Input 2 then power will flow through this element. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.

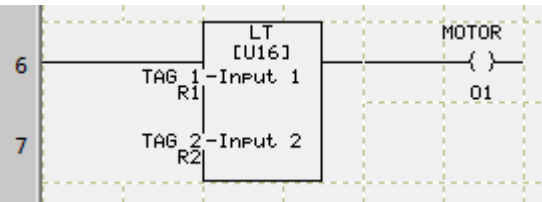


Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, if R1 is Greater Than R2, power will flow out and O1 will be energized.

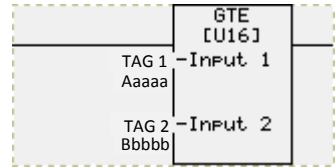


In the example above, if R1 is Less Than R2, power will flow and O1 will be energized.

>= Greater than or equal to

Greater Than Or Equal To:

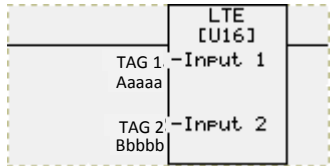
The Greater Than Or Equal To instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 is Greater Than Or Equal To Input 2 then power will flow through this element. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.



<= Less than or equal to

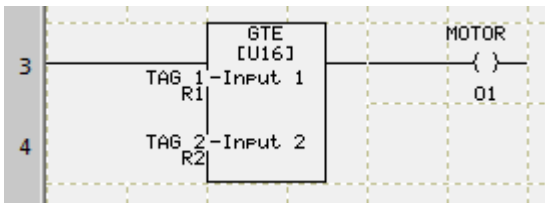
Less Than Or Equal To:

The Less Than Or Equal To instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. If Input 1 is Less Than Or Equal To Input 2 then power will flow through this element. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Input 1 and Input 2 must be of the same data type.

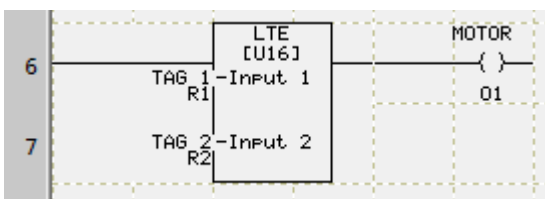


Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, if R1 is Greater Than or Equal To R2, power will flow out and O1 will be energized.

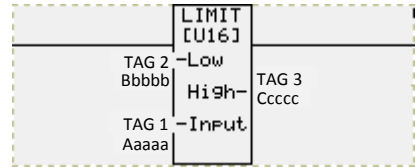


In the example above, if R1 is Less or Equal To R2, power will flow out and O1 will be energized.

LIM Limit

Limit:

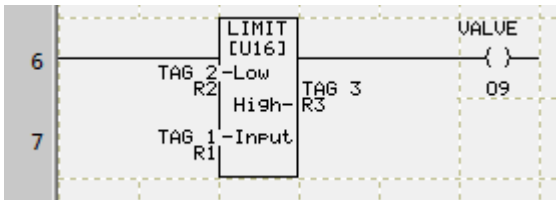
The Limit instruction can be used to compare register data values of the Input at memory location Aaaaa with Low at memory location Bbbbb and High at memory location Ccccc. If $Aaaa \leq Ccccc$ and $Aaaa \geq Bbbbb$ then power will flow through this element. Any of the registers (Input, High or Low) can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



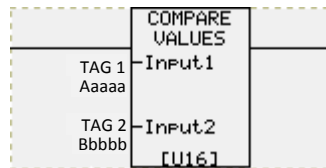
In the example above, if the input R1 is within R2 and R3, power will flow out and O9 will be energized.



Compare Values

Compare Values:

The Compare Values instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. Based on the Input 1 and Input 2 comparison the corresponding discrete tag (Ccccc, Ddddd, or Eeeee) will turn on. This instruction examines whether Input 1 and Input 2 are greater than, equal, or less than. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type. Power constantly flow through this element.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Memory Type	Syntax (C,D, E)	Range (cccc)	Range(ddd)	Range(eeee)
Discrete				
Discrete Outputs	O	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024
Register Discretcs				
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete, all register data type except BCD and ASCII.

Compare Values Instruction

Instruction Details

Compares 2 Inputs and evaluates whether Input 1 (>, <, =) Input 2. Turns on discretes correspondingly.

Inputs

Input 1
 Input 1 Tag: TAG 1
 Data Type: UNSIGNED_INT_16
 Input 2 must be of same data type.

Input 2
 Tag Name: TAG 2
 Constant: (In Decimal)

Results

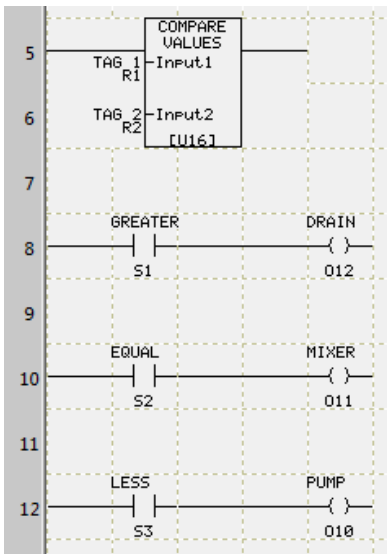
Greater Than Tag (Input 1 > Input 2)
 GREATER

Equal To Tag (Input 1 = Input 2)
 EQUAL

Less Than Tag (Input 1 < Input 2)
 LESS

Display all values in: Decimal

OK Cancel Help



In this example:

- If $R1 > R2$ then S1 will be ON and O12 will be energized.
- If $R1 = R2$ then S2 will be ON and O11 will be energized.
- If $R1 < R2$ then S3 will be ON and O10 will be energized.

3.3.5 Math Instructions

The instructions listed within this chapter perform arithmetical operations on user specified values or addresses. All Math Instructions are always TRUE (that is, power flows through them).

Adding Math Instructions

To configure all of the various Math instructions, perform the following steps:

1. Click on any Math instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

4. Select a Tag name/address from the drop down list for Input 1. Or add a new tag.
5. Select a Tag name/address from the drop down list for Input 2. Or add a new tag.
6. Select a Tag name/address from the drop down list for Result. Or add a new tag.
7. For Absolute, X=Y Conversion, and Binary Conversion instructions, select Source and Destination Tag names/addresses.

8. Choose the correct data format from the last drop down list on dialog box.
9. Data types for all Input 1, Input 2 and Result must be the same (Source and Destination for Absolute and Conversion instructions).

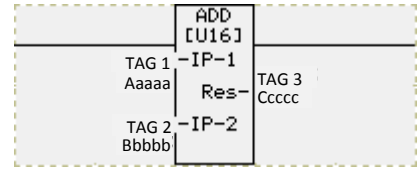
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

+ Add

Add:

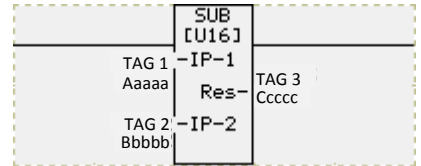
When power flows to this element, the Add instruction adds the register data values of two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. The added value is stored in Result at memory location Ccccc. Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both the Inputs and Result must be of the same data type.



- Subtract

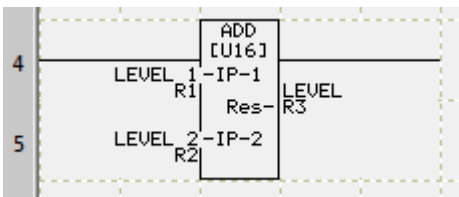
Subtract:

When power flows to this element, the Subtract instruction subtracts the register data value of Input 2 at memory location Bbbbb from Input 1 at memory location Aaaaa. The subtracted value is stored in Result at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both the Inputs and Result must be of the same data type.

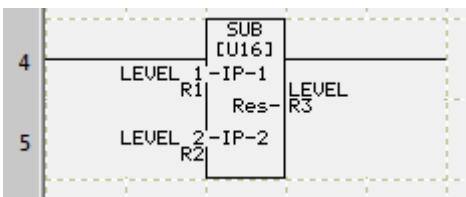


Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, R1 will be added to R2 and the result will be placed in R3.

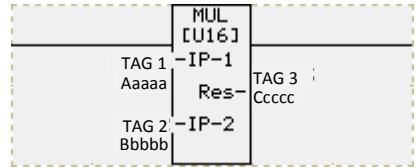


In the example above, R2 will be subtracted from R1 and the result will be placed in R3.

Multiply

Multiply:

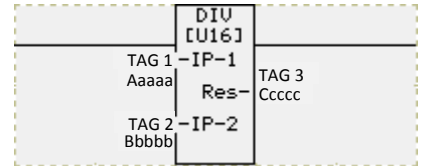
When power flows to this element, the Multiply instruction multiplies the register data values of two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. The multiplied value is stored in Result at memory location Ccccc. Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both the Inputs and Result must be of the same data type.



Divide

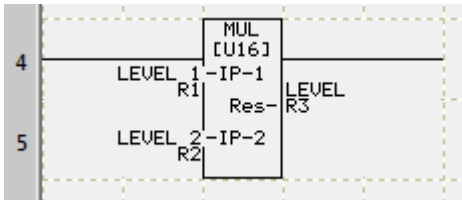
Divide:

When power flows to this element, the Divide instruction divides the register data value of Input 1 at memory location Aaaaa by Input 2 at memory location Bbbbb. The divided value is stored in Result at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both the Inputs and Result must be of the same data type. Power will stop flowing through this element if division by zero is attempted.

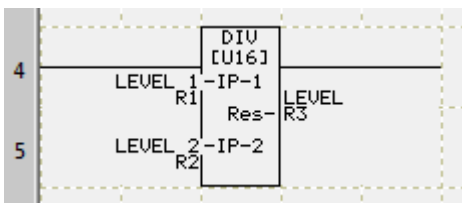


Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, R1 will be multiplied by R2 and the product will be placed in R3.



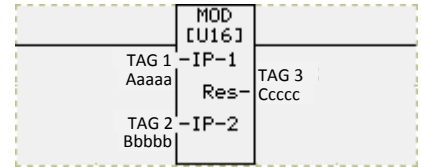
In the example above, R1 will be divided by R2 and the result will be placed in R3.



Modulo

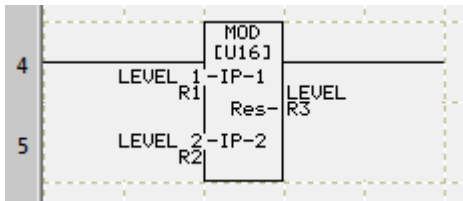
Modulo:

When power flows to this element, the Modulo instruction divides the register data value of Input 1 at memory location Aaaaa by Input 2 at memory location Bbbbb. The Remainder Value is stored in Result at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both the Inputs and Remainder must be of the same data type. Power will stop flowing through this element if division by zero is attempted.



Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

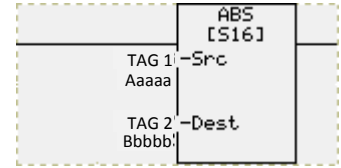
Allowed Data Formats: SIGNED and UNSIGNED data types only.



In the example above, R1 is divided by R2 and only the remainder is placed in R3.

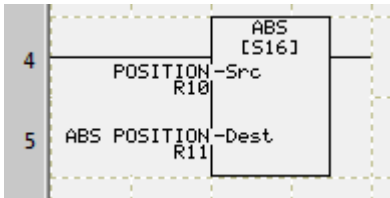
Absolute:

When power flows to this element, the Absolute instruction converts the signed (negative) register data value of Src at memory location Aaaaa to the absolute (positive only) data value and stores it in Dest at memory location Bbbbb. Both Source and Destination must be of the same data type.



Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

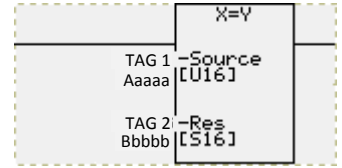
Allowed Data Formats: SIGNED and FLOATING_PT data types only.



In the example above, R11 will contain the Absolute value of R10 (for example, if R10 was -10, R11 will contain +10).

X=Y X=Y Conversion**X=Y Conversion:**

When power flows to this element, the X=Y Conversion instruction converts the register data type of Src at memory location Aaaaa to Res at memory location Bbbbb and copies the converted data value to Res at memory location Bbbbb. If Src has a Floating Point data type it can either be rounded off to the nearest integer value or truncated when converting to other data types. When the integer or floating point data value is converted to an ASCII type data value, the number of digits, decimal position and justification (leading zeros, leading spaces, or trailing spaces) can be assigned as per user.

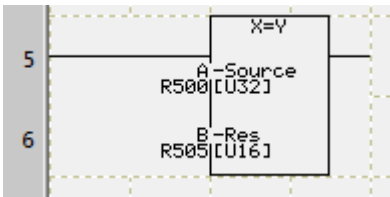


Note: This is the same instruction that can also be found in the Datatype Conversion Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data types.

Note: If converting a signed 16 bit (with a negative value -1) to an unsigned 16 bit register the result will always be zero.



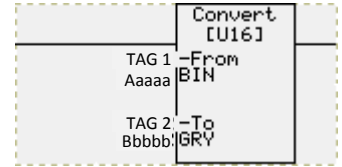
In the example above, variable A, (R500) which is an UNSIGNED_32 (U32) Type, will be converted to an UNSIGNED_16 Type (U16) and saved in B.



Format Conversion

Format Conversion:

When power flows to this element, the Format Conversion instruction converts the data format of From at memory location Aaaaa to To at memory location Bbbbb as follows:



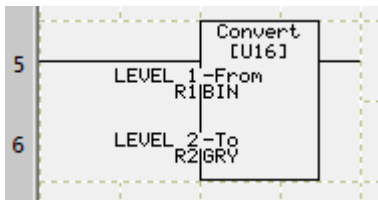
- Binary to BCD
- BCD to Binary
- Binary to Gray Code
- Gray Code to Binary

Both the From and To data types must be a 16 bit Signed Integer, 16 bit Unsigned Integer, or 16 bit BCD for Format Conversion instruction.

Note: This is the same instruction that can also be found in the Datatype Conversion Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: SIGNED_INT_16, UNSIGNED_INT_16, BCD_INT_16

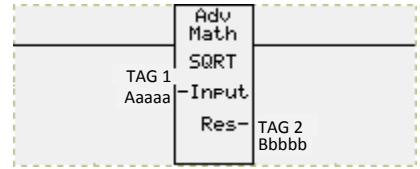


In the example above, R1 which is in Binary format, is converted to Gray Code and saved in R2.



Advanced Math:

When power flows to this element, the Advanced Math instruction performs a selected operation on the register data values of Input at memory location Aaaaaa. The resulting value is stored in Result at memory location Bbbbbb. Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Input can be any data type but the Result must be of type Floating Point. Please see more information below and on next page.



Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

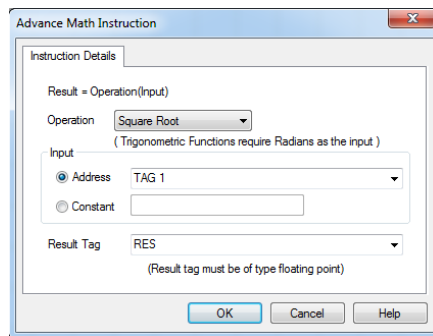
Allowed Data Formats: Input can be all register data types except BCD and ASCII. Result must be FLOATING_PT_32.

Advance Math function allows you to perform following mathematical operations in ladder logic:

- Square Root
- Log (base 10)
- Exponent (e^{Input})
- Natural Log (ln, base e)
- Sin
- Cosine
- Tan
- Inverse Sin
- Inverse Cosine
- Inverse Tan
- Convert to Radians
- Convert to Degrees

Adding Advanced Math Instruction

1. Click on the Advanced Math instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.
4. Select the Operation you would like to perform from the dropdown menu. For more information on operations see descriptions below.
5. Select a proper Tag name/address from the drop down list for Input. Or add a new tag. You can also instead put in a constant.
6. Select a proper Tag name/address from the drop down list for Result. Or add a new tag.
Note: Result tag needs to be FLOATING_PT.



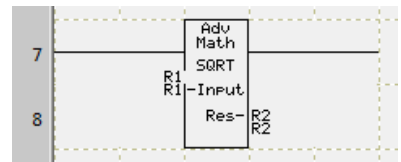
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

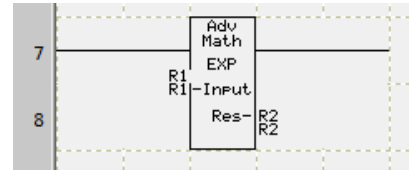
There are several operations under Advance Math Instructions you could use in your PLC program.

1. Square Root & Exponent (e^{Input})

Click on to the drop down button to select **Square root** Operation to see how this works. For example if the Input Tag R1 has the value 25 stored, the Result Tag R2 would have the output to be 5 in this case since square root of 25 is 5. This Justifies **Result = Input**.

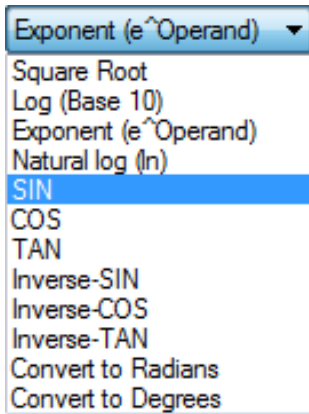


Similarly, if you want to Obtain the **Exponential** of a constant Value, then you would need to have the input Value stored in the Input Tag R1 and the Exponential of the Input Value applied would be stored in the Result Tag R2.

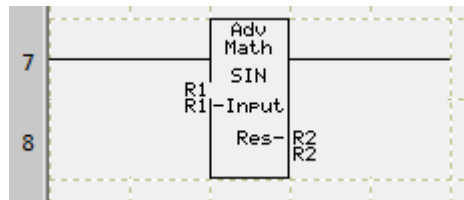


2. Trigonometric Functions:

Note: All Trigonometric functions require radians as the inputs (not degrees); (Convert to Radian function is available to convert degrees to radians)



For Trigonometric functions, click on to the drop down button to select either SIN, COS or TAN Function (sine, cosine, tangent).



Here the Input Tag R1 would have the Value stored in **Radians** and the **Output of the Sine operation is Stored in the Result Tag R2.**

Input Tag Type could be either an Address such as R1 or a constant Value.

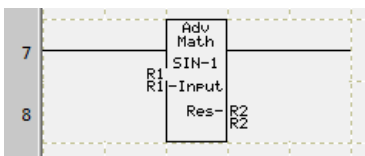
Input Tag Type doesn't necessarily need to be of floating point type but the **Result Tag must be of floating point Type.**

Similarly, you could apply the same steps for Cosine & Tangential functions.

3. Inverse Trigonometric Functions:

Note: Inverse Trigonometric functions return radians in the result tag (not degrees) (Convert to degrees function is available to convert radians to degrees)

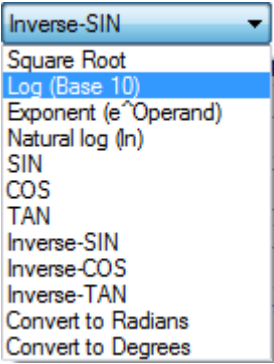
With the Advanced Math Functions, you would be able to calculate the Inverse Sin, Cosine or Tangent of a Value. Shown below is an example of Inverse-SIN to show how the inverse sine operation works.



Input Value in Radians for which the Inverse-SIN operation is to be done is stored in the Input Tag R1 and the result of the sine inverse operation is stored in the Result Tag R2.

Similarly, you would be able to calculate the cosine inverse and Tangential Inverse.

4. Logarithmic Function



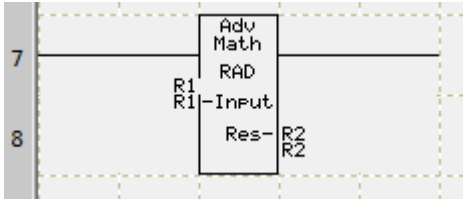
Click on to the drop-down button to select **Log(Base 10)**. Here you could find the Logarithmic function of any value. If the Input Tag R1 has the Value 2, then the Result -> $\text{Log}(2) = 0.414$ is Stored in the Result Tag R2.

Similarly, you could obtain the **Natural Logarithm** (Inverse function of the exponential function) of any value by following the same steps with the result getting stored in the Result Tag.

5. Convert to Radians and Convert to Degrees:

Here is the formula to **convert degrees to radians**

$$\text{Angle in radians} = \text{Angle in degrees} * \text{Pi} / 180$$



Input Tag R1 would have the Value in Degrees stored and the Output in Radians would be stored in the Result Tag R2.

The Formula to **Convert Radians to Degrees** is

$$\text{Angle in degrees} = \text{Angle in radians} * 180 / \text{Pi}$$

3.3.6 Bit Logic Instructions

Bitwise Instructions operate on 16-bit or 32-bit SIGNED and UNSIGNED data types. Operations are performed on the bit patterns of two registers. After the operation, the results are stored in a third register (Res). Neither input is changed.

Adding Bit Logic Instructions

To configure all of the various Bitwise instructions, perform the following steps:

1. Click on any Bitwise instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

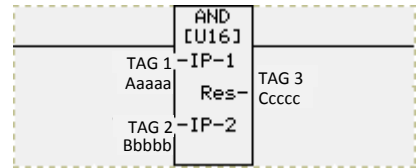
4. Select a proper Tag name/address from the drop down list for Input 1. Or add a new tag.
5. Select a proper Tag name/address from the drop down list for Input 2. Or add a new tag.
6. Select a proper Tag name/address from the drop down list for Result. Or add a new tag.
7. Choose the correct data format from the last drop down list on dialog box.
8. Data types for all Input 1, Input 2 and Result must be the same.

Adding tags:

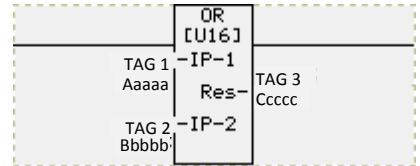
- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

AND AND**AND:**

When power flows through this element, the AND instruction performs a bitwise AND operation on data values of Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb and stores the output in Res at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Res must be of the same data type.

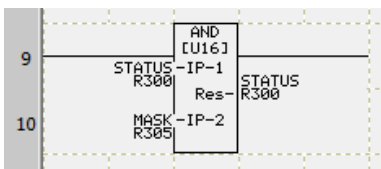
**OR** OR**OR:**

When power flows through this element, the OR instruction performs a bitwise OR operation on data values of two registers Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb and stores the output in Res at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Res must be of the same data type.



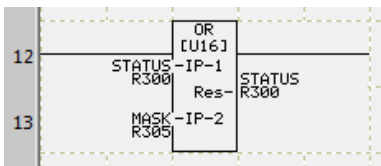
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: *SIGNED_INT_16*, *SIGNED_INT_32*, *UNSIGNED_INT_16*, *UNSIGNED_INT_32*.



STATUS = 0001 0011 0101 0111
 MASK = 0000 1111 0000 0000
 STATUS After AND:
 0000 0011 0000 0000

In the example above, Status in R300 is ANDed with MASK in R305 and the result is stored in Status.

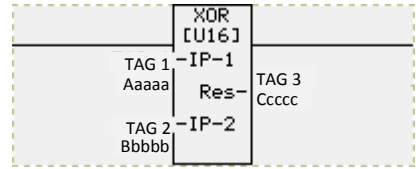


STATUS = 0001 0011 0101 0111
 MASK = 0000 1111 0000 0000
 STATUS After OR:
 0001 1111 0101 0111

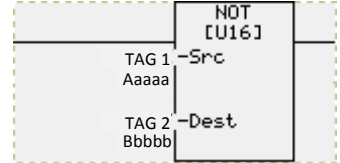
In the example above, Status in R300 is ORed with MASK in R305 and the result is stored in Status

XOR XOR**XOR:**

When power flows through this element, the XOR instruction performs a bitwise XOR operation on data values of two registers, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb and stores the output in Res at memory location Ccccc. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Result must be of the same data type.

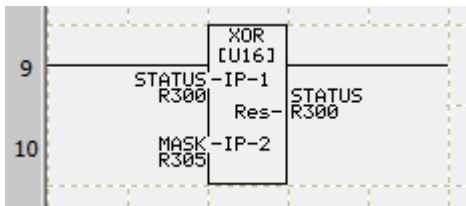
**NOT** NOT**NOT:**

When power flows through this element, the NOT instruction performs a bitwise NOT operation on data value of Src at memory location Aaaaa and stores the output in Dest at memory location Bcccc. Src can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Source and Destination must be of the same data type.



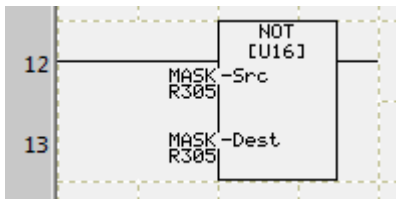
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: *SIGNED_INT_16*, *SIGNED_INT_32*, *UNSIGNED_INT_16*, *UNSIGNED_INT_32*.



```
STATUS = 0001 0011 0101 0111
MASK = 0000 1111 0000 0000
MASK After XOR:
0001 0000 0101 0111
```

In the example above, Status in R300 is XORed with MASK in R305 and the result is stored in Status.



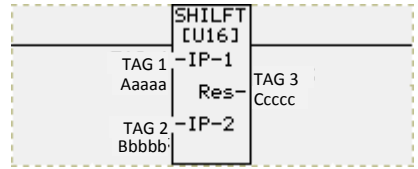
```
MASK = 0000 1111 0000 0000
MASK After NOT:
1111 0000 1111 1111
```

In the example above, the MASK is inverted and saved back in MASK



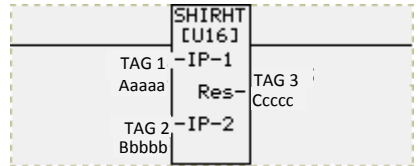
Shift Left:

When power flows through this element, the Shift Left instruction performs a Logical Shift Left on Input 1 at memory location Aaaaa by the value of Input 2 at memory location Bbbbb and stores the result in Res at memory location Ccccc. No bits are shifted into the right and any bits shifted from the left are lost. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Res must be of the same data type.



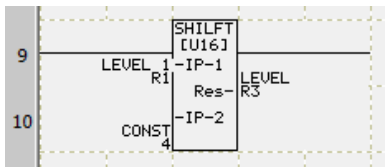
Shift Right:

When power flows through this element, the Shift Right instruction performs a Logical Shift Right on Input 1 at memory location Aaaaa by the value of Input 2 at memory location Bbbbb and stores the result in Res at memory location Ccccc. No bits are shifted in from the left and any bits shifted from the right are lost. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Result must be of the same data type.



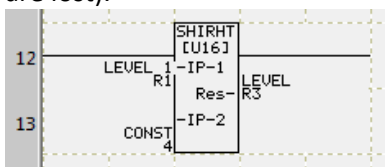
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: *SIGNED_INT_16*, *SIGNED_INT_32*, *UNSIGNED_INT_16*, *UNSIGNED_INT_32*.



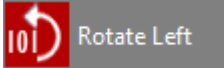
R1 = 1100 0000 0000 0101
 Shift Left by = 4
 R3 after shift = 0000 0000 0101 0000

In the example above, the value of Level is shifted Left by 4. All bits are shifted left by 4 (MS bits are lost).



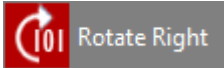
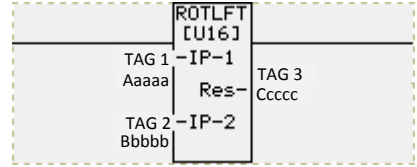
R1 = 1100 0000 0000 0101
 Shift Right by = 4
 R3 after shift = 0000 1100 0000 0000

In the example above, the value of Level is Shifted Right by 4. All bits are shifted right by 4 (LS bits are lost).



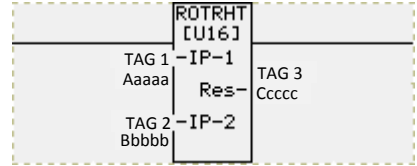
Rotate Left:

When power flows through this element, the Rotate Left instruction performs a logical Rotate Left on Input 1 at memory location Aaaaa by the value of Input 2 at memory location Bbbbb and stores the result in Res at memory location Ccccc. Bits are rotated into the right and any bits shifted from the left are rotated in. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Result must be of the same data type.



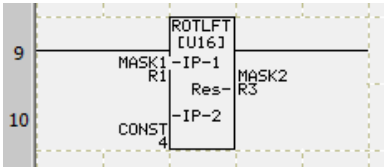
Rotate Right:

When power flows through this element, the Rotate Right instruction performs a logical Rotate Right on Input 1 at memory location Aaaaa by the value of Input 2 at memory location Bbbbb and stores the result in Res at memory location Ccccc. Bits are rotated into the left and any bits shifted from the right are rotated in. Input 1 and Input 2 can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. The Inputs and Result must be of the same data type.



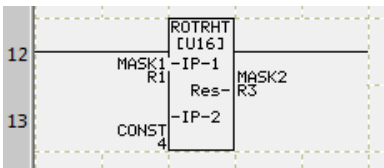
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: *SIGNED_INT_16, SIGNED_INT_32, UNSIGNED_INT_16, UNSIGNED_INT_32.*



MASK1 = 1100 0000 0000 0101
 Rotate Right by = 4
 MASK2 after shift = 0000 0000 0101 1100

In the example above, MASK1 is rotated right by 4 and saved in MASK2.



MASK1 = 1100 0000 0000 0101
 Rotate Right by = 4
 MASK2 after shift = 0101 1100 0000 0000

In the example above, MASK1 is rotated left by 4 and saved in MASK2.

3.3.7 Move Instructions

Move Instructions allow the movement of data between registers. Move based instructions can also be used to move constant values into registers, move blocks of data from one location to another, or to fill a block of registers with the same value.

Power Flow

Move instructions are always true so power flow always passes through the rung. The exception to this is the Indirect Move Element. In this case, the move is considered invalid and power flow is false if either the source or destination register contains 0 (zero) or the length of the move exceeds the number of elements available in the controller.

Adding Move Instructions

To configure all of the various Move Instructions, perform the following steps:

1. Click on a Move instruction icon on the right side of the screen.
2. Position the mouse over the area on the Ladder diagram where you want to insert the instruction and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

4. For a Bit Move instruction, choose if you want to move register bits to discrete or vice versa.
5. Enter a Tag Name in the Data Type field or use the drop arrow to make your selection.
6. Select a proper Tag name/address from the drop down list for Source. Or add a new tag.
7. Select a proper Tag name/address from the drop down list for Destination. Or add a new tag.

8. Enter the number of elements to move/fill (only for the Move Block and Block Fill instructions).

9. Enter the numeric constants in the Table of Constants and select a proper Tag name/address for the Destination from the drop down list (only for the Move table of Constants instruction).

10. Choose the correct data format from the last drop down list on dialog box.
11. Data types for both source and destination must be the same.

Adding tags:

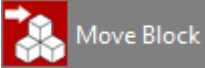
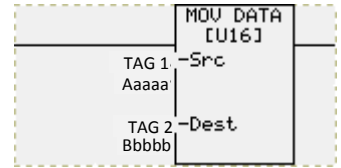
- If need be you can add a new tag by entering a new Tag Name
- Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- Enter the Tag Address in this screen.



Move Data

Move Data:

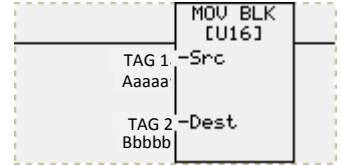
When power flows through this element, the Move Data instruction moves data value from Src at memory location Aaaaa to Dest at memory location Bbbbb. Src can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Src and Dest must be of the same data type.



Move Block

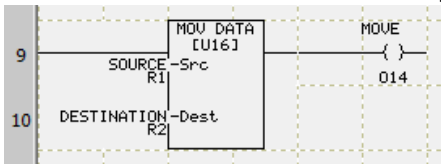
Move Block:

When power flows through this element, the Move Block instruction moves a block of memory area. Src at memory location Aaaaa provides the starting address of the memory area to move from and Dest at memory location Bbbbb provides the starting address of the memory area to move to. The number of elements to move is user specified. The maximum number of elements that can be moved with one Move Block instruction is 128 for 16 Bit registers and 64 for 32 Bit registers. Src can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Src and Dest must be of the same data type.

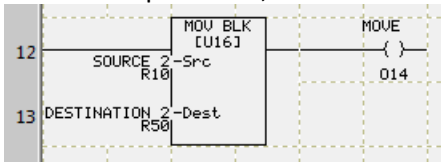


Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384

Allowed Data Formats: all register data type except BCD and ASCII



In the example above, R2 = R1 after the move.



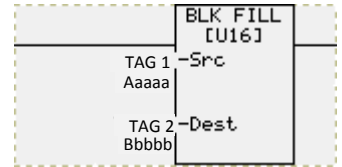
This instruction is used to copy multiple elements. In this example, 10 registers starting from R10 (R10-19), are copied to R50-49.



Block Fill

Block Fill:

When power flows through this element, the Block Fill instruction fills a block of memory area. Src at memory location Aaaaa provides the data value to fill with; whereas Dest at memory location Bbbbb provides the starting address of memory area to fill to. The number of elements to fill is user specified. The maximum number of elements that can be filled with one Block Fill instruction is 128 for 16 Bit registers and 64 for 32 Bit registers. Src can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. Both Source and Destination must be of the same data type.



Block Fill Instruction

Instruction Details

Fill a block of memory area with the value from Source. The block starts at the address of "Destination", and continues for numbers of elements.

Source (Src)

Tag Name SOURCE

Constant (In Decimal)

Data Type: UNSIGNED_INT_16 Destination must be of same data type

Number of elements to move: 10

Destination (Dest): DESTINATION

Display all values in: Decimal

OK Cancel Help

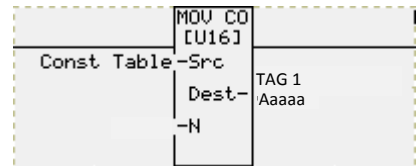


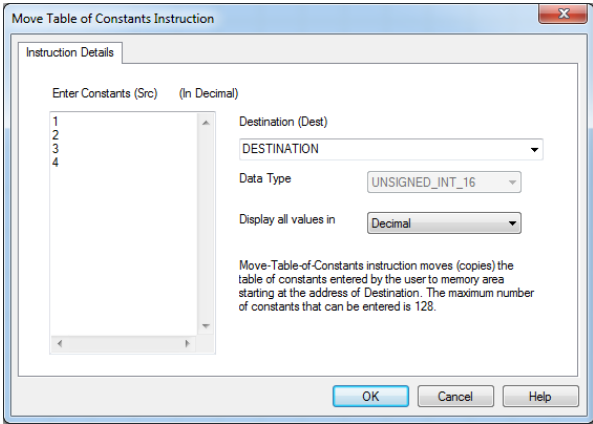
Move Table of Constants

Move Table of Constants:

When power flows through this element, the Move Table of Constants instruction loads user specified table of constants to consecutive memory addresses with the starting memory address defined by Dest at memory location Aaaaa. Src is the user specified table of constants.

The maximum number of constants that can be moved are 128 for 16bit registers and 64 for 32bit registers. N displays the number of Dest addresses occupied by the user specified table of constants. Source and Destination must be of the same data type.

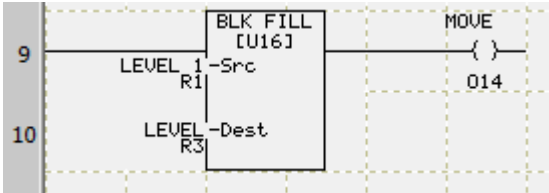




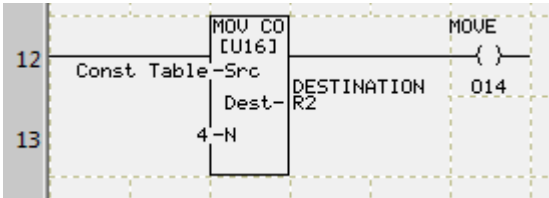
REAL numbers less than zero must contain a leading zero (e.g., .999 is not valid, 0.999 is valid). It is possible to copy and paste data to/from other Windows applications including Microsoft Excel and Word. Constants are placed one on each individual line.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384

Allowed Data Formats: all register data type except BCD and ASCII.



In the example above, Value of R1 is copied to 10 registers starting with register R3 (the number of elements in instruction is specified as 10).



In the example above, a table of constant is copied to registers starting with R2. Number of elements are shown as N (4 in this case).



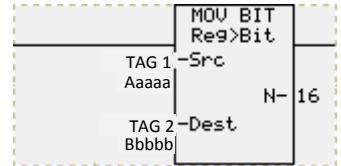
Move Bit

Move Bit:

When power flows through this element, the Move Bit instruction can either copy bits from a maximum of 16 contiguous discrete bits to a single 16-bit word register or a single 16-bit word register to a maximum of 16 contiguous discrete bits. The two available modes are available as follows:

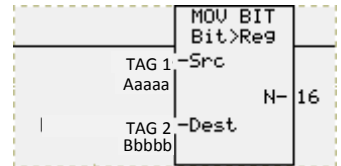
- Map Register Bits to Discretes

When using the Move Bit instruction to map register bits to discretes, Src at memory location Aaaaa provides the address of the register from which the bits are to be moved. The Number of Bits selected by you defines the total number of consecutive bits which are to be moved starting from the Src address location. Dest at memory location Bbbbb provides the address of the register where bits from Src are being moved to. The user selectable Start Bit Number specifies the bit location in Dest register where onwards the bits are to be moved in.



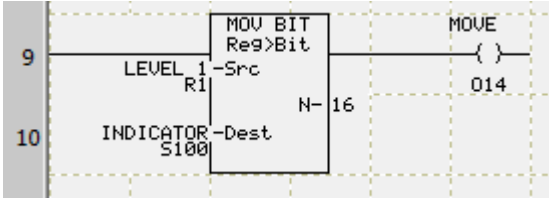
- Map Discretes to Registers

When using the Move Bit instruction to map discretes to registers, Src at memory location Aaaaa provides the address of the register where bits are to be moved from. The user selectable Start Bit Number specifies the starting point in the register where onwards the bits are to be moved and the Number of Bits specify the total number of bits to be moved. Dest at memory location Bbbbb provides the starting address for consecutive bits which are being moved into from Src register.



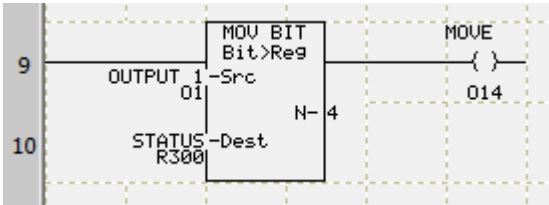
Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Discrete			
Discrete Inputs	I	1-128	1-128
Discrete Outputs	O	1-128	1-128
Discrete Internals	S	1-1024	1-1024
System Discretes	SD	1-16	1-16
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.			

Allowed Data Formats: SIGNED_INT_16, UNSIGNED_INT_16, BCD_INT_16, Discrete



In the example above, all 16 bits (N=16) of R1 are copied to Scratch Bits S100 to S115. The Least significant bit of R1 is moved to S100, and the most significant to S115. Value moved of register R1 is 255.

Discrete Internals (S)	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100
Value	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1



In the example above, 4 bits (N=4) starting from O1 are copied to Status Tag (R300).

R300 Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value of													04	03	02	01

3.3.8 Timer/Counter/Drum Instructions

Timer, Counter and Drum instructions allow you to control operations based on time or number of events.

Adding Timer Instruction

To configure the Timer instruction, perform the following steps:

1. Click the Timer instruction icon side of the screen.
2. Position the mouse over the area on the Ladder diagram where you want to insert the Timer instruction and click the mouse to place it.
3. To configure Timer types, double click the Timer instruction to open its dialog box.

TimerTags	Tag Name	Address
Accumulated Value	TIMER.ACC	R1
Reset Bit	TIMER.RST	R2/13
Timer Enable Bit	TIMER.EN	R2/14
Timer Done Bit	TIMER.DN	R2/15
Preset Value	TIMER.PRE	R3

4. Check the box for desired Timer Action, Delayed ON, Delayed OFF or Delayed ON-Retentive Counts.

5. Select one of the Time Base options.

6. Select a proper Tag name/address from the drop down list for the Timer tag. Or add a new tag.

7. The Timer instruction will auto generate Timer Tags which can be used for timer control. *Note: It will use 2 register address right after the Timer address (even if they are used elsewhere).*

8. Check the Constant Preset Value if you would like to have it constant.

Adding tags:

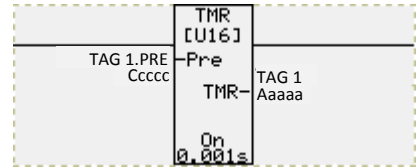
- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Timer

Timer Instruction:

When power flows to this element, the Timer instruction starts timing. Once it reaches the Preset Value as defined by the Timer Preset register, it will stop timing and either allow power flow or stop power flow based on the type of Timer instruction used. When using a Retentive timer, you must use a Reset bit to reset the timer. When using a Non-Retentive timer then the timer will reset each time power flow is stopped.



Timer Instruction

Instruction Details

Timer Action

Delayed ON

Delayed OFF

Delayed ON- Retentive Counts

Time Base

1 Millisecond

10 Milliseconds

100 Milliseconds

1 Second

Timer Register

TIMER

Constant Preset Value

TimerTags	Tag Name	Address
Accumulated Value	TIMER.ACC	Aaaaa
Reset Bit	TIMER.RST	Bbbbb/13
Timer Enable Bit	TIMER.EN	Bbbbb/14
Timer Done Bit	TIMER.DN	Bbbbb/15
Preset Value	TIMER.PRE	Ccccc

OK Cancel Help

Timer Register and Timer Status Register:

TIMER at memory location Aaaaa defines the timer register. The TIMER.ACC is also at location Aaaaa. The Timer instruction will automatically create tags at memory locations Bbbbb and Ccccc where Bbbbb = A(aaaa+1) and Ccccc = A(aaaa+2). The TIMER.EN at Bbbbb/14 is the timer enable bit which will be on when the timer is timing. Also the TIMER.DN at Bbbbb/15 is the timer done bit which will be on when the timer is done.

Timer Preset Value (TIMER.PRE):

Preset at memory location Ccccc defines the timer preset value, it is also automatically created as soon as TIMER is defined at location Aaaaa. Preset can also be assigned a constant value. The Timer preset value allows the Timer instruction to count to a certain value based on the Time Base Selected.

Timer Reset (TIMER.RST)

The timer reset bit at memory location Bbbbb/13 defines the reset input bit for the Retentive Timer ONLY. As soon as the TIMER.RST turns ON the TIMER.ACC will become zero till the TIMER.RST is turned OFF. *Note: For other types of timers the reset bit exists but does not do anything. The other types of timers are automatically reset when they are not timing.*

Time Base:

The Time Base is user selectable and allows one of the following time bases:

- 1 Millisecond
- 10 Millisecond
- 100 Millisecond
- 1 Second

e.g. If Preset = 15 and Time Base = 10 Millisecond, then the Timer instruction will time for 150 Milliseconds. Similarly, if Pre =11 and Time Base = 100 Millisecond, then the Timer instruction will time for 1100 Milliseconds.

Timer Action

- Delayed ON
- Delayed OFF
- Delayed ON-Retentive Counts

Types of Timer:

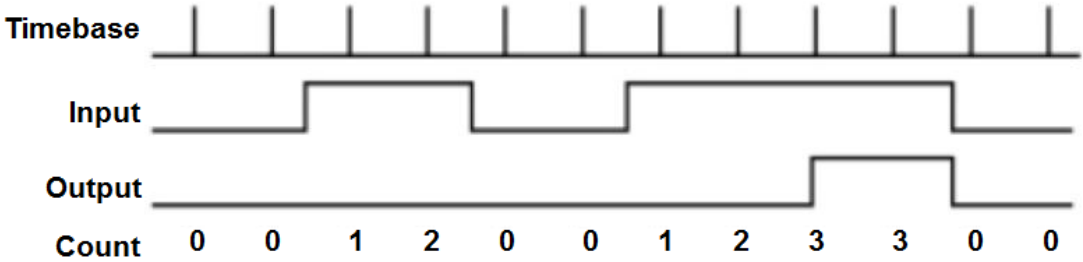
There are three types of Timers available as specified by you: Delayed ON, Delayed OFF, and Delayed ON-Retentive Counts.

Delayed ON: When power flows to this type of Timer it starts timing until it reaches the Timer Preset Value. Once it

completes the specified count, it allows power flow through this element. If power flow to this Timer stops before it reaches the Timer Preset value, it resets itself to zero and starts timing from 0 when power flows to this instruction again.

Delayed ON Timing Diagram

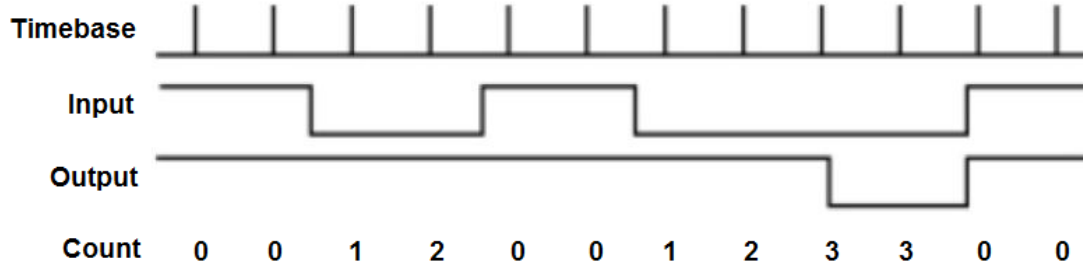
Shows Timer with Delayed ON that counts to 3 when power flows to Timer Instruction.



Delayed OFF: This type of Timer allows power flow though it as long as power flows to this element. When power flow STOPS to this type of timer, it still allows power flow through it and starts counting at the same time. When the Timer reaches the Timer Preset Value, it STOPS the power flow through it. If power flows back to this Timer before it reaches the Timer Preset value, it resets itself and starts timing from 0 again anytime power flow stops to it.

Delayed OFF Timing Diagram

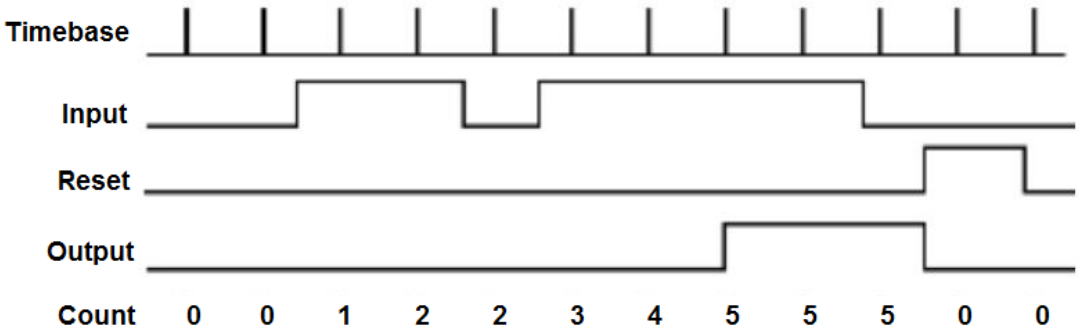
Shows Timer with Delayed OFF that counts to 3 when power flows to Timer Instruction stops.



Delayed ON – Retentive Counts: When power flows to this type of timer it starts timing until it reaches the Timer Preset Value. Once it completes the specified count, it allows power flow through this element. If power flow to this timer stops before it reaches the count, it retains the count and starts from the point where it had stopped timing. Once it reaches the Timer Preset Value it will allow power flow through it. This remains true unless the Reset Input Bit is toggled, at which point it resets itself and starts timing whenever power flows to it.

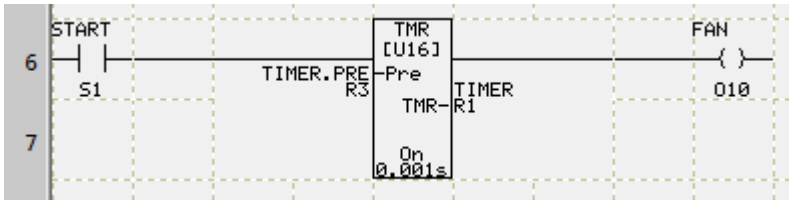
Delayed ON – Retentive Counts Timing Diagram

Shows Timer with Delayed ON — Retentive Counts that counts (retentively) to 5 when power flows to Timer Instruction.



Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Discrete				
Bit Access Registers*	R		(aaaa + 1) / 13-15	
Registers				
Registers Internals	R	1-16384		(aaaa + 2)
*Bit level access to registers is possible. Timer instruction will auto generate these discrete bit access registers and they can be used to control/monitor the Timer in the rest of your ladder logic.				

Data formats supported: UNSIGNED_INT_16

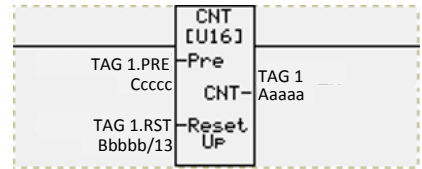


In the example above, the timer is an ON timer, with 0.001's time base. The preset value is R3, and accumulated value in R1. If R1=1000, then once timer is enabled (S1 is ON), it will timer for 1000x0.001=1s then power will flow out of it energizing the fan.



Counter Instruction:

When called, the Counter instruction will count up or down by increments of one until the counter reaches the data value of the Preset Value register. The counter will then allow power flow through the rung.



Adding Counter Instruction

To configure the Counter instruction, perform the following steps:

1. Click the Counter instruction icon side of the screen.
2. Position the mouse over the area on the Ladder diagram where you want to insert the Counter instruction and click the mouse to place it.

Counter Tags	Tag Name	Address
Accumulated Value	COUNTER.ACC	R1
Reset Bit	COUNTER.RST	R2/13
Timer Enable Bit	COUNTER.EN	R2/14
Timer Done Bit	COUNTER.DN	R2/15
Preset Value	COUNTER.PRE	R3

3. To enter Preset/Counter types, double click the Counter instruction to open its dialogue box.

4. Check the box for desired Counter Action (Up or Down).

5. Select a proper Tag name/address from the drop down list for the Counter tag. Or add a new tag.

6. The Counter instruction will auto generate Counter Tags which can be used for counter control. *Note: It will use 2 register address right after the Counter address (even if they are used elsewhere).*

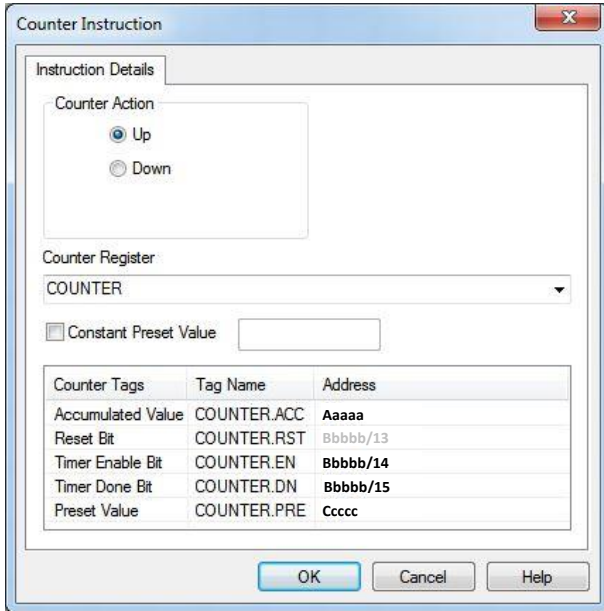
7. Check the Constant Preset Value if you would like to have it constant.

Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

Counter:

When the power flow to this element is switched from OFF (0) to ON (1), this instruction keeps track of the number of times power flow switches. Once it reaches its specified preset, it allows power flow through it.



Counter Register and Counter Status

Register: COUNTER at memory location Aaaaa defines the Counter register value. The COUNT.ACC is also at location Aaaaa. The Counter instruction will automatically create tags at memory locations Bbbbb and Ccccc where Bbbbb = A(aaaa+1) and Ccccc = A(aaaa+2). The COUNTER.EN at Bbbbb/14 is the counter enable bit which will be ON when the counter is counting (COUNT.ACC not equal to COUNT.PRE). Also the COUNT.DN at Bbbbb/15 is the counter done bit which will be on when the counter is done (COUNT.ACC equal to COUNT.PRE).
Note: If counter is using 32 bit register then memory addresses used are Bbbbb = A(aaaa+2) and Ccccc = A(aaaa+4).

Counter Preset Value (COUNT.PRE):

Counter preset at memory location Ccccc defines the Counter Preset Value, it is also automatically created as soon as COUNT is defined at location Aaaaa. This is the value that the counter will increment to or decrement from. Counter preset can occupy a 16 or 32 bit register and can also be assigned a constant value.

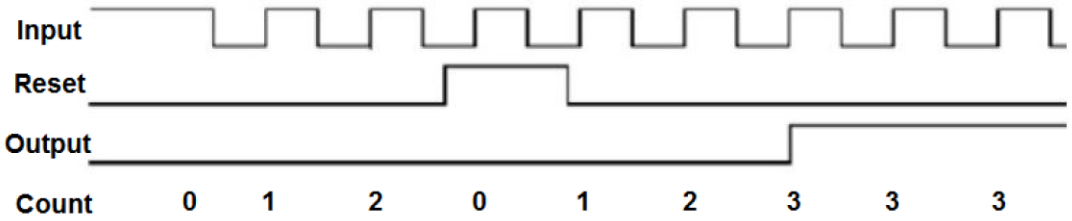
Reset Input Bit (COUNT.RST):

Reset at memory location Bbbbb/13 defines the Reset Input Bit for the Counter instruction. When this bit is enabled, the Counter instruction is reset to its default value based on the type of Counter instruction being used. There are two types of counters, Up Counter and Down Counter, which are user selectable as follows:

Up Counter: When the Reset Input Bit is disabled (0) and the power flow to the counter instruction switches from 0 to 1, the count register increments one count. When the Counter Preset Value and Counter register value become equal, power flows through it. Whenever the reset input is enabled the Counter register value is set to 0 and the power flow through it is stopped.

Up Counter Counting Diagram

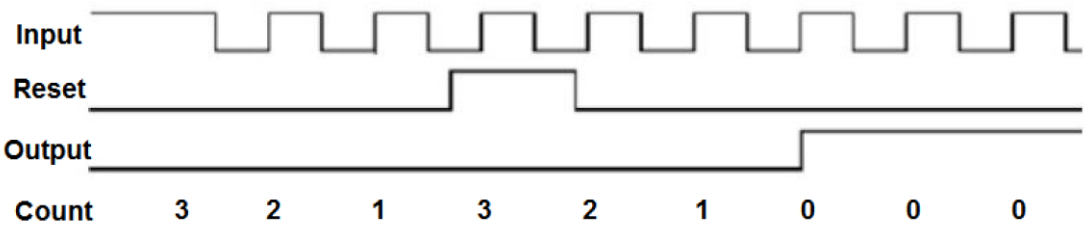
Counts up to the preset of 3. Reset will reset value to 0. Once preset reached output is ON and counter will no longer increment.



Down Counter: When the Reset Input Bit is disabled (0) and the power flow to the counter instruction switches from 0 to 1, the count register decrements one count. When the Counter Preset Value and Counter register value become equal, power flows through it. Whenever the Reset input is enabled the Counter register value is set to Counter Preset Value and the power flow through it stops.

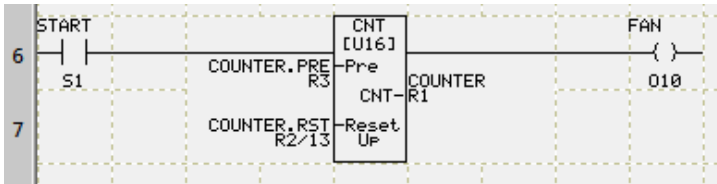
Down Counter Counting Diagram

Counts down from preset of 3. Reset will reset value to 3. Once zero reached output is ON and counter will no longer increment.



Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Discrete				
Bit Access Registers*	R		(aaaa + 1) / 13-15	
Registers				
Registers Internals	R	1-16384		(aaaa + 2)
<i>*Bit level access to registers is possible. Counter instruction will auto generate these discrete bit access registers and they can be used to control/monitor the Counter in the rest of your ladder logic.</i>				

Allowed Data Formats: UNSIGNED INT 16, UNSIGNED INT 32



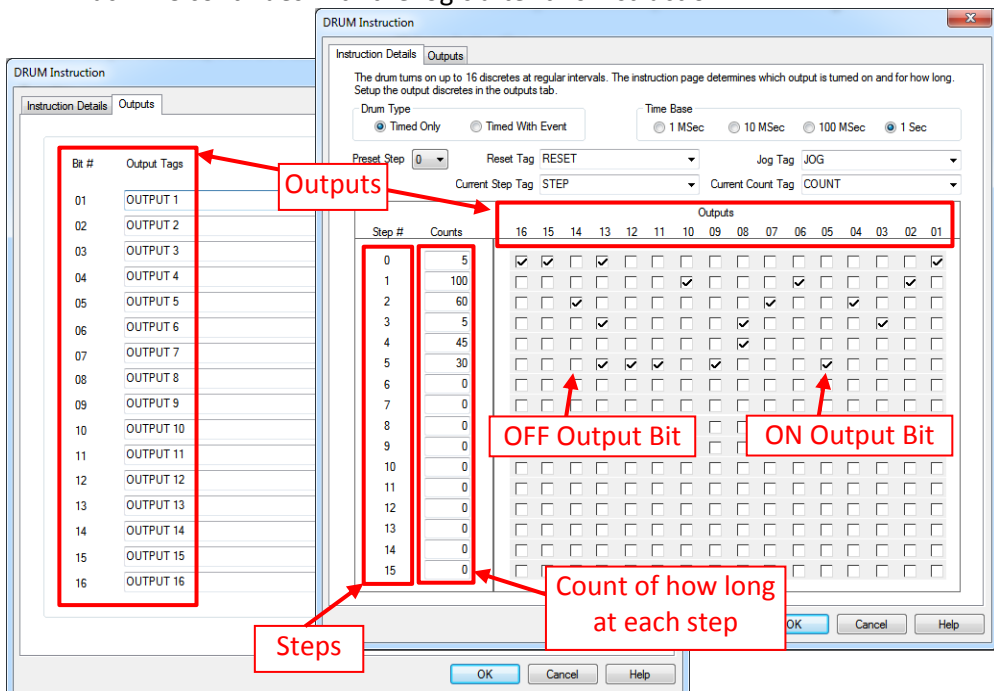
In this example, the counter is a 16 bit UP counter [U16]. Once the count value = Preset value, the power flows out of the instruction.

Drum Instruction

Introduction to Drum Sequencing

Conventionally, electro-mechanical drums are used in control of processes where a certain number of steps is repeated over time. Such drums are a popular control technique because they save a lot of logic programming. Drum sequencing instruction in EZRack PLC mimics the electro-mechanical drums. There are 2 types of Drums, 1) Timed and 2) Timed with Event.

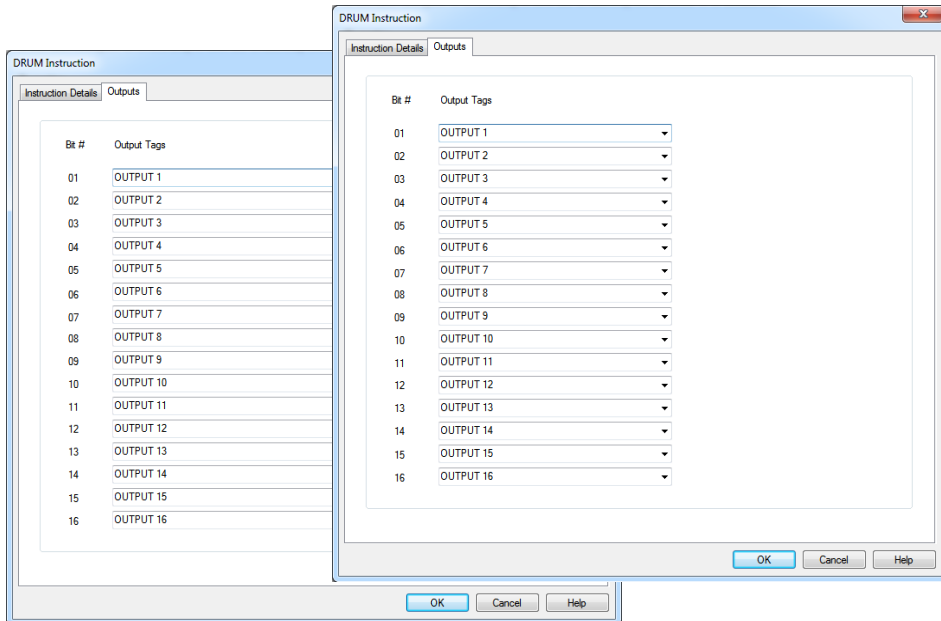
- Each row on a drum chart represents a step on the drum. When rung power condition is true the drum resets to a particular reset step defined by the user.
- Each column in a drum chart represents an output from the drum. We can have 16 discrete outputs numbered from 1 to 16. The outputs are updated during each step.
- The Drum advances from one step to the next per the timer or after triggered by an external event. A Jog tag can also be used to control the drum movement.
- Checked boxes on the drum chart mark ON states of outputs on a particular step. Empty boxes represent OFF outputs.
- Each Drum sequences up to 16 steps having 16 discrete outputs per step.
- Counts have a specified time base and every step has its own counter along with an event to trigger the count.
- When power flows through this element, the Drum instruction starts its sequence while EZRack PLC continues with the logic after this instruction.



Adding the Drum Instruction:

To configure the Drum instruction, perform the following steps:

1. Click on the Drum icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. Double click the instruction to open its dialog box.
4. Click on the Outputs tab on the top to define your Output bits. Add or select tags for total outputs used (Min: 1 and Max: 16)
5. Return to the main dialog box by clicking onto the Instruction Details tab.



6. Select the Drum type (timed or timed with event).
7. Select the Preset Step (default preset step is 1).
8. Choose the Time base (1 ms, 10 ms, 100 ms or 1s).
9. Select or Add Reset, Current Step, Current Count and Jog tags.
10. Define counters for each step.
11. Check the ON-Off states of outputs in each step.

Adding tags:

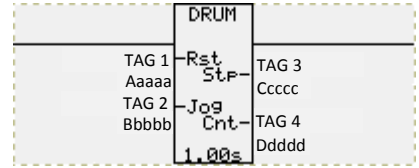
- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Drum

Drum

When power flows through this element, the Drum instruction starts a sequence of outputs which can be either Timed or Timed with Event. The maximum number of steps that can be defined in a sequence is 16 with the maximum number of outputs per step being 16 as well.



Drum Type:

There are two types of user selectable Drum types:

- Timed Only

When you select this option, the Drum instruction completes its sequence based on time specified by Count with specified Time Base only. When the Count is completed, it enables (1) or disables (0) the specified outputs as selected by the user through checkboxes.

- Timed with Event

When you select this option, the Drum instruction completes its sequence based on the time specified by Count with specified Time Base and Events. When this selection is chosen, a tab for Events is available for you to select the desired addresses for events for every step.

Step #:

If using the Timed Only Drum instruction, the total number of programmable steps is 16. When using the Timed with Event Drum instruction then the total number of programmable steps is limited to 10.

Counts:

Every Step has a Count associated with it. The Count is a user specified constant which controls the duration of time before a certain step is executed. The Count can have a different time base as specified by the user in Time Base.

Time Base:

Time Base allows the Count variable to be mapped to different Time Bases as follows:

- 1 millisecond
- 10 millisecond
- 100 millisecond
- 1 second

If the Time Base is set to 1 millisecond, then a Count value of 10 would correspond to 10 milliseconds. Similarly, if the Time Base is set to 10 milliseconds, then a Count value of 10 would correspond to 100 milliseconds and so on.

Preset Step:

This user selected value is used in conjunction with the Reset Tag. If the Reset Tag is enabled (1) then the drum sequence jumps to the step specified by the Preset Step.

Reset Tag:

Rst address at memory location Aaaaa is used to reset the drum sequence to a user selected Step location every time the Rst bit transitions from disable (0) to enable (1). When Rst is enabled, the Drum Sequence is immediately shifted to the Preset Step regardless of its current position and Count value.

Jog Tag:

Jog address at memory location Bbbbb is used to jog the Drum Sequence to the next step. If present on Step 16, it will be jogged to step 1. When Jog is enabled, the Drum Sequence is immediately shifted to the next step regardless of its current position and Count value.

Current Step Tag:

Stp address at memory location Cccc is used by the Drum instruction to write the current value of Step where Drum Sequence exists at any given time during its operation.

Current Count Tag:

Cnt address at memory location Dddd is used by the Drum instruction to write the current value of Count where the Drum Sequence exists at any given time during its operation.

Outputs:

The total number of Outputs that can be used per Step is 16 which is reduced to 10 when using the Timed with Event type Drum instruction. Every Output utilized in any step must have a Discrete memory location assigned to it. Memory locations are assigned in the second tab when adding a Drum instruction. During Drum instruction operation, if the checkbox corresponding to a certain Output is checked, it will be enabled, otherwise it is disabled.

Events:

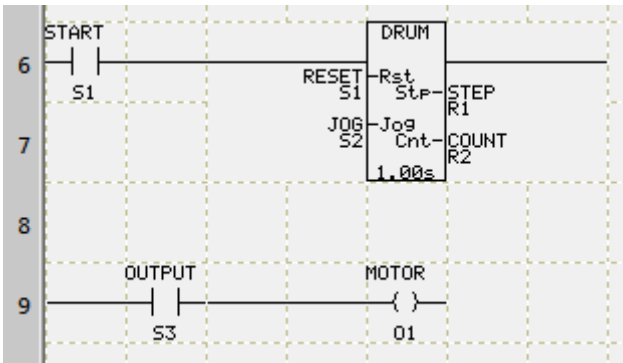
This is an optional tab which only appears if the Timed with Event type Drum instruction is used. For every Step utilized in the Time and Event type Drum instruction, there must be a corresponding Event address assigned to a discrete bit. During Drum Sequence, after the time corresponding to a certain Step is elapsed, the instruction looks at the corresponding Event address. If enabled, Drum Sequence will advance to the next step; otherwise it will start the Count again for the same Step. Once the Count is elapsed it will look again at the Event address to see if it's enabled. If enabled, it will move forward to the next Step, otherwise it will repeat until the corresponding Event address is enabled.

Memory Type	Syntax (A, B)	Range (aaaa)	Range(bbbb) ¹
Discrete			
Discrete Inputs	I	1-128	1-128
Discrete Outputs	O	1-128	1-128
Discrete Internals	S	1-1024	1-1024
System Discretes	SD	1-16	1-16
Register Discretes			
Bit Access Registers*	R	1-16384 / 0-15	1-64 / 0-15

¹Jog tag behaves like positive and negative edge triggers and therefore is restricted to first 64 registers when used with bits within word registers

Memory Type	Syntax (C, D)	Range (cccc)	Range (dddd)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: DISCRETES and UNSIGNED_INT_16



In this example, once the Drum is enabled (S1 is ON), it will activate outputs like S3 following the set pattern. The outputs are used to control process like here it turns on a Motor (O1).

3.3.9 Program Control Instructions

Use the Program Control instructions to alter the sequence of Main Logic Program scan.

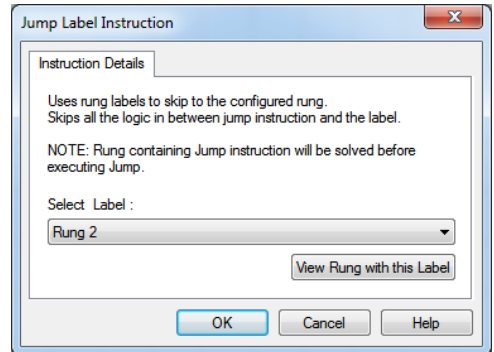
Adding Program Control Instructions

To configure Program Control instructions, perform the following steps:

1. Click the instruction icon on the right side of the screen.
2. Position the mouse over the Ladder Logic and click the mouse to place the instruction.
3. Double click the Jump, For Loop, and Call Subroutine instruction to open the instruction's dialog box. Next and Return do not have dialog boxes.

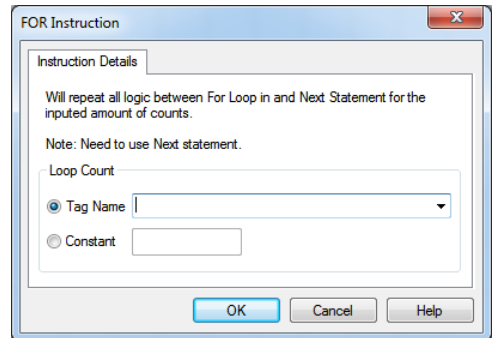
A) Jump Instruction

Select a proper Rung label from the drop down list. Labels need to be added before using this instruction. Please see section 2.5.5 for more information on adding labels to rungs.



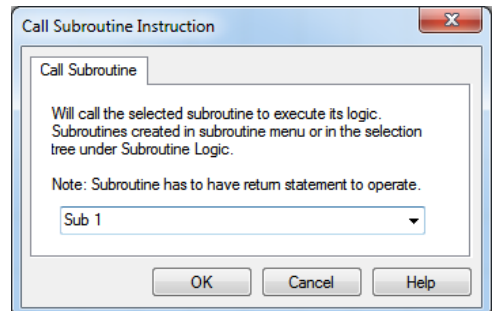
B) For Loop Instruction

Select a proper Tag name/address from the drop down list for the Loop Count or enter a number in the Constant Value field.



C) Call Subroutine Instruction

Select a Subroutine from the drop down list.

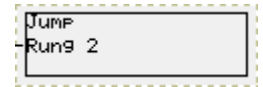
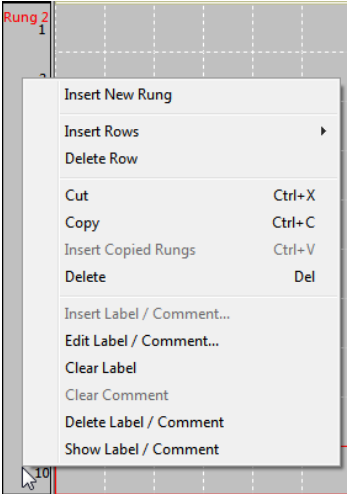




Jump

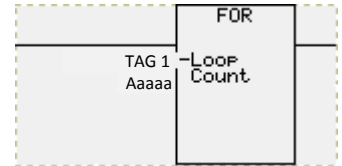
Jump:

When power flows to this element, the Jump instruction skips from the rung where used to a rung with the Label specified in the Jump instruction and continues executing the program thereafter. Before the Jump instruction skips to the specified label instruction, the rung containing the Jump instruction is executed first. The Jump instruction can only be used to skip forward in the direction of the ladder logic flow. When a new rung is created, you can add “label” and “comments” for every rung added by right clicking on the Rung sidebar. Only rungs which are labeled can be utilized by the Jump instruction. The Select Label pull down menu only shows rungs which have been labeled by the user.



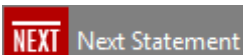
For Loop:

When power flows to this element, the For Loop instruction loops/repeats the ladder logic (RLL) between itself and the Next instruction for the number of times specified by the data value of the Loop Count at memory location Aaaaa. When the For Loop instruction is done executing the RLL between itself and the Next instruction by the number specified by the Loop Count, it allows execution of ladder logic after the Next instruction. The Loop Count can also be assigned a constant value.



Memory Type	Syntax (A)	Range (aaaa)
Registers		
Input Registers	IR	1-64
Output Registers	OR	1-64
Registers Internals	R	1-16384

Allowed Data Formats: *UNSIGNED_INT_16*



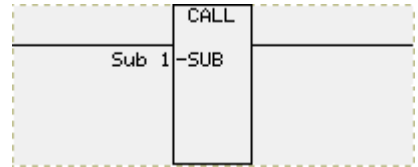
Next Statement:

When power flows to this element, the Next Statement instruction specifies the end point of the For Loop instruction and shifts power flow back to the point where the For Loop instruction is located. Once For Loop execution is completed for the number of times specified by the Loop Count, power will flow through this element.



SUB Call Subroutine**Call Subroutine:**

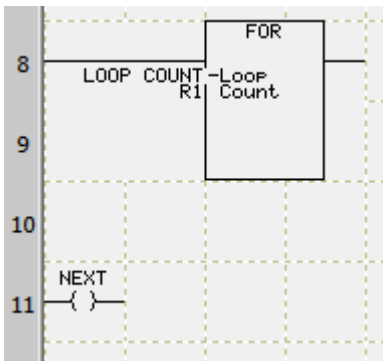
When power flows to this element, the Call Subroutine instruction invokes a subroutine as specified by SUB. You can either specify an existing Subroutine, or create a new one. When a subroutine is added in SUB which already does not exist, it is automatically added under Subroutine Logic. Once a subroutine is used, it must contain a Return instruction to return back to the main logic.



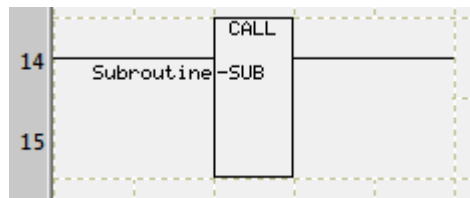
Note: Subroutines are very useful for organizing the main body of ladder logic. They can be utilized to break the body of ladder logic into sections which are either specific to a certain operation or are repeated in the main logic. If certain logic is to be repeated several times, it is useful to place that logic in a subroutine and call that subroutine by using the Call Subroutine instruction instead. By utilizing subroutines efficiently, the number of rungs in ladder logic could be reduced drastically.

RET Return**Return Statement:**

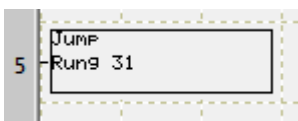
When power flows to this element, the Return Statement instruction specifies the end of the Subroutine logic where present and returns back to the Main logic. The Return statement can only be used in a Subroutine. Any logic after the return statement is not executed so if you want to save no longer need logic you can do so in subroutines.



In this example, any instructions between For and Next will be executed multiple times. The number of times the instructions will be executed is equal to the value of the "Loop Count" variable.



When power flows to this instruction, subroutine named "Subroutine" would be called



In this example, when power flows to this instruction, the execution jumps to "Rung 31".

3.3.10 String Instructions

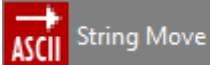
A string is succession of characters. EZRack PLC's string instructions operate on ASCII String Data files only. Please see *Section 3.3.1* for more information on ASCII String Data type.

Adding String Instructions

1. To configure String instructions, click on the String instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Tag Name/Address, double click the instruction to open its dialog box.
4. To configure the String Move and String Compare instructions, perform the following steps:
 - a. Select a Tag name/address from the drop down list for the Source register (Source1 for String Comparison Instruction). Or add a new tag.
 - b. Enter a value in the Number of Characters field.
 - c. Select a Tag name/address from the drop down list for the Destination register (Source 2 for String Comparison Instruction). Or add a new tag.
5. Adding String Length Instruction
 - a. Select a Tag name/address from the drop down list for the 'String' register. Or add a new tag. Needs to be ASCII String tag.
 - b. Select a Tag name/address from the drop down list for the 'Save in' register. Or add a new tag.
6. For adding String Pack and Unpack please see individual sections for those instructions.

Adding tags:

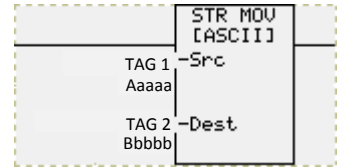
- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



String Move:

When power flows through this element, the String Move instruction moves an ASCII string with a starting address of Src at memory location Aaaaa to Dest at memory location Bbbbb by the number of characters defined by the user.

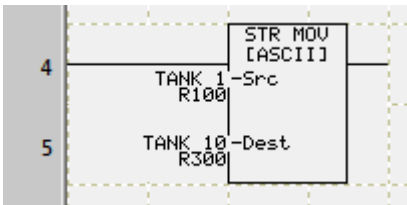
This instruction can move up to 126 characters with every two characters occupying one ASCII register.



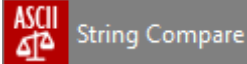
For example, if the number of characters to move is 2, this instruction will move the single Src register at memory location Aaaaa to Dest at memory location Bbbbb. If the number of characters to move is 4, then this instruction will move TWO consecutive registers with a starting address of Src to TWO consecutive registers with starting address of Dest. Similarly, 6 characters would move THREE consecutive registers, 8 characters would move FOUR consecutive registers and so on.

Memory Type	Syntax (A)	Range (aaaa)
Registers		
Registers Internals	R	1-16384

Data Format: ASCII only

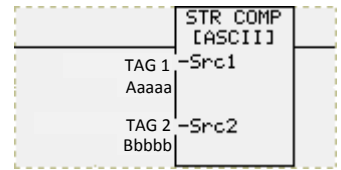


In the example above, the “number of characters” defined in the dialog box are moved starting from R100 to the destination starting from R300. If a null is found in the source string before all the “number of characters” are moved, the rest of the characters are padded with null in the destination.



String Compare:

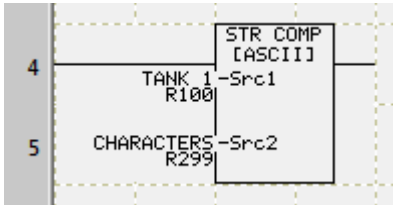
The String Compare instruction is used to compare an ASCII String with a starting address of Src1 at memory location Aaaaa and Src2 at memory location Bbbbb by the number of characters specified by the user. If Src1 = Src2 power will flow through this element. This instruction can compare up to 126 characters with every two characters occupying one ASCII register.



For example, if the number of characters to compare is 2, this instruction will compare the single Src1 register at memory location Aaaaa to Src2 at the memory location Bbbbb. If the number of characters to compare is 4, then this instruction will compare TWO consecutive registers with a starting address of Src to TWO consecutive registers with a starting address of Src2. Similarly, 6 characters would compare THREE consecutive registers, 8 characters would compare FOUR consecutive registers and so on.

Memory Type	Syntax (A)	Range (aaaa)
Registers		
Registers Internals	R	1-16384

Data Format: ASCII only

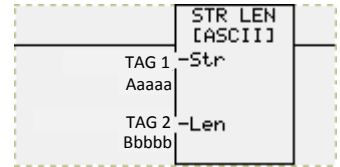


In the example above, the string starting from R100 is compared with the string at R299. The strings are compared up to the “number of characters”, or 126, or up to a null character in either of the sources, whichever occurs first.

ASCII String Length

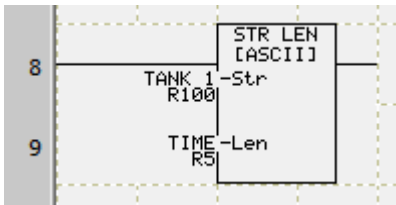
String Length:

When power flows through this element, the String Length instruction counts the number of characters in a null-terminated ASCII string specified by the starting address of Str at memory location Aaaaa. The result is stored in Len at memory location Bbbbb.



Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384

Allowed Data Formats: UNSIGNED_INT_16, UNSIGNED_INT_32, and ASCII_STRING.

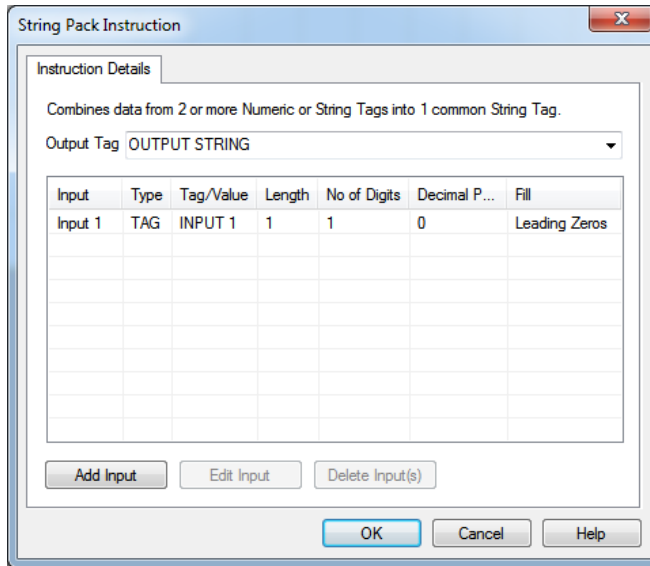


In the example above, when power flows to the instruction, the length of the string starting at R100 is computed. The computation stops when a null character is found. The length value is saved in R5.

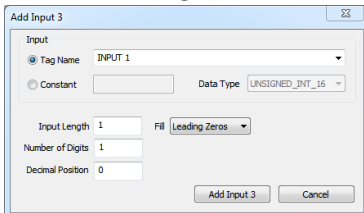
Adding the Pack Instruction:

To configure the Pack instruction, perform the following steps:

1. Click on the Pack icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. Double click the instruction to open its dialog box.
4. Select or Add Output Tag. *Note: The total length of Output has to be equal or greater than the total combined lengths of the inputs to be packed.*



5. Add up to 16 Inputs you would like to combine in order of combination by clicking Add Input. (Input 1 will be first in the output, Input 2 is second, etc.)
6. In the dialog Select or Add Input.

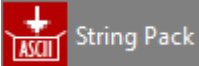


a. For ASCII String Inputs specify the Input length.

b. For other Register types specify both the length, the number of digits, Fill type and the decimal position.

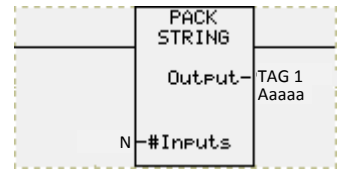
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



String Pack:

When power flows through this element, the String Pack takes any register type and combines them into an ASCII string. The resulting ASCII string is stored at memory location Aaaaa. The instruction takes up to 16 different inputs from multiple different memory locations. Any input can also be a constant.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

For each input the string length to pack needs to be specified. If an input has a floating point data type it can either be rounded off to the nearest integer value or truncated. When an input of integer or floating point data value is converted, the number of digits, decimal position and justification (leading zeros, leading spaces, or trailing spaces) must be assigned. Power will stop flowing through this element if an error occurs such as overflow, underflow, or divide by zero.

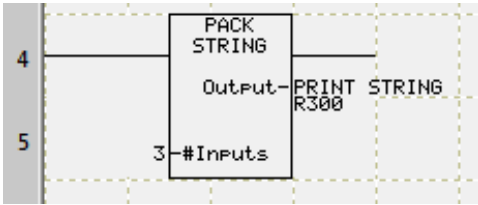
The String Pack instruction will only combine the inputted length specified for each input element even if the input's size is greater e.g. if length specified is 10 but ASCII string is 12 long will only pack the first 10 characters of the ASCII String.

Note: For signed registers and registers with decimals please note that the length to pack needs to be one greater than the number of digits due to the need for a +/- sign and or decimal point. If number has both then the length to pack needs to be 2 greater than the number of digits.

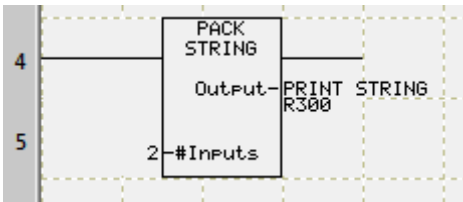
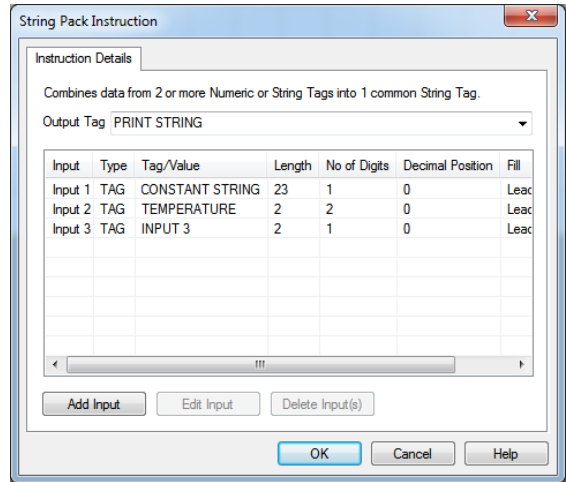
Note: For Floating point registers be mindful of the scientific notation. Based on the numeric value of the register the number can be displayed in decimal notation (##.##) or scientific notation (###e+###). For scientific notation the length to pack needs to be four greater than the number of digits.

Memory Type	Syntax (A, Inputs)	Range (aaaa)	Inputs
Registers			
Output Registers	OR		1-64
Registers Internals	R	1-16384	1-16384

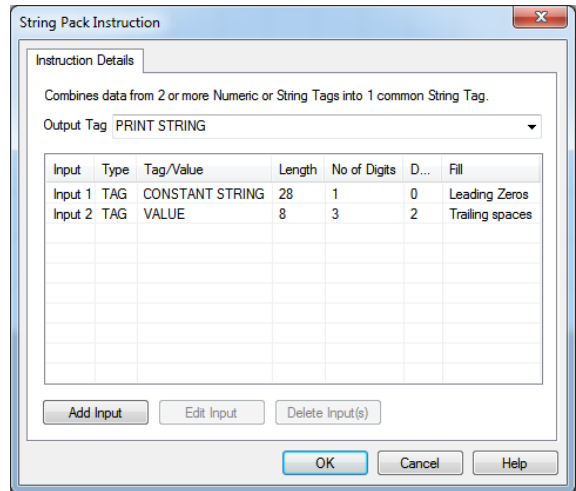
Allowed Data Formats: all register data types.



In this example, string R100, register R20, and string R120 are packed together into R300 when power flows. For this example it would create string "Current Temperature is 20°C". With string R100 being 23 Char long ("Current Temperature is "). The register R20 would be the temperature with length 2 and Number of digits 2 as well ("20"). Finally string R120 would be 2 char long ("°C").



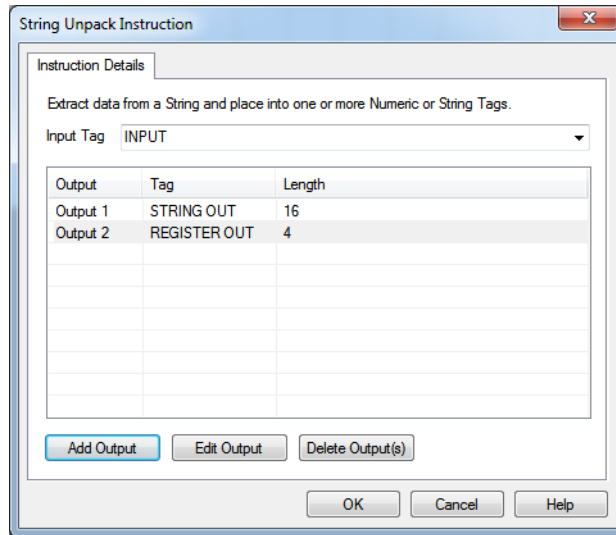
In this example, string R100, and register R20 are packed together into R300 when power flows. For this example it would create string "Current calculated value is 2.45e+03". With string R100 being 28 Char long ("Current calculated value is "). The register R20 would be the value with length 8 ("2.45e+03"), Number of digits 3 ("2.45"), and Decimal Position is 2 (".45").



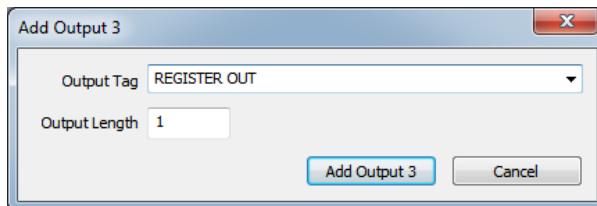
Adding the Unpack Instruction:

To configure the Unpack instruction, perform the following steps:

1. Click on the Unpack icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. Double click the instruction to open its dialog box.
4. Select or Add Input Tag. *Note: Input Length has to be equal or greater than the total combined length of the outputs.*



5. Using the Add Output button add up to 16 Outputs you would like to unpack the input into. These outputs can be of any type and only the length needs to be specified.
6. In the Add Output dialog select or add an output tag. Then specify the length to be outputted to the tag.



Adding tags:

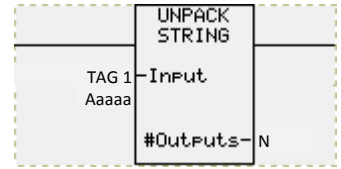
- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



String Unpack

String Unpack:

When power flows through this element, the String Unpack takes an ASCII String and splits it into different registers of any type. The Input ASCII string is stored at memory location Aaaaa. The instruction outputs to up to 16 different outputs from multiple different memory locations. For each output the string length to unpack needs to be specified. Power will stop flowing through this element if an error occurs such as overflow, underflow, or divide by zero.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

The instruction will split the ASCII string starting from the beginning into the different registers per the specified length. Therefore if the length of the Input String is shorter than the total length of the outputs, only the outputs up to the Input String length will be split into.

Note: An ASCII register takes 2 char per register address, e.g. for 10 char need R1-R5. Only the first address needs to be specified, the others are automatically used.

Note: An ASCII character cannot be split into a non-ASCII register. Therefore if such is attempted the output register will be set to 0, e.g. if try to input AB into UNSIGNED_INT_16 this will result in the register being 0.

Note: For Floating point registers be mindful of the scientific notation. Floating point values can be displayed in decimal notation (##.##) or scientific notation (###e+###). For scientific notation the length to unpack needs to be four greater than the number of digits.

Memory Type	Syntax (A, Outputs)	Range (aaaa)	Outputs
Registers			
Output Registers	OR		1-64
Registers Internals	R	1-16384	1-16384

Allowed Data Formats: all register data types.

Examples of how it will unpack different length strings:

Example 1:

Input: ABCDEF

If splitting into Output 1 (Length 10) and Output 2 (Length 10).

Then only Output 1 will be used since Input length < Output 1 length.

Output 1: ABDCEF ____

Output 2: _____

Example 2:

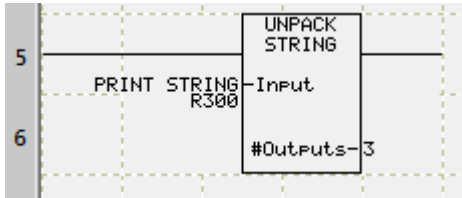
Input: ABCDEFGHIJK

If splitting into Output 1 (Length 10) and Output 2 (Length 10).

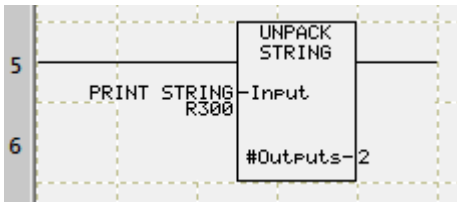
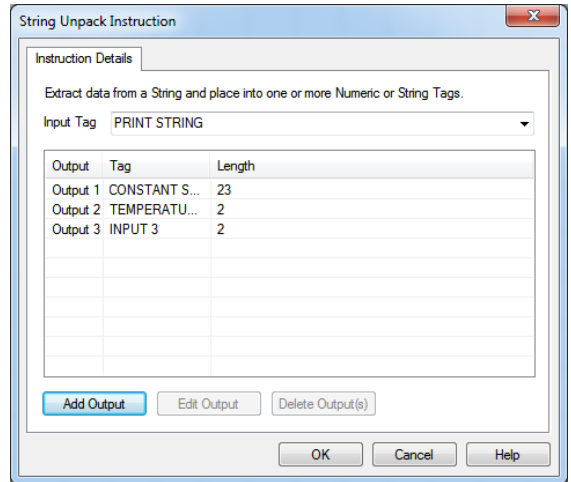
Then both Output 1 and Output 2 will be used but since Input length < (Output 1 + Output 2) length, then only some of Output 2 will be used.

Output 1: ABCDEFGHIJ

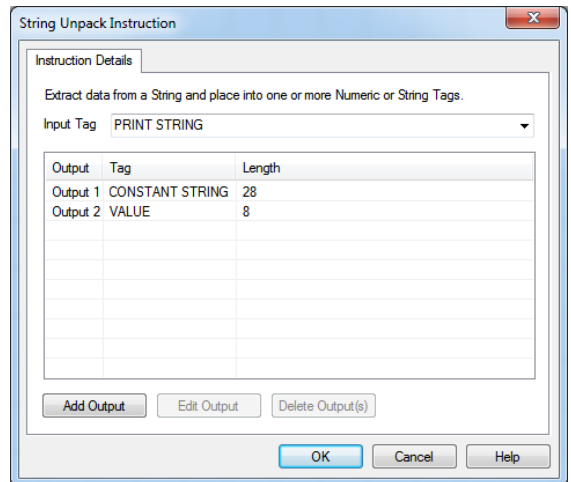
Output 2: K _____



In this example, R300 is unpacked into string R100, register R20, and string R120 when power flows. For this example it would take apart string "Current Temperature is 20°C". With string R100 being 23 Char long ("Current Temperature is "). The register R20 would be the temperature with length 2 ("20"). Finally string R120 would be 2 char long ("°C").



In this example, R300 is unpacked into string R100 and register R20 when power flows. For this example it would take apart string "Current calculated value is 2.45e+03". With string R100 being 28 Char long ("Current calculated value is "). The register R20 would be the value with length 8 ("2.45e+03").



3.3.11 Communication Instructions

Use Communication instructions to open and close the serial port for sending ASCII data to communicate with external devices.

Adding Communication Instructions

To configure String instructions, perform the following steps:

1. Click on the Communication instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. Double click the instruction to open its dialog box.

Adding Open Port Instructions

To configure the Open Port instruction, perform the following steps:

Open Port Instruction

Instruction Details

Opens the serial port of the PLC for third party communication. As soon as the port is open for third party communication, program loader cannot communicate to PLC through this serial port.

Use with Sent To Serial Port, Receive From Serial Port, Send to Marquee, and/or Modbus Master.
Note: Only 1 open port command needed.

NOTE:

1. Choose the format for the character sequence (HEX or ASCII). MUST Select HEX format if NULL (value 00) is one of the characters.
2. Enter maximum of 4 characters separated by comma (no spaces). Example: a,b,c,d
3. Each HEX character can be specified with 1 or 2 HEX digits. Example: 1,2,03,4

Baud Rate: 9600
Parity: None
Data Bits: Eight
Stop Bits: One
Select Mode: RS422
Protocol: No Protocol

Enter/Display Char Sequence
 Hex ASCII

Send Character Sequence
Start Characters:
End Characters:

Make Receive Sequence Same as Start
Receive Start Characters:
Receive End Characters:

OK Cancel Help

1. Select a Baud Rate using the drop down list.
2. Select a Parity value (None, Odd, or Even) using the drop down list.
3. Select Data Bits (7 or 8) using the drop down list.
4. Select Stop Bits (1 or 2) using the drop down list.
5. Select Mode (RS232, RS422, or RS485) using the drop down list.

6. Select a Protocol (None, Xon / Xoff, Modbus Master, Modbus Slave) using the drop down list.

Enter Optional Parameters:

1. Select how the Char Sequence is inputted (Hex or ASCII).
2. Enter Send Start Characters in the Start Characters field (up to 4 characters).
3. Enter Send End Characters in the End Characters field (up to 4 characters).
4. Enter Receive Start Characters in the Start Characters field (up to 4 characters).
5. Enter Receive End Characters in the End Characters field (up to 4 characters).

Adding Send To and Receive From Port Instructions

To add the Send to Port and Receive From Port instructions, perform the following steps:

1. Select or Add an ASCII tag that contains the string to be sent in the Source Tag field using the drop down list (for a Receive instruction: the String that will receive the characters from the serial port in the Destination Tag field).

2. Select an integer register used by the instruction for status in the Control Register Tag field using the drop down list. The following table describes the control bits in the register:

Bit Number	Function
Bit 0 (lsb)	Enable (0 = Disabled, 1 = Port is Open AND Instruction is Enabled (Power flows to instruction))
Bit 1	Serial transmission done (1= function (transmit or receive) done, 0=not done)

Other bits of the register are used for internal purposes and change state during transmission/receiving.

3. Select or Add an integer register that displays the number of characters transferred from the source tag to the serial output buffer in the Character Count Tag field using the drop down list (for a Receive instruction: the Number of characters transferred from the serial port to the destination tag).
4. Check either Send Start Character or Send End Character box if needed.

Adding Send to Marquee Instruction

To add the Send to Marquee instruction, perform the following steps:

1. Select or Add a Source Tag name/Address using the drop down list.
2. Check the Use Mask box and enter a value, if you want to use Mask capabilities to compute message number.
3. Enter a numeric constant as an Offset value to the message number if desired.
4. Select or Add a Message Status Tag name/Address using the drop down list.

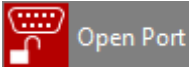
5. Check one option for the action for Unmatched message numbers.
6. Add/Edit the Message database by clicking on the View/Edit Message Database button.

Adding Modbus Master Instruction

Please go to *Chapter 7*.

Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Open Port:

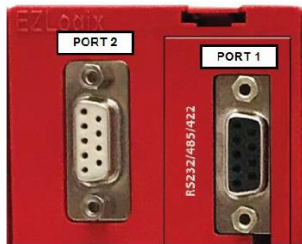
When power flows through this element, the Open Port instruction opens **only** Port 1 serial port using user specified parameters available as follows:

- Baud Rate: 1200, 2400, 4800, 9600, 19200, 38400
- Parity: None, Odd, Even
- Data Bits: Seven, Eight
- Stop Bits: One, Two
- Mode: RS232, RS422, RS485
- Protocol: No Protocol, XOn /XOff, Modbus Master, Modbus Slave

Open Port	
9600	-BaudRate
No	-Parity
Eight	-DataBits
One	-StopBits
RS422	-Mode
No Protocol	-Protocol

Note: When the serial Port 1 is being used for 3rd party communication, it cannot be used to communicate with the EZRack PLC Designer Pro. Please therefore use other communication options like Ethernet or Micro-USB.

Serial Ports:

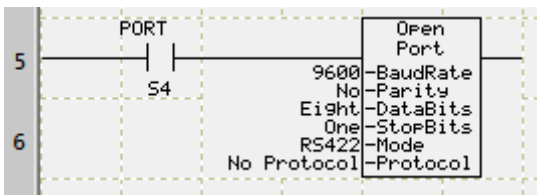


PIN CONFIGURATION	
Pin Number	Function
1	SD -
2	TXD
3	RXD
4	RD -
5	GND
6	SD +
7	CTS
8	RTS
9	RD +

Send Character Sequence can be used to add up to a maximum of FOUR characters in the beginning and/or ending of every command that is sent out using this port. The 4 characters must be separated by a comma.

Receive Character Sequence can also be used to verify a maximum of FOUR characters in the beginning and/or ending of every command that is received using this port. The 4 characters must be separated by a comma. You can also specify to make the Receive Character Sequence the same as the Send Character Sequence.

If HEX values are used for the two sequences, two characters must be used to specify 1 HEX value.



In this example, if S4 is on, the port will be opened with the parameters shown in the instruction. Please note that the Port command is executed **ONLY** once every time S4 changes state from 0 to 1.

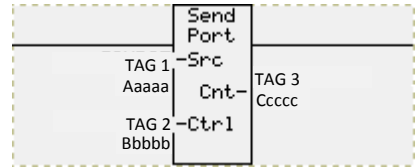
NOTE: The Open Port instruction is executed once every time the power flows to the instruction. It is recommended that the port be opened once, unless the com parameters have to be changed. In that case, the port should first be closed, and then reopened with different parameters.



Send To Serial Port

Send to Serial Port:

When power flows through this element, the Send to Serial Port instruction will send an ASCII string present in Src at memory location Aaaaa to the RS422 port. The control and character count used for sending the ASCII string is specified by Cnt at memory location Ccccc and Ctrl at memory location Bbbbb, respectively.



This instruction can only send out the specified ASCII string if the corresponding serial port has been already opened by the Open Port instruction in advance. If the serial port has not been initiated, the Send to Serial Port instruction will not send the ASCII string to the specified port.

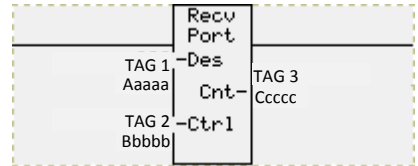
Start and End characters can also be sent along with the ASCII string being sent out from the Src register. You can specify Start and/or End characters to be included along with the ASCII string. The starting and ending characters are specified in the Open Serial Port Instruction.



Receive From Serial Port

Receive From Serial Port:

When power flows through this instruction, the Receive From Serial Port instruction will receive an ASCII string from the serial port and store it in Dest at memory location Aaaaa. The control and character count used for receiving the ASCII string is specified by Cnt at memory location Ccccc and Ctrl at memory location Bbbbb, respectively.



This instruction can only receive the specified ASCII string if the corresponding serial port has been already opened by the Open Port instruction in advance. If serial port has not been initiated, the Receive from Serial Port instruction will not receive the ASCII string.

Start and End characters can also be received along with the ASCII string being received. You can specify Start and or End characters to be verified when received along with the ASCII string. The starting and ending characters are specified in the Open Serial Port Instruction.



Close Port

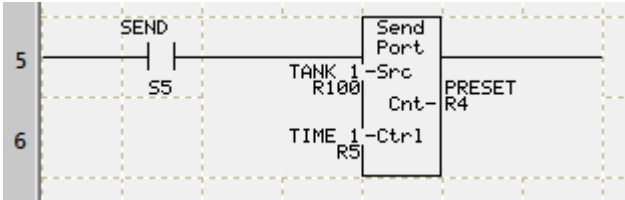
Close Port:

When power flows through this element, the Close Port instruction closes the serial port previously opened for communication by the Open Port instruction. Once the port is closed, it cannot be used unless it is re-opened by the Open Port instruction.

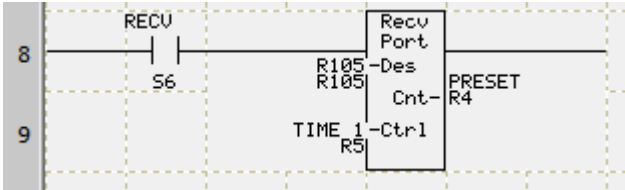
Close Port

Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

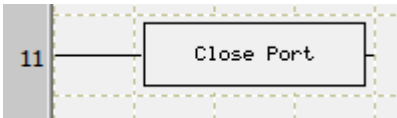
Allowed Data Formats: ASCII and UNSIGNED_INT_16.



In the example above, if S5 is ON (and the Port is Open), the Send Port command would send the ASCII string as per programmed parameters. If the port is not yet open, the instruction will do nothing, and the Enable Bit in the control register will remain 0, even if the S5 is on.



In the example above, if S6 is ON (and the Port is Open), the Send Port command would receive the ASCII string as per programmed parameters. If the port is not yet open, the instruction will do nothing, and the Enable Bit in the control register will remain 0, even if the S6 is on.



In the example above, if power flows to the close port instructions, the opened serial port is closed. Once the port is closed, it will not send any serial communication for any command (such as Send to Port, Send to Marquee) without reopening the port.

Message Status Tag:

Ctrl is used by the Send to Marquee instruction to specify the status of the message being sent to a marquee. If the message is being transmitted to the serial port, the bit 0 (lsb) of Ctrl is enabled (1). When the message is successfully sent to the serial port, the bit 1 of Ctrl is enabled (1).

Message Database:

When the Send to Marquee instruction is assigned a message number through Src, the message corresponding to the message number is selected for transmitting to the serial port. The Message database is populated by using “View/Edit Message Database” tab. When adding a new message the text can be assigned a message number and attributes such as blinking, scrolling, and centering of messages etc. The very first message in the Message Database is the default message. This message is sent to the specified marquee(s) (broadcast or a certain unit) when the message number assigned by Src does not have the matching message in this database.

Note: Only messages with the correct message number as per Src register will be displayed.

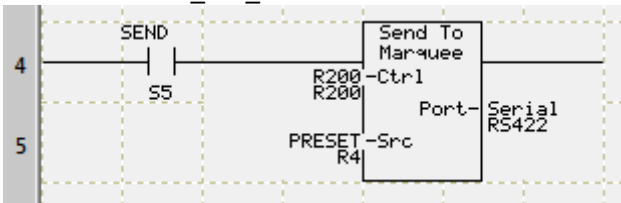
Actions for Unmatched Messages:

If the Src register points to a message number that does not exist in the Message Database, then there are three options, one of which can be selected by the user for appropriate action.

- Send Default Message: Sends a “Default Message” as specified in the Message Database
- Send Blank Message: Sends out a blank message with no text; clears the display line of the marquee(s) specified in the Default Message
- Do Nothing: No action is taken if the correct message is not found

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: ASCII, UNSIGNED_INT_16, UNSIGNED_INT_32, SIGNED_INT_32, and BCD_INT_16.



In the example above, if S5 is On (AND the Port is open and not busy), the message is sent to Marquee. The message is sent ONLY once every time S5 changes state. So to refresh the message on the Marquee (for example, if an embedded

variable changed), ensure that S5 changes state. Another example for the send marquee is shown on next page. In this example, the logic is monitoring 3 temperature ranges, and displays one of the 3 messages based on the temperature value.

Modbus Master:

Please see *Chapter 7* for more information on EZRack PLC working as Modbus Master or Modbus Slave.

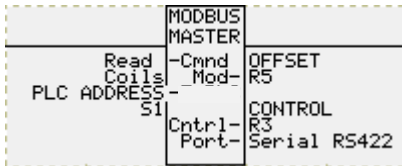
Basic Modbus overview:

EZRack PLC provides connectivity to other devices over Modbus RTU and Modbus TCP/IP protocol. You can use EZRack PLC either as a Modbus Master/Client or a Modbus Slave/Server.

When used as a Modbus Master/client, EZRack PLC communicates and exchanges data with other Modbus slaves/servers. When used as a Modbus Slave, the EZRack PLC can respond to Modbus commands from a Master. The serial port on EZRack PLC is used for the Modbus RTU connection. The Ethernet Port is used for Modbus TCP/IP connection.

Modbus Master Instruction basics:

Select Modbus Master Instruction from the Menu item **Instruction> Communication>Modbus** or from the Instruction side bar. The instruction on ladder logic appears as follows:



Please note the following about the Modbus Master instruction:

- The instruction is initiated when the rung is true (i.e. all instructions in the rung preceding the Modbus instruction are true)
- The instruction involves sending a command to the addressed slave, and processing the reply back from the slave, which is asynchronous to the ladder scan. The Power flows out of the instruction only after the instruction is completed, i.e. after either the reply is received or the instruction times out. Control Word can be used to see the progress of the instruction.
- If the rung condition becomes false before the completion of the instruction, the instruction is not completed. The sending of the command is completed but the reply is not processed, even if received. If the command was a write command, the values MAY be written, but cannot be guaranteed.
- It is advisable to use check the error code register for any potential errors after the instruction is completed.

Double click to bring-up the dialog box:

The image shows a screenshot of the Modbus Master Dialog box with several red callout boxes pointing to specific fields and controls. The dialog box is titled 'Modbus Master' and contains the following fields and controls:

- Communications:** A dropdown menu set to 'Serial RS422 (Modbus RTU)'. A callout box points to it with the text: 'Select whether using Modbus RTU or Modbus TCP/IP'.
- IP:** A text field containing '0 . 0 . 0 . 0'. A callout box points to it with the text: 'For Modbus TCP/IP enter the Modbus Slaves IP Address'.
- Slave ID:** A section with two radio buttons: 'Tag Name' (unselected) and 'Constant' (selected). Below 'Constant' is a text field containing '1'. A callout box points to it with the text: 'For Modbus RTU enter the Modbus Slaves ID'.
- Modbus Command:** A dropdown menu set to 'Read Input Registers (04)'. A callout box points to it with the text: 'Select the Modbus operation this instruction will do. The options including Read or Write, Coils or Registers, One or Many.'
- Byte Order:** Two radio buttons: 'Low Byte, High Byte' (unselected) and 'High Byte, Low Byte' (selected). A callout box points to it with the text: 'For Register Communication select Byte Order'.
- Offset:** A text field containing '5'. A callout box points to it with the text: 'Use the offset to select the address you will be communicating to. If you use offset 5 then the address that will be read is 300005.'
- Data Length:** A section with two radio buttons: 'Tag Name' (unselected) and 'Count' (selected). Below 'Count' is a text field containing '1'. A callout box points to it with the text: 'Enter how many consecutive registers or coils written or read from the Slave Device.'
- PLC ADDRESS:** A dropdown menu set to 'PLC ADDRESS'. A callout box points to it with the text: 'PLC Address is the starting location in the EZRack PLC where written or read information is stored.'
- Control:** A dropdown menu set to 'CONTROL'. A callout box points to it with the text: 'Control and Error are the EZRack registers with the Modbus Master Instruction status information.'
- Error:** A dropdown menu set to 'ERROR'. A callout box points to it with the text: 'Control and Error are the EZRack registers with the Modbus Master Instruction status information.'
- Timeout Time:** A text field containing '30'. A callout box points to it with the text: 'Control and Error are the EZRack registers with the Modbus Master Instruction status information.'

An 'OK' button is located at the bottom right of the dialog box.

The following attributes need to be set in the Modbus Master Dialog box.

1. Communications

Use this dropdown to select whether RTU or Modbus TCP/IP is used. If TCP/IP is used a field will appear to input the IP Address of the Slave.

2. Slave ID

The Network ID number of the Slave Device we are communicating to. This may either be stored in a Tag or defined as a constant. Not used for TCP/IP.

3. Modbus Command and Modbus Address

Select Modbus command and address to communicate to. You don't need to enter the command codes. In addition the Modbus address type is not entered; only the offset within the address type is entered. For example for holding register 400123, use only 123. The address type is implied by the command.

4. Byte Order

Modbus registers are usually arranged as MSB-LSB. This flag allows you to change the order if necessary.

5. Data Length

Number of Data Items to process. The data length may either be stored in a Tag or defined as a constant.

6. EZRack PLC Address

Please enter the Starting EZRack PLC Address where information will be stored or read from.

7. Control

Enter the EZRack PLC address that will store the state of the execution of the Modbus Master instruction. Bit 0 (LSB) to Bit 4 of the Control address are used to indicate the status of the Modbus instruction as follows:

Bit Number	Status when set
B0 (LSB)	Modbus serial Enable
B1	Waiting on reply
B2	Reply processed
B3	Not used
B4	Invalid length for starting address

8. Error

Enter the EZRack PLC address that will store the Error codes if there is any error in execution of the instruction. A zero value (0) indicates no error has occurred. The error code must be checked only after the instruction is completed (i.e. the power flows out of instruction). See below for error codes and their descriptions.

ERROR CODE	Error	Description
01	Illegal Function	The function code (command code) in the Modbus Master command is not understood by the Slave.
02	Illegal Data Address	The Modbus Master command tried to access an address not available in the Modbus slave device.
03	Illegal Data Value	The Modbus Master Instruction sent a value not acceptable to the slave.
04	Slave Device Failure	An error occurred in slave device, while the slave was trying to perform action requested by Modbus Master.

05	Timeout	A reply was never received from the slave (the communication link Between the Master and the Slave may be disconnected.)
07	Checksum Error	Error in check sum of the reply
08	Slave ID Failure	The slave id in the master command message does not match the slave id Returned in the reply message from the Slave.
09	Port not open error	The Port on EZRack PLC is not opened for Modbus Master Instruction

9. Timeout

Enter the timeout period in tenth of seconds. EZRack PLC Modbus Instruction will time out if a slave does not respond to a command within this time.

3.3.12 Data Logging Instructions

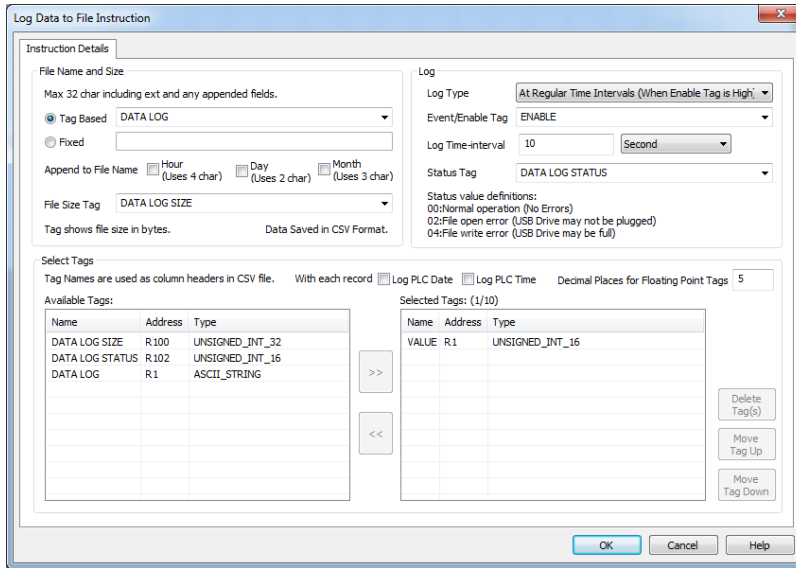
Use Data Logging instructions to log tag data to an USB.

Adding the Log Data to File Instruction:

To configure the Log Data to File instruction, perform the following steps:

1. Click on the Log Data to File icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.

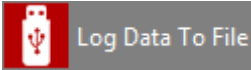
3. Double click the instruction to open its dialog box.



4. Select or Add a tag for Data Log Name. This can also be constant.
5. Select whether to append hour, day, and month to Data Log Name. This means that every new hour, day and month a new .csv file is created.
6. Select of Add a tag for File Size, Event/Enable and Status.
7. Select the Log Type from the drop down. Descriptions are on next page.
8. Enter Log Time Interval for time based data logging and select time base using the drop down options.
9. Finally select up to 10 tags to data log. Options include data logging the date and time in the .csv file as well.

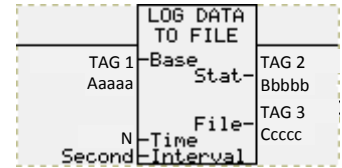
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Log Data to File

When power flows through this instruction, the Data Log instruction will create and update a csv file on an inserted USB thumb drive (up to 64GB). The instruction will log the current value of up to 10 tags in this csv. You can use up to a maximum of 8 of these instructions.



Log Data to File Instruction

Instruction Details

File Name and Size
 Max 32 char including ext and any appended fields.
 Tag Based **File Name (Aaaaa)**
 Fixed
 Append to File Name Hour (Uses 4 char) Day (Uses 2 char) Month (Uses 3 char)
 File Size Tag **Data Log Size (Ccccc)**
 Tag shows file size in bytes. Data Saved in CSV Format.

Log
 Log Type **At Regular Time Intervals (When Enable Tag is High)**
 Event/Enable Tag **Event/Enable Tag (Ddddd)**
 Log Time-interval **10** **Second**
 Status Tag **Data Log Status (Bbbbb)**
 Status value definitions:
 00:Normal operation (No Errors)
 02:File open error (USB Drive may not be plugged)
 04:File write error (USB Drive may be full)

Select Tags
 Tag Names are used as column headers in CSV file. With each record Log PLC Date Log PLC Time Decimal Places for Floating Point Tags **5**

Available Tags:

Name	Address	Type
Value 2	Ffff	UNSIGNED_INT_32
Value 3	Ggggg	UNSIGNED_INT_16

>> <<

Selected Tags: (1/10)

Name	Address	Type
Value	Eeeee	UNSIGNED_INT_16

Delete Tag(s) Move Tag Up Move Tag Down

OK Cancel Help

File Name:

This option allows the user to set the File Name that the .csv is saved under in the USB. The user can either use an ASCII tag stored at memory location Aaaaa, or they can make this constant. Further options are to append the hour, day and month that this data log happens. Appending these creates new .csv files each time the hour, day and month are different.

Data Log Size:

This is an unsigned 32 bit word stored at memory location Ccccc which will tell you the file size of the .csv on the USB.

Data Log Type:

There are four types of selectable options for data type:

- **On Rising Edge of Event Tag**
This options allows for **Event Based** only data logging. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only data logs if this tag goes from OFF (0) to ON (1).
- **On Falling Edge of Event Tag**
This options allows for **Event Based** only data logging. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only data logs if this tag goes from ON (1) to OFF (0).
- **On Both Edges of Event Tag**
This options allows for **Event Based** only data logging. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only data logs if this tag goes from ON (1) to OFF (0) or if it goes OFF (1) to ON (1). Combined other two options.
- **At Regular Time Intervals (When Enable Tag is High)**
This options allows for **Time Based** only data logging and **Event/Time Based** combined data logging. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and data logs at the specified Log Time-interval if this tag is ON.
 - **Time Based:** To do Time Based only data logging just either turn ON (1) the Event/Enable tag or set the Event/Enable tag to have an initial value of 1 (ON).
 - **Event/Time Based:** To do Event/Time Based only data logging just use the Event/Enable tag as the event control. As soon as the Event/Enable tag is ON (1) the instruction will data log at specified Log Time-interval until this tag is turned OFF (0).

Log Time-interval

The Data Log instruction will log based on the Log Time-interval if the Log Type is **Time Based** or **Event/Time Based**. Use the time base dropdown to set the units and enter the amount of time between each data log.

Time Base:

The Time Base is user selectable and allows one of the following time bases:

- Second
- Minute
- Hour

e.g. If Log Time Interval = 15 and Time Base = Second, then the instruction will data log every 15 Seconds. Similarly, if Log Time Interval =11 and Time Base = Hour, then the instruction will data log every 11 hours.

Data Log Status:

The data log will display current status of the instruction in this tag stored at memory location Bbbbb. More than 1 status can be true at a time so if status is 66 then it is status 2 and 64.

Please see below for the available statuses:

Status Value	Description	Explanation
00	Currently Data Logging (Normal Operation)	This means the data log instruction is currently writing to the .csv file.
02	File Open Error	Cannot open .csv file. USB drive may not be plugged in.
04	File Write Error	Cannot write to .csv file. USB driver might be full. Or the .csv file may be read only.
64	Done Data Logging	Have finished writing to the .csv file. Happens if the instruction no longer has power or when it is data logging only at set time intervals.

Note: System Discrete SD25 will indicate the status of the USB port. It will be ON (1) if there is an USB in the port, it will be OFF (0) if there is no USB.

Selected Tags for Data Logging:

The available tags field will have all the currently created discretives and registers for the project. To add a tag to be data, just select the tag and press the >> button to move it over to the selected tags. Tag from any location can be data logged. Tags are data logged in columns which are in the order of top to bottom (left to right). The .csv file can also have a column for PLC Data and Time if it is selected. If data logging floating point tags the Decimal Places for Floating Point field will give how many decimals are logged (if 5 selected then number will be #.#####).

Memory Type	Syntax (D)	Range (dddd)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretives	SD	1-16
Register Discretives		
Bit Access Registers*	R	1-16384 / 0-15
<small>*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.</small>		

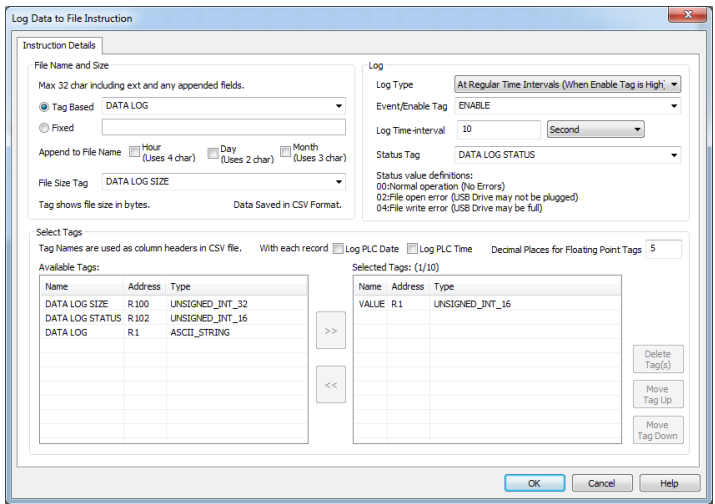
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Note: Memory table for E, F, and G pertains to all 10 tags that can be added to Data Log Instruction.

Memory Type	Syntax (E, F, G)	Range (eeee)	Range (ffff)	Range (gggg)
Discrete				
Discrete Inputs	I	1-128	1-128	1-128
Discrete Outputs	O	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024
System Discretes	SD	1-16	1-16	1-16
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

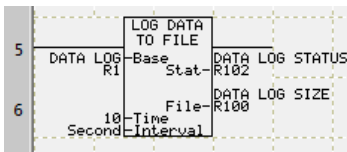
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete and all Register data types except BCD.



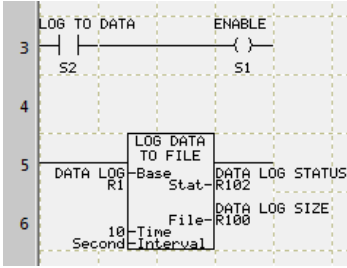
All examples use the above settings except the Log Type is changed.

Example Time Based:



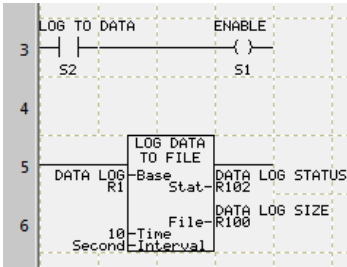
In this example, Log Type is **At Regular Time Intervals (When Enable Tag is High)**. The data log will Log at 10 second intervals if the Enable tag (S1) is ON (1).

Example Event Based:



In this example, Log Type is **On Both Edges of Event Tag**. The data log will Log when the state of the Enable tag (S1) is changed (from ON to OFF or OFF to ON).

Example Both Event and Time Based:



In this example, Log Type is **At Regular Time Intervals (When Enable Tag is High)**. The data log will Log at 10 second intervals if the Enable tag (S1) is ON (1). The S2 bit will therefore control the data log instructions.

3.3.13 Datatype Conversion

The instructions listed within this chapter perform datatype conversion operations on user specified values or addresses.

Adding Datatype Conversion Instructions

To configure all of the various Datatype Conversion instructions, perform the following steps:

1. Click on any Datatype Conversion instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.
4. Select or add Source and Destination Tag names/addresses.
5. For Format Conversion select the type of conversion.
6. For X=Y Conversion select if applicable the rounding for floating point and also the ASCII details.

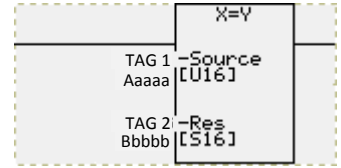
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

X=Y X=Y Conversion

X=Y Conversion:

When power flows to this element, the X=Y Conversion instruction converts the register data type of Src at memory location Aaaaa to Res at memory location Bbbbb and copies the converted data value to Res at memory location Bbbbb. If Src has a Floating Point data type it can either be rounded off to the nearest integer value or truncated when converting to other data types. When the integer or floating point data value is converted to an ASCII type data value, the number of digits, decimal position and justification (leading zeros, leading spaces, or trailing spaces) can be assigned as per user.

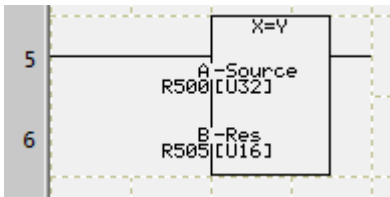


Note: This is the same instruction that can also be found in the Math Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data types.

Note: If converting a signed 16 bit (with a negative value -1) to an unsigned 16 bit register the result will always be zero.



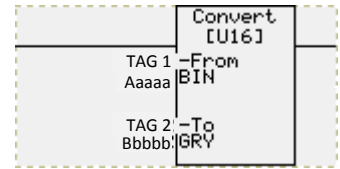
In the example above, variable A, (R500) which is an UNSIGNED_32 (U32) Type, will be converted to an UNSIGNED_16 Type (U16) and saved in B.



Format Conversion

Format Conversion:

When power flows to this element, the Format Conversion instruction converts the data format of From at memory location Aaaaa to To at memory location Bbbbb as follows:



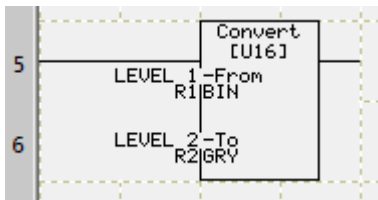
- Binary to BCD
- BCD to Binary
- Binary to Gray Code
- Gray Code to Binary

Both the From and To data types must be a 16 bit Signed Integer, 16 bit Unsigned Integer, or 16 bit BCD for Format Conversion instruction.

Note: This is the same instruction that can also be found in the Math Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: SIGNED_INT_16, UNSIGNED_INT_16, BCD_INT_16



In the example above, R1 which is in Binary format, is converted to Gray Code and saved in R2.

3.3.14 Process Alarms/Faults

The instructions listed within this chapter include all the Alarm/Fault operations that can be used in the Ladder Logic.

Adding Alarm Instruction

To configure an Alarm Instruction, perform the following steps:

1. Click on the Alarm instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

Instruction Details

Evaluates the current condition of an input, whether it is above or below certain setpoints (2 setpoints above and 2 below).

Input

Input Tag INPUT TAG

Data Type UNSIGNED_INT_16 High High, High, Low and Low Low must be of same data type.

Limits / Set Points

High High

Tag Name HIGH HIGH

Constant (In Decimal)

High

Tag Name HIGH

Constant (In Decimal)

Low

Tag Name LOW

Constant (In Decimal)

Low Low

Tag Name LOW LOW

Constant (In Decimal)

Alarms / Results

High High Alarm Tag (Set when input > HH)

ALARM HIGH HIGH

High Alarm Tag (Set when input > H)

ALARM HIGH

Low Alarm Tag (Set when input < L)

ALARM LOW

Low Low Alarm Tag (Set when input < LL)

ALARM LOW LOW

Display all values in Decimal

OK Cancel Help

4. Select or Add Registers for the Input, High High, High, Low, and Low Low Tags. The limits can be constant.

5. Select or Add Discretes for the Alarm tags (High High, High, Low and Low Low).

Adding tags:

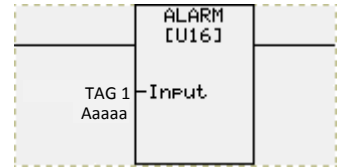
- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Alarm

Alarm:

The Alarm instruction can be used to compare register data values of the Input at memory location Aaaaa with High High at memory location Bbbbb, High at memory location Ccccc, Low at memory location Ddddd, and Low Low at memory location Eeeee.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

The instruction will turn ON/OFF discretives based on following conditions:

- If Aaaaa > Bbbbb then the High High Alarm discrete tag at memory location Fffff will turn ON.
- If Aaaaa > Ccccc then the High Alarm discrete tag at memory location Ggggg will turn ON.
- If Aaaaa < Ddddd then the Low Alarm discrete tag at memory location Hhhhh will turn ON.
- If Aaaaa < Eeeee then the Low Low Alarm discrete tag at memory location Iiiii will turn ON.

Note: Only if Ccccc < Aaaaa < Ddddd then no discretives will be ON. Also when Low Low Alarm is ON the Low Alarm will also be ON. Similarly if High High Alarm is ON the High Alarm will also be ON.

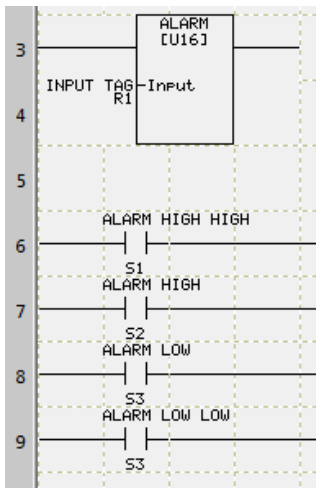
Any of the registers (Low Low, Low, High, or High High) can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type.

Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Memory Type	Syntax (D, E)	Range (dddd)	Range (eeee)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Memory Type	Syntax (F, G, H, I)	Range (ffff)	Range (gggg)	Range (hhhh)	Range (iiii)
Discrete					
Discrete Outputs	O	1-128	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024	1-1024
System Discretes	SD	1-16	1-16	1-16	1-16
Register Discretes					
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.



In this example based on Input (R1) the Normal Open Contacts will turn ON/OFF and execute their logic. For example a pump and drain could be controlled and Input would be water level.

Adding User Defined Fault Instruction

To configure a User Defined Fault Instruction, perform the following steps:

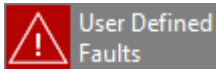
1. Click on the User Defined Fault instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.

Faults	Input 1	Condition	Input 2 Type	Input 2
Fault 1	TEMPERATURE	Greater Than	CONST	60

4. Select or Add a Register for Fault Code Tag.
5. Select or Add Discretes for the Fault Found, and Reset Tag.
6. Use the Add Fault button to add up to 8 faults.
7. For each fault Select or Add an Input 1 Register. Then select the condition from the dropdown. Finally either Select or Add an Input 2 Register (this can also be constant).

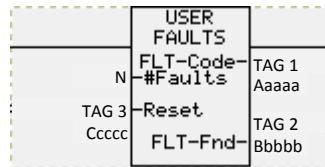
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.

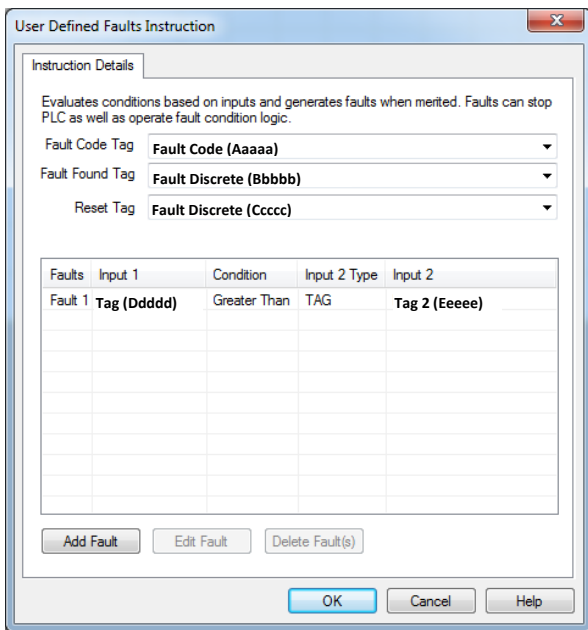


User Defined Faults:

When power flows to the User Defined Fault instruction than it will look at all the user defined conditions and if they are true it will turn ON fault tag at memory location Bbbbb. Also it will indicate the condition which caused the fault in the fault code tag at memory location Aaaaa. This instruction can compare up to 8 conditions and indicates the fault condition by turning ON the corresponding bit to the condition e.g. condition 1 is indicated by bit 0 (register value 1), condition 5 is indicated by bit 4 (register value 16). Use the Reset Fault Tag at memory location Ccccc.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.



Note: If multiple conditions are true then the fault code tag will indicate this with multiple bits being ON e.g. condition 1 & 5 than fault code tag will equal 17 (1+16).

The User Defined Fault instruction will not reset the Fault Code and Fault tag unless the Reset tag turns true. Even if the fault condition is not true anymore, Fault Code and Fault tag will retain the information about the last fault condition seen.

Memory Type	Syntax (B, C)	Range (bbbb)	Range (cccc)
Discrete			
Discrete Outputs	O	1-128	1-128
Discrete Internals	S	1-1024	1-1024
System Discretes	SD	1-16	1-16
Register Discretes			
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15

**Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.*

Note: Memory table for D and E pertains to all 16 tags that can be added for Fault Inputs.

Memory Type	Syntax (A, D, E)	Range (aaaa)	Range (dddd)	Range (eeee)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.

In this example, when the temperature is greater than 60 the Fault Discrete will turn ON (1) and the Fault Code will be 1 (since fault 1). Therefore the logic below will execute and the Fan (O1) will turn ON (1). Also it will try to reset the fault but that will only work if the Temperature goes below 60.

User Defined Faults Instruction

Instruction Details

Evaluates conditions based on inputs and generates faults when merited. Faults can stop PLC as well as operate fault condition logic.

Fault Code Tag:

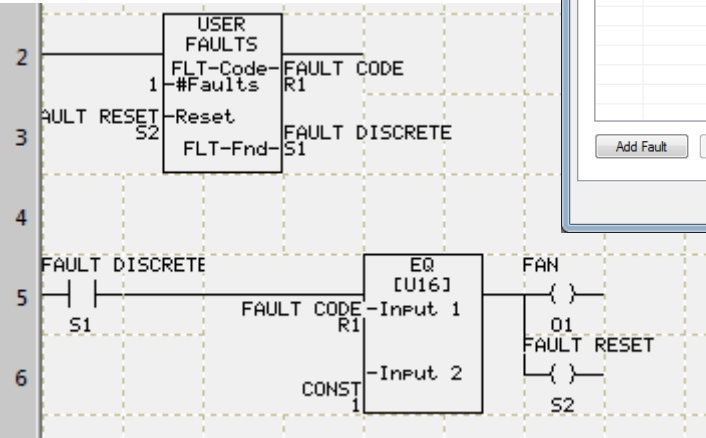
Fault Found Tag:

Reset Tag:

Faults	Input 1	Condition	Input 2 Type	Input 2
Fault 1	TEMPERATURE	Greater Than	CONST	60

Add Fault Edit Fault Delete Fault(s)

OK Cancel Help



3.3.15 Analog

The instructions listed within this chapter perform analog increases or decreases on user specified values or addresses.

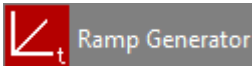
Adding Analog Instructions

To configure all of the various Analog instructions, perform the following steps:

1. Click on any Analog instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.
4. Select or Add the needed Register tags for the different instructions. Noting that all tags need to be of the same datatype. *Note: Ramp Generator will automatically generate three reserved tags starting from the Output tag memory location.*
5. For the Ramp Generator also add a Discrete Overflow tag.
6. For Scale Non-Linear please use the Add point to select or add up to 10 reference points total.

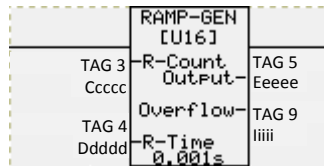
Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Ramp Generator:

When power flows to the Ramp Generator it will increment the output at memory location Eeeee at a set rate based on the rate count at memory location Ccccc and rate time at memory location Ddddd. If the ramp generator reaches either the maximum or minimum as indicate at memory locations Aaaaa and Bbbbb, than the overflow tag at memory location lllll will turn ON. The values for maximum, minimum, rate count, and rate time can be constant. The Rate time is based on the selected time base.



Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

Note: The ramp generator can increment both in the upward and in the downward direction. To decrement please use a negative value for the rate count.

Note: This instruction needs to use three extra registers for internal calculations. Therefore it will auto generate the needed register starting from the memory address of the output Eeeee (Ffff= E(eeee+1), Gggg = E(eeee+2), Hhhh = E(eeee+3)). If double words (INT_32) used then addresses are +2.

Time Base:

Time Base allows the time variable to be mapped to different Time Bases as follows:

- 1 millisecond
- 10 millisecond
- 100 millisecond
- 1 second

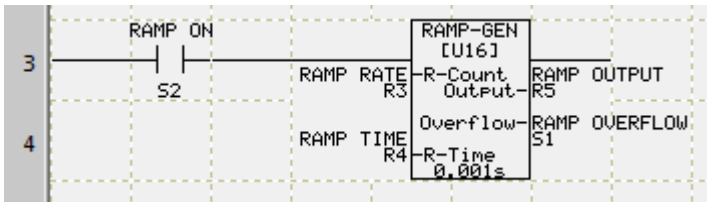
Memory Type	Syntax (A, B, C, D)	Range (aaaa)	Range (bbbb)	Range (cccc)	Range (dddd)
Registers					
Input Registers	IR	1-64	1-64	1-64	
Output Registers	OR	1-64	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20	1-20

Memory Type	Syntax (E, F, G, H)	Range (eeee)	Range (ffff)	Range (gggg)	Range (hhhh)
Registers					
Registers Internals	R	1-16384	1-16384	1-16384	1-16384

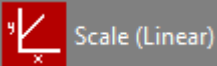
Memory Type	Syntax (I)	Range (iiii)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretes	SD	1-16
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15

**Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.*

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.



In this example when S2 is ON (1), the ramp generator will increment R5 by the amount in R3 every time in R4 milliseconds.

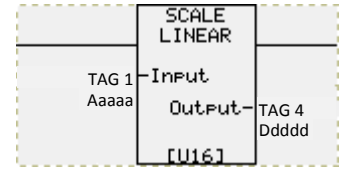


Scale (Linear)

Scale (Linear):

When power flows to the Scale (Linear)

Instruction it will scale the input at memory location Aaaaa to the output at memory location Ddddd. The scaling factor is determined by Point 1 and Point 2 inputs and outputs at memory locations Bbbbb, Ccccc, Eeeee, and Fffff. The values for Inputs and Outputs of Point 1 and Point 2 can be constant. Power will stop flowing through this element if try to divide by zero. The equation for scaling is equal to:



$$Output = Input \times \left(\frac{Point\ 2\ Output - Point\ 1\ Output}{Point\ 2\ Input - Point\ 1\ Input} \right) - Constant$$

Where the constant is determined by solving equation for point 1 and point 2.

Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.

Scale Linear Instruction

Instruction Details

Scales the Input Value to the Output Value using a 2 point reference.

Input

Input Tag:

Data Type: Point 1 Input, Point 1 Output, Point 2 Input and Point 2 Output must be of same data

Example

Point 1: Input Value = 1, Output Value = 10
 Point 2: Input Value = 5, Output Value = 50
 Result: If Input Value = 53, then Output Value = 530

Point 1

Point 1 Input: Tag Name (In Decimal)

Point 1 Output

Point 1 Output: Tag Name (In Decimal)

Point 2

Point 2 Input: Tag Name (In Decimal)

Point 2 Output

Point 2 Output: Tag Name (In Decimal)

Result

Output Tag:

Display all values in:

Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Memory Type	Syntax (D, E, F)	Range (dddd)	Range (eeee)	Range (ffff)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: All Register data types except BCD and ASCII.

Example:

Point 1: Input Value= 1000, Output Value = 0

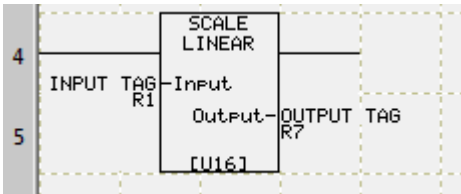
Point 2: Input Value= 4000, Output Value = 100

Equation:

$$Output = Input \times \left(\frac{100 - 0}{4000 - 1000} \right) - \frac{1000}{30}$$

$$Output = \frac{Input}{30} - \frac{1000}{30}$$

Result: If Input Value = 3000, Output Value = 67



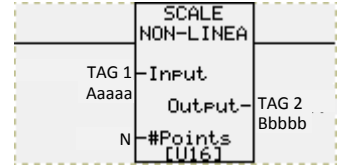
In this example, the value of the input tag R1 is scaled by the settings stored in the output tag R7.

Scale (Non-Linear)

Scale (Non-Linear):

When power flows to the Scale (Non-Linear) Instruction it will scale the input at memory location Aaaaa to the output at memory location Bbbbb. The instruction uses up to 10 reference points with corresponding input and output value. The input is scaled linearly between every two reference points. The overall result is a non-linear approximation through multiple linear approximations. All linear approximations behave like the Scale (Linear) Instruction. Reference points can be taken from multiple memory locations or can be constant. Power will stop flowing through this element if it tries to divide by zero.

Note: This is the same instruction that can also be found in the Function Instruction section. It is also located here for convenience sake.



Scale Non Linear Instruction

Instruction Details

Scales the Input Value to the Output Value non-linearly using up to 10 reference points. It is scaled linearly in between every 2 points.

Input

Input Tag:

Data Type: Output and Points must be of same data type.

Output Tag:

Display all values in:

Points	Input Type	Input	Output Type	Output
Point 1	TAG	Point 1 In (Ccccc)	TAG	Point 1 Out (Ddddd)
Point 2	TAG	Point 2 In (Eeeee)	TAG	Point 2 Out (Fffff)

Use the Add point button to add points. At least 2 points with both Input and Output are needed. With only 2 points the instruction will be a simple linear scale.

The equation for each line between points is the same as the equation used for Scale (Linear).

Note: All points have to be greater than the points before them, example:

Point 2 Input > Point 1 Input

Point 2 Output > Point 1 Output

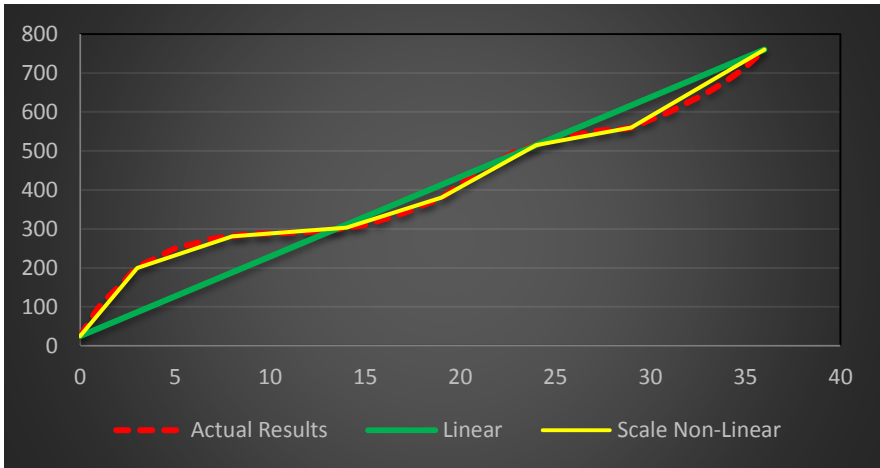
Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Note:

Memory table for C, D, E, and F pertains to all 10 points (inputs and outputs).

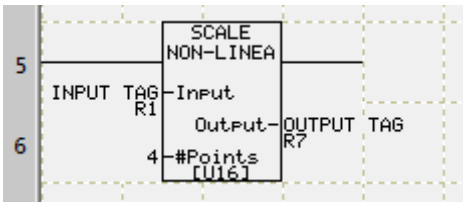
Memory Type	Syntax (C, D, E, F)	Range (cccc)	Range (dddd)	Range (eeee)	Range (ffff)
Registers					
Input Registers	IR	1-64	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20	1-20

Allowed Data Formats: All Register data types except BCD and ASCII.



This graph illustrates the difference between Scale (Linear) and Scale (Non-Linear). The red line shows the actual conditions that you would like to scale sensor information to. For example the volume of an asymmetrical tank. If the sensor data were to be scaled using

just Scale (Linear) you would have the green line with its large error in some places. On the other hand using the Scale (Non-Linear) the actual result can be approximated as seen by the yellow line.



In this example, the value of the input tag R1 is scaled by the settings stored in the output tag R7.

3.3.16 Function Blocks

The instructions listed within this chapter perform advanced functions with multiple tag inputs and outputs.

Adding Function Block Instructions

To configure all of the various Function Block instructions, perform the following steps:

1. Click on any Function Block instruction icon on the right side of the screen.
2. Position the mouse over the Ladder diagram and click the mouse to place the instruction.
3. To enter Data/Display types, double click the instruction to open its Dialog box.
4. Select or Add the needed Register tags for the different instructions. Noting that all tags need to be of the same datatype. *Note: Instructions are very different. Some need many tags others need a few.*
5. Most instructions also exist in other locations please see *Sections 3.3.3, 3.3.4, 3.3.9, 3.3.13 and 3.3.14.*
6. Instructions Change of Value, Find Min & Max Value, Flasher and Ramp Generator will generate extra internal calculation tags.

Adding tags:

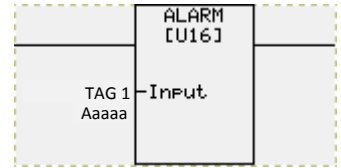
- a. If need be you can add a new tag by entering a new Tag Name
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



Alarm

Alarm:

The Alarm instruction can be used to compare register data values of the Input at memory location Aaaaa with High High at memory location Bbbbb, High at memory location Ccccc, Low at memory location Ddddd, and Low Low at memory location Eeeee.



Note: This is the same instruction that can also be found in the Process Alarms/Faults Instruction section. It is also located here for convenience sake.

The instruction will turn ON/OFF discrettes based on following conditions:

- If Aaaaa > Bbbbb than the High High Alarm discrete tag at memory location Ffff will turn ON.
- If Aaaaa > Ccccc than the High Alarm discrete tag at memory location Ggggg will turn ON.
- If Aaaaa < Ddddd than the Low Alarm discrete tag at memory location Hhhhh will turn ON.
- If Aaaaa < Eeeee than the Low Low Alarm discrete tag at memory location Iiiii will turn ON.

Note: Only if Ccccc < Aaaaa < Ddddd then no discrettes will be ON. Also when Low Low Alarm is ON the Low Alarm will also be ON. Similarly if High High Alarm is ON the High Alarm will also be ON.

Any of the registers (Low Low, Low, High, or High High) can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type.

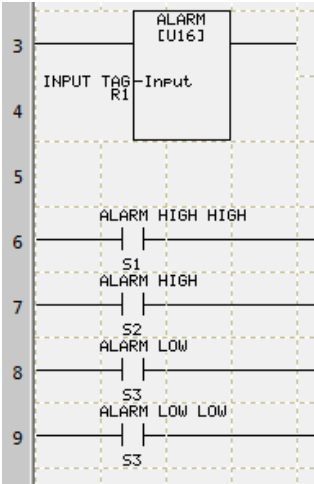
Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Memory Type	Syntax (D, E)	Range (dddd)	Range (eeee)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Memory Type	Syntax (F, G, H, I)	Range (ffff)	Range (gggg)	Range (hhhh)	Range (iiii)
Discrete					
Discrete Outputs	O	1-128	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024	1-1024
System Discretes	SD	1-16	1-16	1-16	1-16
Register Discretes					
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15

**Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.*

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.



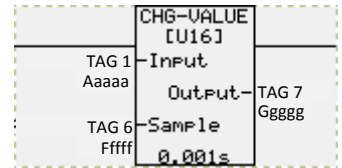
In this example based on Input (R1) the Normal Open Contacts will turn ON/OFF and execute their logic. For example a pump and drain could be controlled and Input would be water level.



Change of Value

Change of Value:

The Change of Value instruction can be used to find the change of an Input at memory location Aaaaa during a set amount of time. When power flows to this instruction then it will output the change in the register value of the Input after a selected time interval. The time interval is the sample rate at memory location Fffff, or it can be constant. The time interval is set using the Time Base. The change of value will output to register at memory address Ggggg. When power flow is off then output register value will be 0. Power will always flow through this element. Input and Output registers have to be of the same type.



Change of Value Instruction

Instruction Details

Finds the change of value of 1 input between two points of time.
 (Input at Time 2) - (Input at Time 1) = Change of Value where T2 > T1

Input Tag

Reserved Tags for internal calculations (4 tags after Input Tag address needed)

Tag Name	Address
INPUT TAG.RES1	Bbbbb
INPUT TAG.RES2	Ccccc
INPUT TAG.RES3	Ddddd
INPUT TAG.RES4	Eeeee

Sample Rate

Tag Name

Constant (In Decimal)

Time Base

Output Tag

OK Cancel Help

Note: For internal calculations this instruction will automatically generate four extra registers where each register memory address is 1 greater than previous register (e.g. Bbbbb = Aaaaa + 1). If double word (INT_32) used then memory addresses generate will be +2.

Time Base:

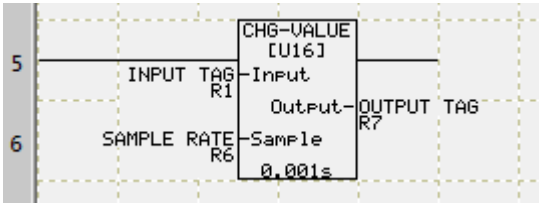
Time Base allows the time variable to be mapped to different Time Bases as follows:

- 1 millisecond
- 10 millisecond
- 100 millisecond
- 1 second

Memory Type	Syntax (A, B, C, D)	Range (aaaa)	Range (bbbb)	Range (cccc)	Range (dddd)
Registers					
Registers Internals	R	1-16384	1-16384	1-16384	1-16384

Memory Type	Syntax (E, F, G)	Range (eeee)	Range (ffff)	Range (gggg)
Registers				
Output Registers	OR		1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR		1-20	1-20

Allowed Data Formats: All Register data types except BCD and ASCII.

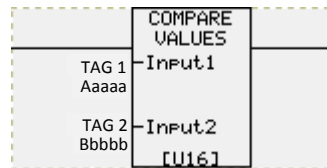


In this example, when the input tag R1 changes, the amount of change per sample rate R6 will be outputted to R7.



Compare Values:

The Compare Values instruction can be used to compare two Inputs, Input 1 at memory location Aaaaa and Input 2 at memory location Bbbbb. Based on the Input 1 and Input 2 comparison the corresponding discrete tag (Ccccc, Ddddd, or Eeeee) will turn on. This instruction examines whether Input 1 and Input 2 are greater than, equal, or less than. Either Input can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type. Power constantly flow through this element.



Note: This is the same instruction that can also be found in the Compare Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Memory Type	Syntax (C,D, E)	Range (cccc)	Range(ddd)	Range(eeee)
Discrete				
Discrete Outputs	O	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024
Register Discretcs				
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15

*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete, all register data type except BCD and ASCII.

Compare Values Instruction

Instruction Details

Compares 2 Inputs and evaluates whether Input 1 (>, <, =) Input 2. Turns on discretés correspondingly.

Inputs

Input 1
 Input 1 Tag: TAG 1
 Data Type: UNSIGNED_INT_16
 Input 2 must be of same data type.

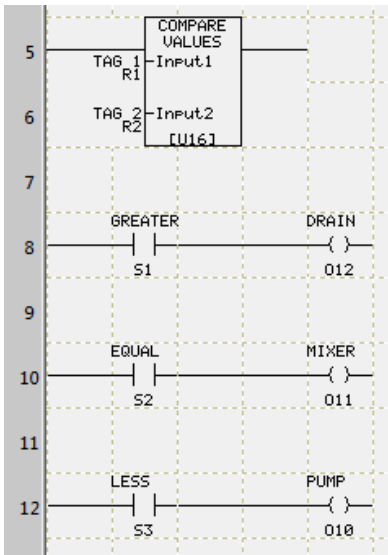
Input 2
 Tag Name: TAG 2
 Constant: (In Decimal)

Results

Greater Than Tag (Input 1 > Input 2): GREATER
 Equal To Tag (Input 1 = Input 2): EQUAL
 Less Than Tag (Input 1 < Input 2): LESS

Display all values in: Decimal

OK Cancel Help



In this example:

- If $R1 > R2$ then S1 will be ON and O12 will be energized.
- If $R1 = R2$ then S2 will be ON and O11 will be energized.
- If $R1 < R2$ then S3 will be ON and O10 will be energized.

MAX Find Min & Max
MIN Value

Find Min & Max Value:

The Find Min & Max instruction looks at the Input Value at memory location Aaaaa and records the largest and smallest number reached by the Input. When power flows then it will always compare max value at memory location Eeeee, min value at memory location Ddddd, and current value at memory location Aaaaa. If current value is greater than max value, the current value will become the new max value. Also if current value is less than min value, the current value will become the new min value. The reset register at memory location Ccccc, will reset the min/max values to



zero if there is no power flow or to current value if there is power flow. Please see condition table for more information. All registers have to be of the same data type. Power will always flow through this element.

Note: For internal calculations instruction will automatically generate one extra register where Bbbbb = (Aaaaa + 1).

Note: Instruction only functions as described in description when Power ON and Reset OFF.

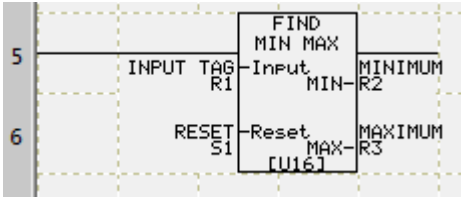
Condition table

Power Flow	Reset Tag	Min Value	Max Value
Power ON	Reset OFF	Min Value (Instruction Functions)	Max Value (Instruction Functions)
Power ON	Reset ON	Current Value	Current Value
Power OFF	Reset ON	0	0
Power OFF	Reset OFF	Last recorded Min from when power ON	Last recorded Max from when power ON

Memory Type	Syntax (A, B, D, E)	Range (aaaa)	Range (bbbb)	Range (dddd)	Range (eeee)
Registers					
Output Registers	OR			1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR			1-20	1-20

Memory Type	Syntax (C)	Range (cccc)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.		

Allowed Data Formats: Discrete, all register data type except BCD and ASCII.

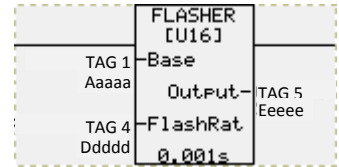


In this example, the outputs R2 and R3 will have the highest and lowest value the input tag R1 has reached since this instruction was last reset.



Flasher:

The flasher will turn ON/OFF a specified Output discrete at memory location Eeeee. When power flows to this instruction then the Output discrete will be ON for time value from flash rate register at memory location Ccccc and then OFF for the same time.



Note: For internal calculations this instruction will automatically generate two extra registers where each register memory address is 1 greater than previous register (e.g. Bbbbb = Aaaaa + 1).

Time Base:

Time Base allows the time variable to be mapped to different Time Bases as follows:

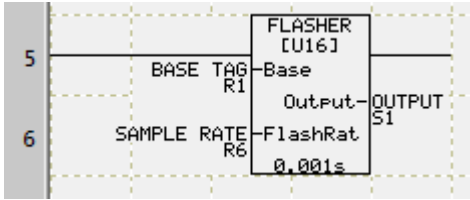
- 1 millisecond
- 10 millisecond
- 100 millisecond
- 1 second

Memory Type	Syntax (A, B, C, D)	Range (aaaa)	Range (bbbb)	Range (cccc)	Range (dddd)
Registers					
Output Registers	OR				1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR				1-20

Allowed Data Formats: Discrete, all register data type except BCD and ASCII.

Memory Type	Syntax (E)	Range (eeee)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.		

Allowed Data Formats: Discrete and UNSIGNED_INT_16

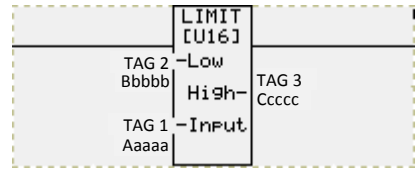


In this example, the output S1 is ON for the sample rate R6 milliseconds and then OFF for the same amount of time. The base tag R1 is used for internal calculations, please do not use for anything else.

LIM Limit

Limit:

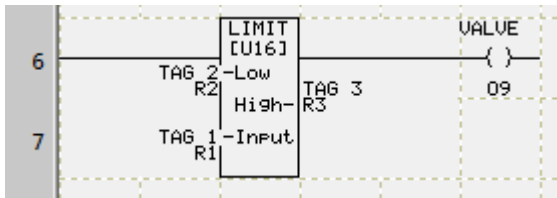
The Limit instruction can be used to compare register data values of the Input at memory location Aaaaa with Low at memory location Bbbbb and High at memory location Ccccc. If $Aaaa \leq Ccccc$ and $Aaaa \geq Bbbbb$ then power will flow through this element. Any of the registers (Input, High or Low) can be assigned a constant value. Values can be displayed in Decimal, Hex, or Octal format. All the registers must be of the same data type.



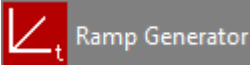
Note: This is the same instruction that can also be found in the Compare Instruction section. It is also located here for convenience sake.

Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Allowed Data Formats: all register data type except BCD and ASCII.

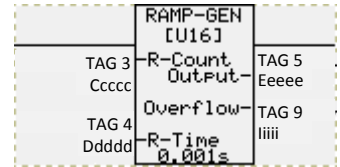


In the example above, if the input R1 is within R2 and R3, power will flow out and O9 will be energized.



Ramp Generator:

When power flows to the Ramp Generator it will increment the output at memory location Eeeee at a set rate based on the rate count at memory location Ccccc and rate time at memory location Ddddd. If the ramp generator reaches either the maximum or minimum as indicate at memory locations Aaaaa and Bbbbb, than the overflow tag at memory location lllll will turn ON. The values for maximum, minimum, rate count, and rate time can be constant. The Rate time is based on the selected time base.



Note: This is the same instruction that can also be found in the Analog Instruction section. It is also located here for convenience sake.

Ramp Generator Instruction

Instruction Details

Increments/Decrements an output value at a constant rate.

Limits

Maximum Output
 Tag Name: **Maximum (Aaaaa)**
 Constant: [] (In Decimal)
 Data Type: UNSIGNED_INT_16

Minimum Output
 Tag Name: **Minimum (Bbbbb)**
 Constant: [] (In Decimal)

Ramp Parameters

Ramp Rate Count
 Tag Name: **Ramp Rate (Ccccc)**
 Constant: [] (In Decimal)

Ramp Rate Time
 Tag Name: **Ramp Time (Ddddd)**
 Constant: [] (In Decimal)
 Time Base: MSec

Display all values in: Decimal

Results

Current Output Tag
Output (Eeeee)

Reserved Tags for internal calculations (3 tags after Output Tag address needed)

Tag Name	Address
RAMP OUTPUT.RES1	Ffff
RAMP OUTPUT.RES2	Gggg
RAMP OUTPUT.RES3	Hhhh

Overflow Tag
Ramp Overflow (lllll)

OK Cancel Help

Note: The ramp generator can increment both in the upward and in the downward direction. To decrement please use a negative value for the rate count.

Note: This instruction needs to use three extra registers for internal calculations. Therefore it will auto generate the needed register starting from the memory address of the output Eeeee (Ffff= E(eeee+1), Gggg = E(eeee+2), Hhhh = E(eeee+3)). If double words (INT_32) used then addresses are +2.

Time Base:

Time Base allows the time variable to be mapped to different Time Bases as follows:

- 1 millisecond
- 10 millisecond
- 100 millisecond
- 1 second

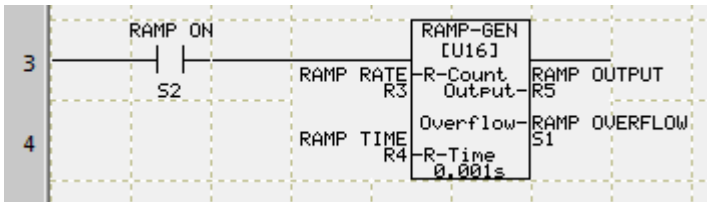
Memory Type	Syntax (A, B, C, D)	Range (aaaa)	Range (bbbb)	Range (cccc)	Range (dddd)
Registers					
Input Registers	IR	1-64	1-64	1-64	
Output Registers	OR	1-64	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20	1-20

Memory Type	Syntax (E, F, G, H)	Range (eeee)	Range (ffff)	Range (gggg)	Range (hhhh)
Registers					
Registers Internals	R	1-16384	1-16384	1-16384	1-16384

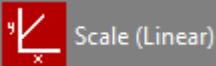
Memory Type	Syntax (I)	Range (iiii)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretes	SD	1-16
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15

**Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.*

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.



In this example when S2 is ON (1), the ramp generator will increment R5 by the amount in R3 every time in R4 milliseconds.

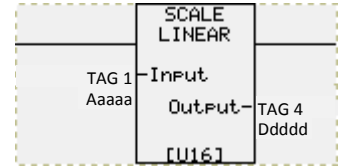


Scale (Linear)

Scale (Linear):

When power flows to the Scale (Linear)

Instruction it will scale the input at memory location Aaaaa to the output at memory location Ddddd. The scaling factor is determined by Point 1 and Point 2 inputs and outputs at memory locations Bbbbb, Ccccc, Eeeee, and Fffff. The values for Inputs and Outputs of Point 1 and Point 2 can be constant. Power will stop flowing through this element if try to divide by zero. The equation for scaling is equal to:



$$Output = Input \times \left(\frac{Point\ 2\ Output - Point\ 1\ Output}{Point\ 2\ Input - Point\ 1\ Input} \right) - Constant$$

Where the constant is determined by solving equation for point 1 and point 2.

Note: This is the same instruction that can also be found in the Analog Instruction section. It is also located here for convenience sake.

Scale Linear Instruction

Instruction Details

Scales the Input Value to the Output Value using a 2 point reference.

Input

Input Tag:

Data Type: Point 1 Input, Point 1 Output, Point 2 Input and Point 2 Output must be of same data

Example

Point 1: Input Value = 1, Output Value = 10
 Point 2: Input Value = 5, Output Value = 50
 Result: If Input Value = 53, then Output Value = 530

Point 1

Point 1 Input: Tag Name (In Decimal)

Point 1 Output

Point 1 Output: Tag Name (In Decimal)

Point 2

Point 2 Input: Tag Name (In Decimal)

Point 2 Output

Point 2 Output: Tag Name (In Decimal)

Result

Output Tag:

Display all values in:

Memory Type	Syntax (A, B, C)	Range (aaaa)	Range (bbbb)	Range (cccc)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Memory Type	Syntax (D, E, F)	Range (dddd)	Range (eeee)	Range (ffff)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: All Register data types except BCD and ASCII.

Example:

Point 1: Input Value= 1000, Output Value = 0

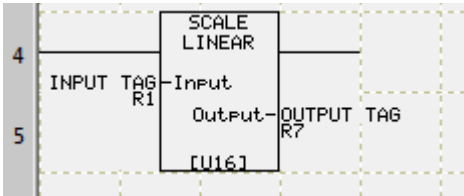
Point 2: Input Value= 4000, Output Value = 100

Equation:

$$Output = Input \times \left(\frac{100 - 0}{4000 - 1000} \right) - \frac{1000}{30}$$

$$Output = \frac{Input}{30} - \frac{1000}{30}$$

Result: If Input Value = 3000, Output Value = 67

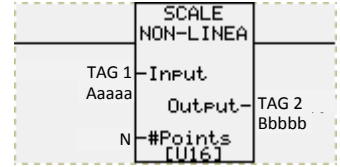


In this example, the value of the input tag R1 is scaled by the settings stored in the output tag R7.

Scale (Non-Linear)

Scale (Non-Linear):

When power flows to the Scale (Non-Linear) Instruction it will scale the input at memory location Aaaaa to the output at memory location Bbbbb. The instruction uses up to 10 reference points with corresponding input and output value. The input is scaled linearly between every two reference points. The overall result is a non-linear approximation through multiple linear approximations. All linear approximations behave like the Scale (Linear) Instruction. Reference points can be taken from multiple memory locations or can be constant. Power will stop flowing through this element if it tries to divide by zero.



Note: This is the same instruction that can also be found in the Analog Instruction section. It is also located here for convenience sake.

Scale Non Linear Instruction

Instruction Details

Scales the Input Value to the Output Value non-linearly using up to 10 reference points. It is scaled linearly in between every 2 points.

Input

Input Tag:

Data Type: Output and Points must be of same data type.

Output Tag:

Display all values in:

Points	Input Type	Input	Output Type	Output
Point 1	TAG	Point 1 In (Ccccc)	TAG	Point 1 Out (Ddddd)
Point 2	TAG	Point 2 In (Eeeee)	TAG	Point 2 Out (Fffff)

Use the Add point button to add points. At least 2 points with both Input and Output are needed. With only 2 points the instruction will be a simple linear scale.

The equation for each line between points is the same as the equation used for Scale (Linear).

Note: All points have to be greater than the points before them, example:

Point 2 Input > Point 1 Input

Point 2 Output > Point 1 Output

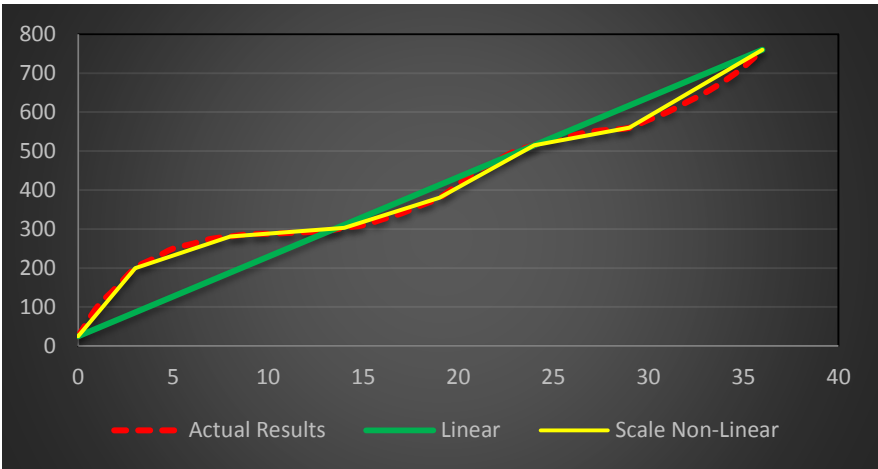
Memory Type	Syntax (A, B)	Range (aaaa)	Range (bbbb)
Registers			
Input Registers	IR	1-64	1-64
Output Registers	OR	1-64	1-64
Registers Internals	R	1-16384	1-16384
System Registers	SR	1-20	1-20

Note:

Memory table for C, D, E, and F pertains to all 10 points (inputs and outputs).

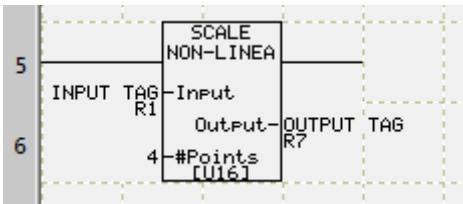
Memory Type	Syntax (C, D, E, F)	Range (cccc)	Range (dddd)	Range (eeee)	Range (ffff)
Registers					
Input Registers	IR	1-64	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20	1-20

Allowed Data Formats: All Register data types except BCD and ASCII.

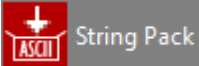


This graph illustrates the difference between Scale (Linear) and Scale (Non-Linear). The red line shows the actual conditions that you would like to scale sensor information to. For example the volume of an asymmetrical tank. If the sensor data were to be scaled using

just Scale (Linear) you would have the green line with its large error in some places. On the other hand using the Scale (Non-Linear) the actual result can be approximated as seen by the yellow line.

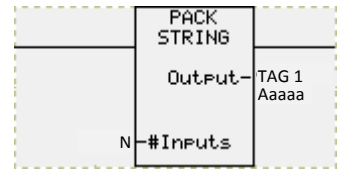


In this example, the value of the input tag R1 is scaled by the settings stored in the output tag R7.



String Pack:

When power flows through this element, the String Pack takes any register type and combines them into an ASCII string. The resulting ASCII string is stored at memory location Aaaaa. The instruction takes up to 16 different inputs from multiple different memory locations. Any input can also be a constant.



Note: This is the same instruction that can also be found in the String Instruction section. It is also located here for convenience sake.

For each input the string length to pack needs to be specified. If an input has a floating point data type it can either be rounded off to the nearest integer value or truncated. When an input of integer or floating point data value is converted, the number of digits, decimal position and justification (leading zeros, leading spaces, or trailing spaces) must be assigned. Power will stop flowing through this element if an error occurs such as overflow, underflow, or divide by zero.

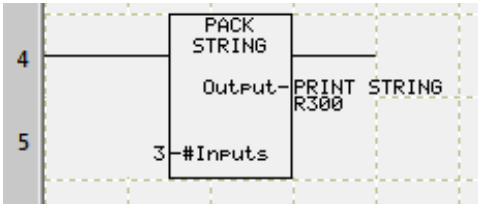
The String Pack instruction will only combine the inputted length specified for each input element even if the input's size is greater e.g. if length specified is 10 but ASCII string is 12 long will only pack the first 10 characters of the ASCII String.

Note: For signed registers and registers with decimals please note that the length to pack needs to be one greater than the number of digits due to the need for a +/- sign and or decimal point. If number has both then the length to pack needs to be 2 greater than the number of digits.

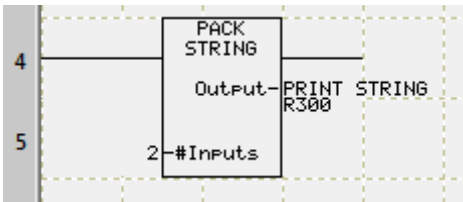
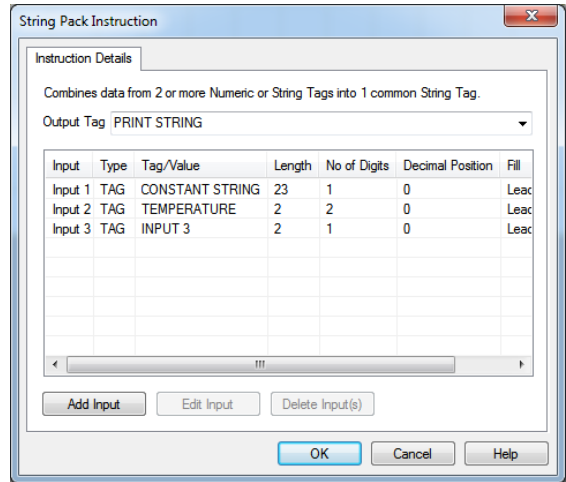
Note: For Floating point registers be mindful of the scientific notation. Based on the numeric value of the register the number can be displayed in decimal notation (##.##) or scientific notation (###e+###). For scientific notation the length to pack needs to be four greater than the number of digits.

Memory Type	Syntax (A, Inputs)	Range (aaaa)	Inputs
Registers			
Output Registers	OR		1-64
Registers Internals	R	1-16384	1-16384

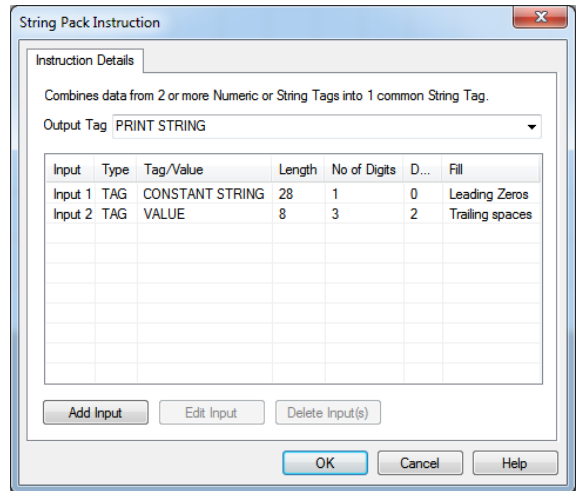
Allowed Data Formats: all register data types.



In this example, string R100, register R20, and string R120 are packed together into R300 when power flows. For this example it would create string "Current Temperature is 20°C". With string R100 being 23 Char long ("Current Temperature is "). The register R20 would be the temperature with length 2 and Number of digits 2 as well ("20"). Finally string R120 would be 2 char long ("°C").



In this example, string R100, and register R20 are packed together into R300 when power flows. For this example it would create string "Current calculated value is 2.45e+03". With string R100 being 28 Char long ("Current calculated value is "). The register R20 would be the value with length 8 ("2.45e+03"), Number of digits 3 ("2.45"), and Decimal Position is 2 (".45").

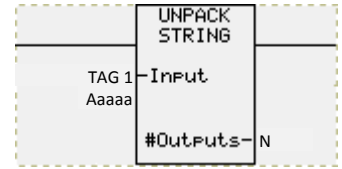




String Unpack

String Unpack:

When power flows through this element, the String Unpack takes an ASCII String and splits it into different registers of any type. The Input ASCII string is stored at memory location Aaaaa. The instruction outputs to up to 16 different outputs from multiple different memory locations. For each output the string length to unpack needs to be specified. Power will stop flowing through this element if an error occurs such as overflow, underflow, or divide by zero.



Note: This is the same instruction that can also be found in the String Instruction section. It is also located here for convenience sake.

The instruction will split the ASCII string starting from the beginning into the different registers per the specified length. Therefore if the length of the Input String is shorter than the total length of the outputs, only the outputs up to the Input String length will be split into.

Note: An ASCII register takes 2 char per register address, e.g. for 10 char need R1-R5. Only the first address needs to be specified, the others are automatically used.

Note: An ASCII character cannot be split into a non-ASCII register. Therefore if such is attempted the output register will be set to 0, e.g. if try to input AB into UNSIGNED_INT_16 this will result in the register being 0.

Note: For Floating point registers be mindful of the scientific notation. Floating point values can be displayed in decimal notation (##.##) or scientific notation (###e+###). For scientific notation the length to unpack needs to be four greater than the number of digits.

Memory Type	Syntax (A, Outputs)	Range (aaaa)	Outputs
Registers			
Output Registers	OR		1-64
Registers Internals	R	1-16384	1-16384

Allowed Data Formats: all register data types.

Examples of how it will unpack different length strings:

Example 1:

Input: ABCDEF

If splitting into Output 1 (Length 10) and Output 2 (Length 10).

Then only Output 1 will be used since Input length < Output 1 length.

Output 1: ABDCEF ____

Output 2: _____

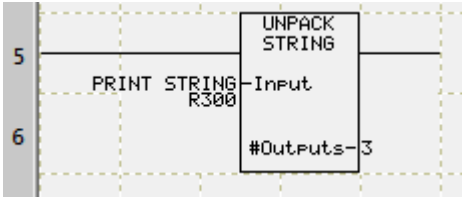
Example 2:

Input: ABCDEFGHIJK

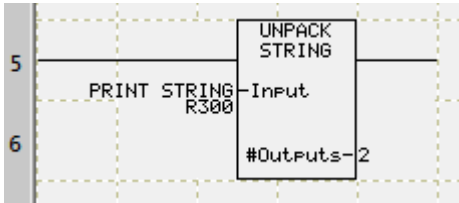
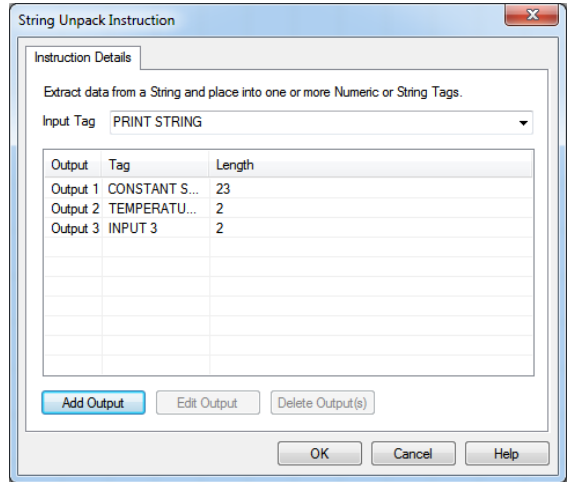
If splitting into Output 1 (Length 10) and Output 2 (Length 10).

Then both Output 1 and Output 2 will be used but since Input length < (Output 1 + Output 2) length, then only some of Output 2 will be used.

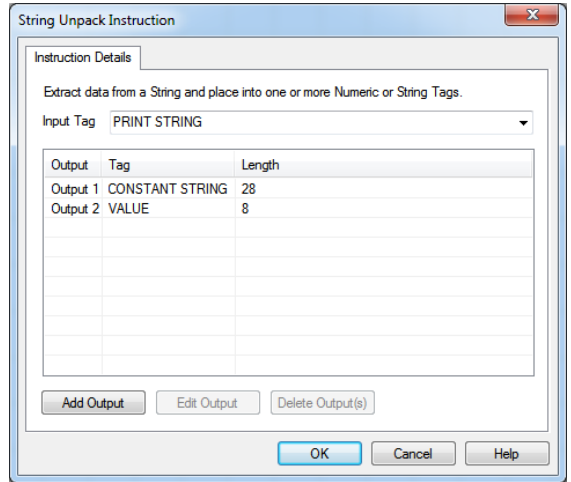
Output 1: ABCDEFGHIJ
 Output 2: K_____



In this example, R300 is unpacked into string R100, register R20, and string R120 when power flows. For this example it would take apart string "Current Temperature is 20°C". With string R100 being 23 Char long ("Current Temperature is "). The register R20 would be the temperature with length 2 ("20"). Finally string R120 would be 2 char long ("°C").



In this example, R300 is unpacked into string R100 and register R20 when power flows. For this example it would take apart string "Current calculated value is 2.45e+03". With string R100 being 28 Char long ("Current calculated value is "). The register R20 would be the value with length 8 ("2.45e+03").

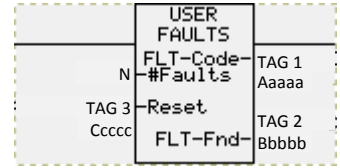




User Defined Faults

User Defined Faults:

When power flows to the User Defined Fault instruction than it will look at all the user defined conditions and if they are true it will turn ON fault tag at memory location Bbbbb. Also it will indicate the condition which caused the fault in the fault code tag at memory location Aaaaa. This instruction can compare up to 8 conditions and indicates the fault condition by turning ON the corresponding bit to the condition e.g. condition 1 is indicated by bit 0 (register value 1), condition 5 is indicated by bit 4 (register value 16). Use the Reset Fault Tag at memory location Ccccc.



Note: This is the same instruction that can also be found in the Process Alarms/Faults Instruction section. It is also located here for convenience sake.

Note: If multiple conditions are true then the fault code tag will indicate this with multiple bits being ON e.g. condition 1 & 5 than fault code tag will equal 17 (1+16).

The User Defined Fault instruction will not reset the Fault Code and Fault tag unless the Reset tag turns true. Even if the fault condition is not true anymore, Fault Code and Fault tag will retain the information about the last fault condition seen.

Memory Type	Syntax (B, C)	Range (bbbb)	Range (cccc)
Discrete			
Discrete Outputs	O	1-128	1-128
Discrete Internals	S	1-1024	1-1024
System Discretes	SD	1-16	1-16
Register Discretes			
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15

**Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.*

Note: Memory table for D and E pertains to all 16 tags that can be added for Fault Inputs.

Memory Type	Syntax (A, D, E)	Range (aaaa)	Range (dddd)	Range (eeee)
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

Allowed Data Formats: Discrete and all Register data types except BCD and ASCII.

In this example, when the temperature is greater than 60 the Fault Discrete will turn ON (1) and the Fault Code will be 1 (since fault 1). Therefore the logic below will execute and the Fan (O1) will turn ON (1). Also it will try to reset the fault but that will only work if the Temperature goes below 60.

User Defined Faults Instruction

Instruction Details

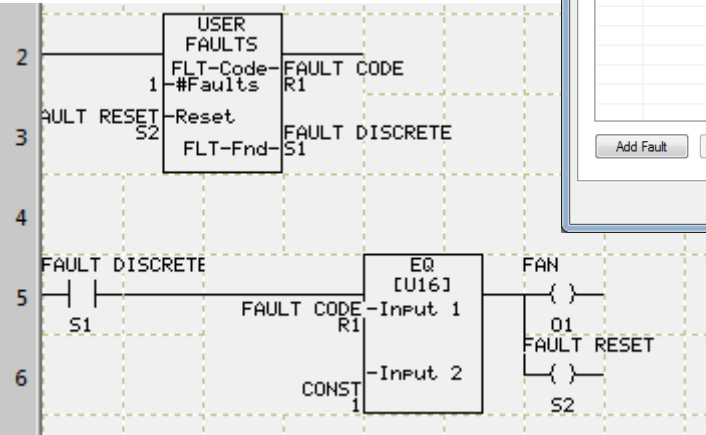
Evaluates conditions based on inputs and generates faults when merited. Faults can stop PLC as well as operate fault condition logic.

Fault Code Tag:

Fault Found Tag:

Reset Tag:

Faults	Input 1	Condition	Input 2 Type	Input 2
Fault 1	TEMPERATURE	Greater Than	CONST	60



3.3.17 IIoT

Use the IIoT (MQTT) Publish to send data to an IIoT (MQTT) Broker. The IIoT (MQTT) Publish allows you to publish data for the Industrial Internet of Things (IIoT). For more information about IIoT and how it works please see *Chapter 8*.

Adding the Log IIoT (MQTT) Publish Instruction:

To configure the IIoT (MQTT) Publish instruction, perform the following steps:

1. Click on the IIoT (MQTT) Publish icon on the right side of the screen.
2. Position the mouse over the Ladder Tags diagram and click the mouse to place the instruction.

The screenshot shows the 'IIoT (MQTT) Publish Instruction' dialog box. It contains the following fields and controls:

- Instruction Details:** Includes a 'MQTT Setup' button.
- Broker and Topic:** 'Publish to Broker: 192.168.0.1:1883' and a 'Topic' dropdown menu.
- Publish:** 'Publish Type' (At Regular Time Intervals (When Enable Tag is High)), 'Event/Enable Tag' (ENABLE), 'Publish Time-interval' (1 Minute), and 'Publish Status Tag' (STATUS).
- Select Tags:** 'Tag Names are used as column headers.' and 'Decimal Places for Floating Point Tags' (5).
- Available Tags:** A table with columns 'Name', 'Address', and 'Type'.
- Selected Tags:** A table with columns 'Name', 'Address', and 'Type' containing two entries: 'MOTOR.SPEED R1 UNSIGNED_INT_16' and 'MOTOR S1 DISCRETE'.
- Buttons:** 'Delete Tag(s)', 'Move Tag Up', 'Move Tag Down', 'OK', 'Cancel', and 'Help'.

3. Double click the instruction to open its dialog box.

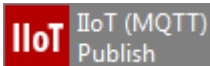
4. Select a topic from the dropdown. If you would like to create a topic use the MQTT Setup to create topics and configure your MQTT Broker. Please see *Section 2.5.8* and *Chapter 8* for more information.

5. Select of Add a tag for Event/Enable and Status.

6. Select the Publish Type from the drop down. Descriptions are on next page.
7. Enter Publish Time Interval for time based data publishing and select time base using the drop down options.
8. Finally select up to 10 tags to publish. Each tag will be published as its own separate topic with the previously selected topic put in front (example from above Topic/Motor).

Adding tags:

- a. If need be you can add a new tag by entering a new Tag Name.
- b. Now either press enter or right click on the Tag Name and the Add New Tag Details dialogue will appear.
- c. Enter the Tag Address in this screen.



IIoT (MQTT) Publish

When power flows through this instruction, the IIoT (MQTT) Publish instruction will send the current value of up to 10 tags to an MQTT Broker. For more information on how IIoT works and how to setup your broker please see *Chapter 8*. You can use up to a maximum of 8 of these instructions.



IIoT (MQTT) Publish Instruction

Instruction Details

Broker and Topic

Publish to Broker: 192.168.0.1:1883 MQTT Setup

Topic

Retain Message: No, QoS: At Most Once

Publish

Publish Type: At Regular Time Intervals (When Enable Tag is High)

Event/Enable Tag: Event/Enable Tag (Aaaaa)

Publish Time-interval: 1 Minute

Publish Status Tag: Data Log Status (Bbbbb)

Status value definitions:
 00: Normal operation (No Errors)
 02: Connect failure
 04: Publish failure
 64: Done

Select Tags

Tag Names are used as column headers.

Available Tags:

Name	Address	Type
Value 2	Ddddd	DISCRETE
Value 3	Eeeee	UNSIGNED_INT_16

Decimal Places for Floating Point Tags: 5

Selected Tags: (1/10)

Name	Address	Type
Value	Ccccc	UNSIGNED_INT_16

>> <<

Delete Tag(s)
Move Tag Up
Move Tag Down

OK Cancel Help

Topic (MQTT Setup)

Please use the MQTT Setup to configure your broker and add topics. For more information on how to do this please see *Section 2.5.8* and *Chapter 8*. Use the dropdown to select any configured topic here in the Publish Instruction.

Publish Type:

There are four types of selectable options for data type:

- **On Rising Edge of Event Tag**
This options allows for **Event Based** only publishing. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only publishes if this tag goes from OFF (0) to ON (1).
- **On Falling Edge of Event Tag**
This options allows for **Event Based** only publishing. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only publishes if this tag goes from ON (1) to OFF (0).
- **On Both Edges of Event Tag**
This options allows for **Event Based** only publishing. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and only publishes if this tag goes from ON (1) to OFF (0) or if it goes OFF (1) to ON (1). Combined other two options.
- **At Regular Time Intervals (When Enable Tag is High)**
This options allows for **Time Based** only publishing and **Event/Time Based** combined publishing. This options looks at the discrete Event/Enable tag stored at memory location Ddddd and publishes at the specified Log Time-interval if this tag is ON.
 - **Time Based:** To do Time Based only publishing just either turn ON (1) the Event/Enable tag or set the Event/Enable tag to have an initial value of 1 (ON).
 - **Event/Time Based:** To do Event/Time Based only publishing just use the Event/Enable tag as the event control. As soon as the Event/Enable tag is ON (1) the instruction will publish at specified Log Time-interval until this tag is turned OFF (0).

Publish Time-interval

The IIoT (MQTT) Publish instruction will publish based on the Log Time-interval if the Publish Type is **Time Based** or **Event/Time Based**. Use the time base dropdown to set the units and enter the amount of time between each publish.

Time Base:

The Time Base is user selectable and allows one of the following time bases:

- Minute
- Hour

e.g. If Publish Time Interval = 15 and Time Base = Minute, then the instruction will publish every 15 Minutes. Similarly, if Publish Time Interval =11 and Time Base = Hour, then the instruction will publish every 11 hours.

Publish Status:

The publish instruction will display current status of the instruction in this tag stored at memory location Bbbbb. More than 1 status can be true at a time so if status is 66 then it is status 2 and 64. Please see below for the available statuses:

Status Value	Description	Explanation
00	Currently Data Logging (Normal Operation)	This means the data log instruction is currently writing to the .csv file.
02	File Open Error	Cannot open .csv file. USB drive may not be plugged in.
04	File Write Error	Cannot write to .csv file. USB driver might be full. Or the .csv file may be read only.
64	Done Data Logging	Have finished writing to the .csv file. Happens if the instruction no longer has power or when it is data logging only at set time intervals.

Selected Tags for Publishing:

The available tags field will have all the currently created discretes and registers for the project. To add a tag to be data, just select the tag and press the >> button to move it over to the selected tags. Tag from any location can be published. If publishing floating point tags the Decimal Places for Floating Point field will give how many decimals are logged (if 5 selected then number will be #.#####).

Note: All tags will be published in the format timestamp, value. Therefore if not using EZAutomation's EZ-IIoT Subscriber Utility then will see values like "1499424774,3218" where 3218 is the value of the tag and 1499424774 is the timestamp in seconds.

Memory Type	Syntax (A)	Range (aaaa)
Discrete		
Discrete Outputs	O	1-128
Discrete Internals	S	1-1024
System Discretes	SD	1-16
Register Discretes		
Bit Access Registers*	R	1-16384 / 0-15
<small>*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.</small>		

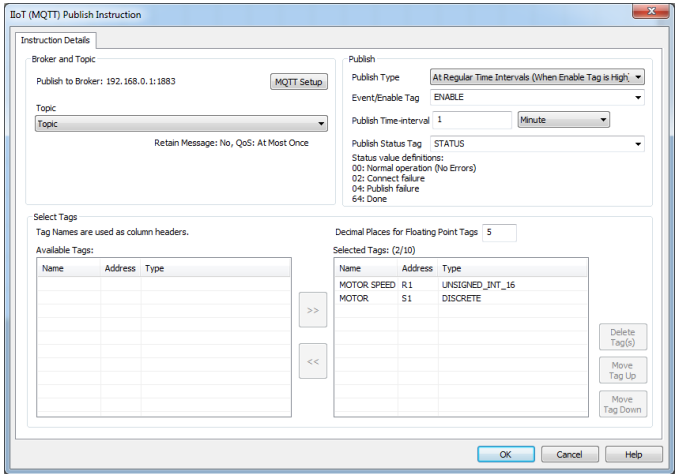
Memory Type	Syntax (B)	Range (bbbb)
Registers		
Output Registers	OR	1-64
Registers Internals	R	1-16384
System Registers	SR	1-20

Note: Memory table for C, D and E pertains to all 10 tags that can be added to Publish Instruction.

Memory Type	Syntax (C, D, E)	Range (cccc)	Range (dddd)	Range (eeee)
Discrete				
Discrete Inputs	I	1-128	1-128	1-128
Discrete Outputs	O	1-128	1-128	1-128
Discrete Internals	S	1-1024	1-1024	1-1024
System Discretes	SD	1-16	1-16	1-16
Bit Access Registers*	R	1-16384 / 0-15	1-16384 / 0-15	1-16384 / 0-15
Registers				
Input Registers	IR	1-64	1-64	1-64
Output Registers	OR	1-64	1-64	1-64
Registers Internals	R	1-16384	1-16384	1-16384
System Registers	SR	1-20	1-20	1-20

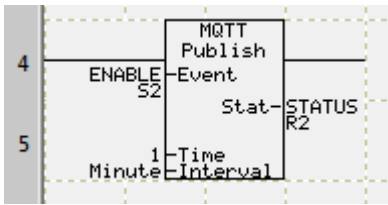
*Bit level access to registers is possible. You will be accessing bits of certain register (only 16 bit registers are allowed) for example for R1 you can access bit 1 using R1/0. See section 3.3.1 for more information.

Allowed Data Formats: Discrete and all Register data types except BCD.



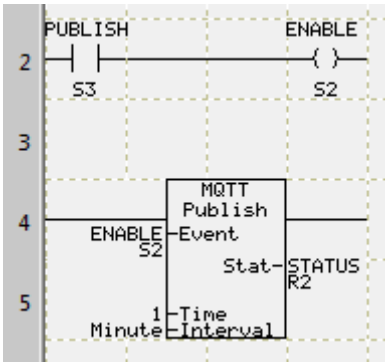
All examples use the above settings except the Publish Type is changed.

Example Time Based:



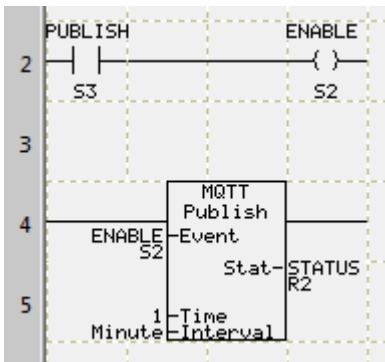
In this example, Publish Type is **At Regular Time Intervals (When Enable Tag is High)**. The instruction will publish at 1 minute intervals if the Enable tag (S2) is ON (1).

Example Event Based:



In this example, Publish Type is **On Both Edges of Event Tag**. The instruction will publish when the state of the Enable tag (S2) is changed (from ON to OFF or OFF to ON).

Example Both Event and Time Based:



In this example, Publish Type is **At Regular Time Intervals (When Enable Tag is High)**. The instruction will publish at 1 minute intervals if the Enable tag (S2) is ON (1). The S3 bit will therefore control the data log instructions.



Chapter 4: Simulating / Monitoring / Debugging PLC Logic

In this Chapter...

4.1 PLC Simulator Functions	235
4.1.1 Simulating your PLC Logic	235
4.1.2 Simulator Functions	236
4.1.3 Simulator IO View	236
4.1.4 Simulator Debugging.....	236
4.2 Online Mode	237
4.2.1 Edit Online.....	238
4.2.2 Monitor Online	240
4.2.3 Forcing I/O	244
4.3 Debugging PLC Logic	245
4.3.1 Debug Mode	245
4.3.2 Breakpoints.....	247
4.3.3 Run/Single Step.....	249

4.1 PLC Simulator Functions

EZRack PLC Designer Pro has entire new feature the PLC Simulator which allow users to simulate and debug your project without any PLC hardware. This allows for easier programming of EZRack PLC, since you can try your developed project even before you have hardware. The PLC Simulator can not only simulate all of your PLC logic but can be used simulate and force your IO outputs.

4.1.1 Simulating your PLC Logic

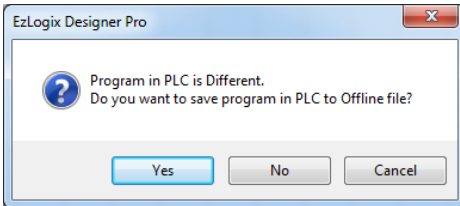
The EZRack PLC Simulator is a built in feature which behaves similarly to being online with the actual EZRack PLC. On the other hand unlike being online no external devices are needed. The simulator is installed with the EZRack PLC Designer Pro and works on any PC device which satisfies the PC requirements mentioned in *Section 1.1.1*.

To use the simulator please follow the steps below:

1. Please create a project to simulate by following the steps in chapter 1.
2. Once the project is finished save the project and then click on the simulate button.
3. The simulator will start and the EZRack PLC Designer Pro will load the project onto the simulator.



- a. The EZRack PLC Designer Pro will at all times try to preserve projects that exist



on the PLC or Simulated PLC. Therefore any time the PLC project differs even slightly from the PC project, going online will prompt you to save the PLC project. *Note: This will also happen if there is no PLC project since that empty project is different than your current project.*

- b. To preserve the PLC Project press Yes and you will be able to save the project under a different file name.
 - c. Press No to continue without saving the PLC Project.
 - d. Before going online the PC project will be transferred to the PLC or PLC Simulator and for that you need to STOP the PLC or Simulator.
 - e. Afterward the PLC or Simulator needs to be started.
4. The EZRack PLC Designer Pro will now behave as if you were online with a (simulated) PLC. This means that all functionality of Online Mode and Debug Mode is available when connected to the simulator. *Note: Some function blocks do not do anything when using simulator (e.g. Data Log, Open Port, etc.).*

Note: Please be careful with the COM configuration since Simulator is one of the options and will the COM will be set to simulator after using the Simulator.

4.1.2 Simulator Functions

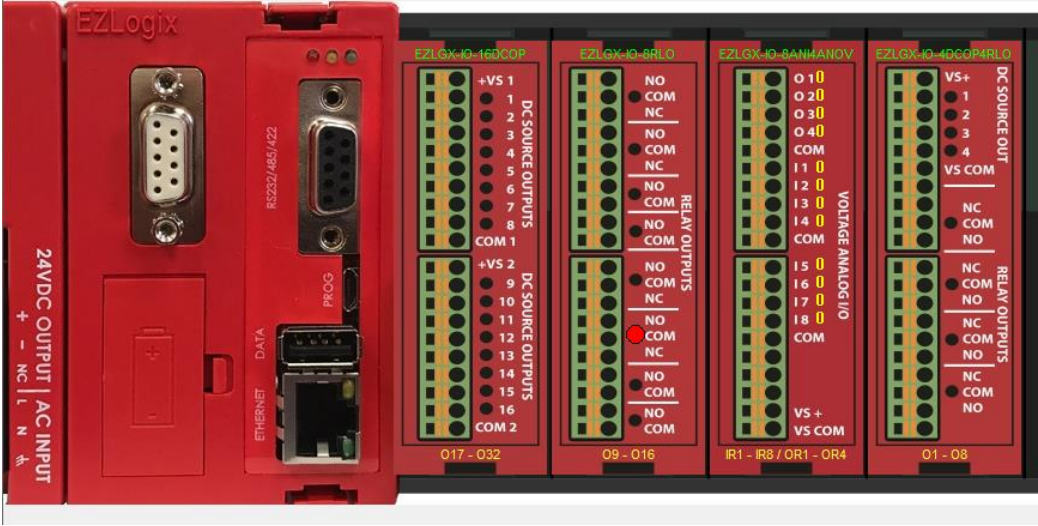
The EZRack PLC Simulator has the same functionality as does being online with the EZRack PLC. Therefore you can monitor what your PLC logic is doing, modify your logic in the simulator, turn discrettes ON/OFF, input different values into registers, and even monitor/modify your IOs. When simulating your logic please look at *Section 4.2* for more information on how to use these operations. *Note: Only 1 instance of simulator can be running at a time.*

Conncted To	Simulator
Mode	ON-LINE

Note: Simulator Mode will be indicated in the left sidebar with the current Mode (ON-LINE, MONITOR, and DEBUG) also noted.

4.1.3 Simulator IO View

The EZRack PLC Simulator can show you what is going on with your inputs. To examine your current input and output status please go to I/O Graphical Layout. Please more about the I/O Graphical Layout in *Section 4.2.3*. The I/O Graphical Layout behaves exactly the same in Monitor Mode with the Simulator as it does when working directly with a PLC.



The I/O Graphical Layout when used when in Monitor Mode will display the current outputs that are ON/OFF. For Analog outputs it will also indicate the current output value. You can also use either the Debug/Monitor Window or the right click menu to Force Inputs ON/OFF to test your logic. Please see *Section 4.2.3* for more information on this functionality

4.1.4 Simulator Debugging

The EZRack PLC Simulator allows you to use the EZRack PLC debugging software as well. Therefore you can debug your logic without any external devices. The debugger in the simulator behaves exactly as the debugger for the EZRack PLC therefore please see *Section 4.3* for more information on how to use the debugger.

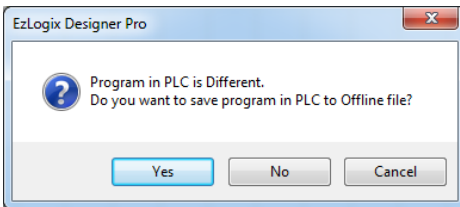
4.2 Online Mode

The EZRack PLC Designer Pro allows you to go online with the PLC at any point and time in your program development to test your current code. Once online you can monitor what your PLC logic is doing, modify your logic in the simulator, turn discrettes ON/OFF, input different values into registers, and even monitor/modify your IOs. To go online please follow the steps below:

1. Please open the project you wish to go online with in the EZLogic Designer Pro. To do this please select Open / New Project in the Project Information Screen. *Note: To go online with current project in PLC, please use the Read from PLC option and save the project to the PC before going online with the PLC.*
2. Please select the project you want to open and press Click to Start Designing.
3. Once the project opens you can either select the Com configuration. Here please select the way you would like to connect to the PLC. Possibilities include over Ethernet, over EZ-WiFi module and serially (serial includes both serial cable and micro-USB cable)



4. Next click the Online button. The project will be transferred to the PLC after the EZRack PLC Designer Pro makes sure the PLC project is not lost.
 - a. The EZRack PLC Designer Pro will at all times try to preserve projects that exist



on the PLC. Therefore any time the PLC project differs even slightly from the PC project, going online will prompt you to save the PLC project. *Note: This will also happen if there is no PLC project since that empty project is different than your current project.*

- b. To preserve the PLC Project press Yes and you will be able to save the project under a different file name.
 - c. Press No to continue without saving the PLC Project.
 - d. Before going online the PC project will be transferred to the PLC and for that you need to STOP the PLC.
 - e. Afterward the PLC needs to be started.
5. Now you are online with the PLC in Online Edit Mode.

Note: To be able to go online with the PLC the project must not have any syntax errors and be able to transfer the project to the PLC.

To exit Online Mode just press the Online button again.

4.2.1 Edit Online

Once the EZRack PLC Designer Pro is Online with the PLC it will look like below. The initial Mode after entering Online Mode is Edit Online. You can at this point make changes to the Ladder Logic and see how it works by changing to Monitor Mode. *Note: Edit Online is for minor changes only, please do not create your project Online. Some functionality is also disabled when in Edit Online like creating subroutines, changing Ethernet Setup, changing MQTT Setup, etc.*

The screenshot shows the EZRack PLC Designer Pro interface. The main window displays a ladder logic diagram for 'Rung 1' with coils and switches. The left sidebar contains a project tree with various configuration options. The top toolbar includes buttons for 'Online', 'Simulate', and 'Go Offline'. A status bar at the bottom indicates 'Connected To: COM2' and 'Mode: ON-LINE'. An 'Ethernet Setup' dialog box is open, showing configuration fields for IP Address, Subnet Mask, Gateway Address, and DNS Servers. Annotations with red boxes and arrows point to specific features: the 'Online' button, the 'Go Offline' button, the 'Switch to Monitor Mode' and 'Start Debug' buttons, the 'Current Online Status' box, the 'Ethernet Setup' dialog, and the ladder logic diagram.

Use the Online button to leave Online Mode. Also the Com button indicates current Com status.

Logic can be modified when in Online Edit Mode. Therefore have access to functions and can make changes to Ladder Logic.

Quick access options here allow for ease of switching between modes (Monitor Mode or Debug Mode). Options also exist in Menus and Toolbars.

Current Online Status
Indicates status and type of Com connection
ON-LINE → Edit Online Mode
MONITOR → Monitor Mode
DEBUG → Debug Mode

Functionality like Ethernet Setup is disabled and is View Only.

Edit Online Functionality

Below is information about some important functionality when being in Edit Online Mode.

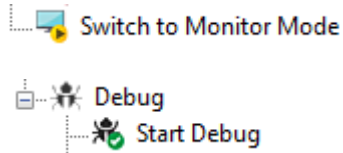
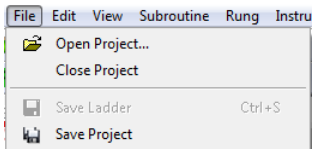
Go Offline

To go offline you can use the Online button and the Go Offline quick access option.



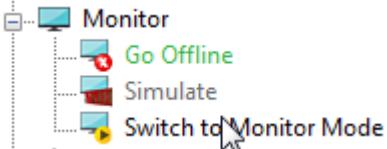
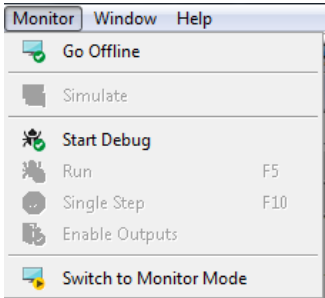
Save changes (PLC and PC Project)

To save the changes you have made to the project and transfer to PLC you can use the Save project toolbar option or menu option. You can also switch to Monitor or Debug mode.



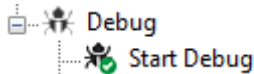
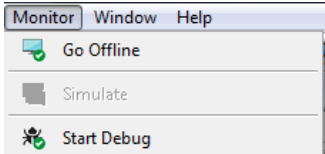
Switch to Monitor Mode

To switch modes to Monitor Mode you can use the Switch to Monitor Mode quick Access option. You can also go to **Monitor > Switch to Monitor Mode**.



Switch to Debug Mode

To switch modes to Debug Mode you can use Start Debug quick Access option. You can also go to **Monitor > Start Debug**.



View Only Options when Online:

Add Subroutines, Ethernet Setup, MQTT Setup, PID Setup, COM Configuration, I/O Configuration, Create USB File Loader File, Create OEM Package

4.2.2 Monitor Online

Once the EZRack PLC Designer Pro Mode is Switched to Monitor Mode it will look like below. Monitor Mode will shows the current PLC values of all tags. This mode also allows for changing the current state of tags and forcing I/O's. This mode is the best way to see your current behavior on the PLC. *Note: Monitor Mode can only be switched to from Edit Online Mode.*

The screenshot shows the EZRack PLC Designer Pro Monitor Mode interface. On the left is a project tree with 'I/O Graphical Layout' selected. The main area displays a ladder logic diagram for Rung 1 with various coils and switches. On the right is a 'Tags' table showing the current status of 15 tags. At the bottom left is a 'Connected To' status box. Annotations with red arrows point to various features: 'I/O Graphical Layout' in the project tree, 'Switch to Edit Mode' in the Monitor menu, the 'Tags' table, the ladder logic diagram, and the 'Connected To' status box.

Use this option to View the I/O Graphical Layout which allows viewing I/O status and forcing I/O.

Monitor Window allows for viewing the status of tags and changing status of tags. Please see more information on the next page.

Current Status of tags will be indicated both in Monitor Window and in the Ladder Logic.

Quick access options here allow for ease of switching to Edit Online Mode.

TagName	Address	Data Type	Current Value	
1	COIL 1	S1	DISCRE...	OFF
2	COIL 2	S2	DISCRE...	OFF
3	COIL 5	S5	DISCRE...	ON
4	COIL 3 (...)	S3	DISCRE...	OFF
5	COIL 6	S6	DISCRE...	OFF
6	COIL 4 (...)	S4	DISCRE...	OFF
7	SET COIL	S7	DISCRE...	OFF
8	VALUE	R1	UNSIG...	0
9	EQUAL	S11	DISCRE...	OFF
10	NOT EQ...	S12	DISCRE...	OFF
11	CREATE...	S13	DISCRE...	OFF
12	LESS TH...	S14	DISCRE...	ON
13	GREATE...	S15	DISCRE...	OFF
14	LESS EQ...	S16	DISCRE...	ON
15	LIMITS	S17	DISCRE...	ON

Conncted To	COM2
Mode	ON-LINE MONITOR
PLC Status	PLC STARTED

For Help, press F1

Zoom Level: 100%

PLC STARTED

ON-LINE - MONITOR Mode

Current Online Status
Indicates status and type of Com connection. Also informs you about the PLC Status (Started / Stopped)

Monitor Window

The Monitor Window allows the user to view current status of tags and then set the value of tags. It also allows the user to force I/O values. For ease of use it has 3 different watch modes and a full list of tags.

Local Tags

#	TagName	Address	DataType	Current Value
1	COIL 1	S1	DISCRE...	OFF
2	COIL 2	S2	DISCRE...	OFF
3	COIL 5	S5	DISCRE...	ON
4	COIL 3 (...	S3	DISCRE...	OFF
5	COIL 6	S6	DISCRE...	OFF
6	COIL 4 (...	S4	DISCRE...	OFF
7	SET COIL	S7	DISCRE...	OFF
8	VALUE	R1	UNSIG...	0
9	EQUAL	S11	DISCRE...	OFF
10	NOT EQ...	S12	DISCRE...	OFF
11	GREATE...	S13	DISCRE...	OFF
12	LESS TH...	S14	DISCRE...	ON
13	GREATE...	S15	DISCRE...	OFF
14	LESS EQ...	S16	DISCRE...	ON
15	LIMITS	S17	DISCRE...	ON

The Local Tags tab will display all the tags that can be currently seen in Ladder Logic Window. The user can use this tab to modify these values.

Set Discrete

To set discrete right click on the discrete and select to set it ON or OFF. *Note: This option will not force it ON/OFF. If ladder logic is setting the current value the discrete value will not change.*

Address	DataType	Current Value
S1	DISCRE...	OFF
S2	DISCRE...	OFF
S5	DISCRE...	ON

Set Registers

To set the value of a register double left click on the value and then modify to wanted value. *Note: This option will not force the tag. If ladder logic is setting the current value the register value will not change.*

7	SET COIL	S7	DISCRE...	OFF
8	VALUE	R1	UNSIG...	0
9	EQUAL	S11	DISCRE...	OFF

Watch Tags

#	TagName	Address	Data Type	Current Value
1	S116	S116	DISCRE...	OFF
2	VALUE	R1	UNSIG...	0
3	ADD	R2	UNSIG...	0

The Watch Tags tab has tags that are selected from the All Tags tab. They are always visible in this section no matter what ladder logic you are viewing. The watch tags tab can be also used to set these values the same way they are set in the Local Tags tab.

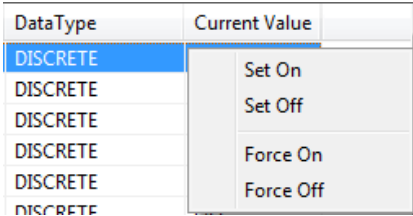
IO Tags

#	TagName	Address	Data Type	Current Value
1	M1 Output 1	O17	DISCRETE	OFF
2	M1 Output 2	O18	DISCRETE	OFF
3	M1 Output 3	O19	DISCRETE	OFF
4	M1 Output 4	O20	DISCRETE	OFF
5	M1 Output 5	O21	DISCRETE	OFF
6	M1 Output 6	O22	DISCRETE	OFF
7	M1 Output 7	O23	DISCRETE	OFF
8	M1 Output 8	O24	DISCRETE	OFF

The IO Tags tab will display all the configured IO tags. It will also show you the module the IO tag is assigned to. This tab allows the user to force IO values.

Force Discrete I/O
To force discrete right click on the

discrete I/O and select to force it ON or OFF.



Force Registers I/O

To force the value of a register I/O double left click on the value and then modify to wanted value.

25	M3 Output 1	OR1	UNSIGNED_INT_16	0
26	M3 Output 2	OR2	UNSIGNED_INT_16	0

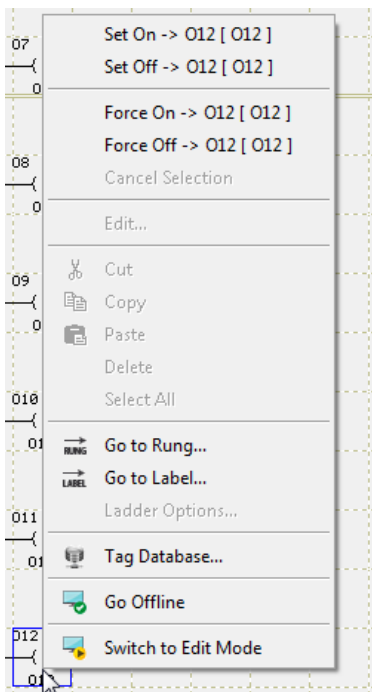
All Tags

#	TagName	Address	Data Type
16	O16	O16	DISCRETE
17	COIL 1	S1	DISCRETE
18	COIL 2	S2	DISCRETE
19	COIL 3 (NEGATIVE COIL)	S3	DISCRETE
20	COIL 4 (POSITIVE COIL)	S4	DISCRETE
21	COIL 5	S5	DISCRETE
22	COIL 6	S6	DISCRETE

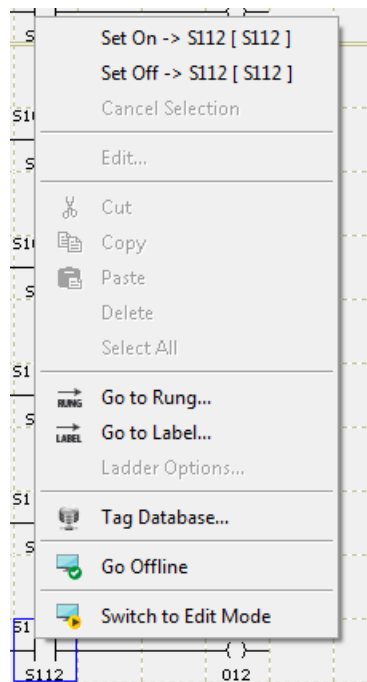
Add Selected Tag(s) to Watch List

The All Tags tab is a list of all tags in the PLC project. Use this tab to select the tags to add to the Watch Tags tab. To add a tag to the Watch List right click on the tag and press the Add Selected Tag to Watch List.

Ladder Logic Right Click Options



The Ladder Logic also has some right click menus. When selecting discretes in the ladder logic you can set them ON/OFF same as you would in the Monitor Window. For I/O discretes you can also force them ON/OFF.



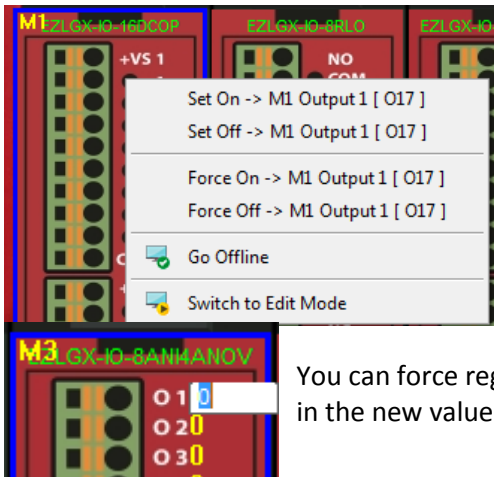
Finally the menu allows for easy of navigation and use with options like Go to Rung..., Go to Label..., Tag Database, Go Offline, and Switch to Edit Mode.

4.2.3 Forcing I/O

The EZRack PLC Designer Pro allows the user to Force I/Os. This can be done either as seen in the previous section by using the Monitor Window, or it can be done using the I/O Graphical View. The I/O Graphical View will display the current status of configure I/O and allow the user to Force I/Os.



The I/O Graphical View will indicate current status of I/O with discrete I/O being red if ON or black if OFF. Further it will indicate the I/O register values with a yellow value next to the I/O. If a value is forced it will be indicated with an F.



You can force discrete I/O values by right clicking on the discrete and select to Force it ON/OFF.

You can force register I/O by double clicking on the I/O value and typing in the new value.

4.3 Debugging PLC Logic

The EZRack PLC Designer Pro has a new feature which allows the user to debug their PLC Logic the same way you would debug code. This mode allows the user to add breakpoints at locations where they would like to stop and examine what their code is doing. It further allows them to change tag values at that point and explore results of these conditions. Please see Run/Single Step for how to use Debug Mode.

4.3.1 Debug Mode

Once the EZRack PLC Designer Pro Mode is in Debug Mode it will look like below. Debug Mode will start with PLC Stopped. The options on the next page will show how to use Debug Mode.

Note: Debug Mode can only be switched to from Edit Online Mode.

The Monitor/Debug Window indicates the current status of breakpoints and tags.

Quick access options here allow for ease of use of debug options.

After a breakpoint is added it will look like this.

The Ladder Logic will indicate the current status of tags.

Connected To	COM2
Mode	ON-LINE DEBUG
PLC Status	PLC STOPPED

Current Online Status
Indicates status and type of Com connection. Also informs you about the PLC Status (Started / Stopped).

PLC STOPPED ON-LINE - DEBUG Mode

Monitor/Debug Window

The Monitor/Debug Window allows the user to view breakpoint status and current status of tags. Further it allows the user to modify breakpoints, tag values and force I/O values. For ease of use it has 4 different watch modes and a full list of tags.

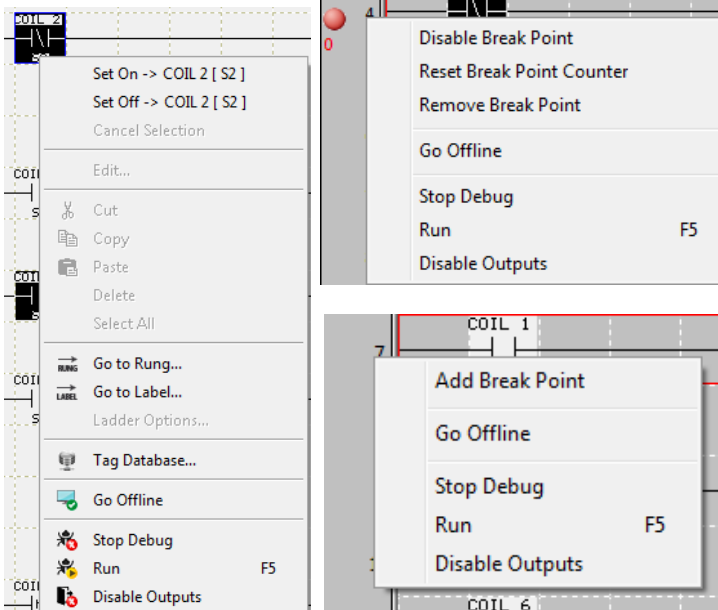
Breakpoints Tab

#	Network	Rung	Row	Column	Enabled	Hit Counter
1	1	1	4	2	Yes	0

how many times that breakpoint has been reached. Further double left clicking on breakpoint brings you to that breakpoint in the Ladder Logic. The option buttons at the top of the Breakpoints tab will be explored in the breakpoint section on the next page.

Local Tags, Watch Tags, IO Tags, and All Tags functionality is the same as in Monitor Mode. Please see *Section 4.2.2* for more information.

Ladder Logic Right Click Options



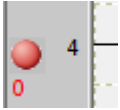
The Breakpoints tab provides information about all the current breakpoints in your project. It gives the position information about this breakpoint, plus whether the breakpoint is enabled, and finally

The right click menus in the ladder logic allow you the same functionality as in Monitor Mode. Further it adds the Debug options to the end of it.

To add breakpoints and modify breakpoints right click on the rung left bar. There you will have the option to Add Break Point. Once a breakpoint is added you have further options. Please see the breakpoint section on the next page for more information.

4.3.2 Breakpoints

In the EZRack PLC Designer Pro breakpoints are locations where the logic will stop when in Debug Mode. In Debug Mode breakpoints are added by right clicking on the Ladder Logic right sidebar.



Adding Breakpoints

Breakpoints can only be added to rows in the Ladder Logic which contain logic. Also breakpoints are added on the right side where the connection point is no matter where line goes. The number below the breakpoint is how many times this breakpoint has been reached during this debug session.

Once added the breakpoints have the following options:



Go to Breakpoint

Select a breakpoint in the Monitor/Debug Window and either double click or use the Go to View option to go to that breakpoint in the ladder logic.



Enable Breakpoint

Use this button or the right click menu to enable a breakpoint. Logic will not stop at a disabled breakpoint, but the breakpoint will still exist.



Disable Breakpoint

Use this button or the right click menu to disable a breakpoint. Logic will not stop at a disabled breakpoint, but the breakpoint will still exist.



Reset Breakpoint Counter

Use this button or the right click menu to reset the breakpoint counter. The breakpoint will count how many times this breakpoint has been reached during this debug session.



Delete Breakpoint

Use this button or the right click menu to delete a selected breakpoint.

**Enable All Breakpoints**

Use this button to enable all breakpoints. Logic will not stop at a disabled breakpoint, but the breakpoint will still exist.

**Disable All Breakpoints**

Use this button to disable all breakpoints. Logic will not stop at a disabled breakpoint, but the breakpoint will still exist.

**Reset All Breakpoint Counters**

Use this button to reset the all breakpoint counters. The breakpoint will count how many times this breakpoint has been reached during this debug session.

**Delete All Breakpoints**

Use this button to delete the all breakpoints.

4.3.3 Run/Single Step

To run debug follow the steps below:

1. After entering debug the first step is to add a breakpoint.
2. Then the user will **Run Debug**.
3. After the user runs debug the PLC will execute logic till it reaches a breakpoint.
 - a. Once the logic has reached a breakpoint the user can then single step through the logic.
 - b. The user can also press to Run Debug again to execute all the logic till reach another breakpoint.

Note: The options below are available in the Quick Access Bar on the left, in the Monitor Menu, and in the right click Menus for the Ladder Logic.

Position of the PLC in the logic

#	Network	Rung	Row	Column	Enabled	Hit Counter
1	1	4	4	2	Yes	1
2	1	1	17	2	Yes	0

When in Debug Mode the PLC location where it has stopped is indicated by a yellow arrow in the Ladder Logic right sidebar. If at a breakpoint the Monitor/Debug Window will indicate that it is at a breakpoint.

Run Debug

This will cause the PLC to execute its logic till it reaches a breakpoint. If there is no breakpoint the PLC will continually be in run mode (PLC Started).

Note: When in Debug Run Mode, the PLC cannot be stopped.

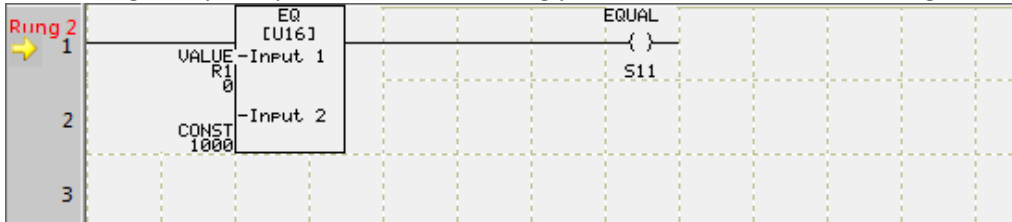
Single Step **Single Step**

This will have the PLC execute the current rung of logic and move to the next rung. No breakpoint is needed. If breakpoint exist then single step will stop at that breakpoint.

Note: Single step will move by rungs not rows. Every single step will bring you to the next rung.

Note: The yellow arrow indicating position of PLC will stop at the first row which has logic in the rung and not at the top of the rung.

After a single step the yellow arrow indicating position will be at the next rung.



Disable Outputs **Disable/Enable Output**

When debugging it might be advantageous to disable outputs. Use this option to disable or enable outputs.

Note: After leaving debug outputs will be enabled.



Chapter 5: Message Display on EZMarquee

In this Chapter...

5.1 Message Display on EZMarquee	252
5.2 Message Controller Function	253
5.2.1 Message Database	255
5.2.3 Displaying Messages	261
5.2.4 Example.....	262



5.1 Message Display on EZMarquee

EZRack PLC allows you to display text messages on EZMarquee LED displays. EZAutomation offers several marquee models, starting from single-line 10-characters, to 4-line 40-characters, for plant-wide communications. Large marquee displays are visible from a distance and get the attention of operators and management, providing them with invaluable production information and machine/process status instantaneously.

EZRack PLC makes it extremely simple to integrate an EZMarquee in a control system. The EZRack PLC has built-in features to make displaying messages on EZMarquee very easy.

The EZRack PLC has a message database where you can define all of the messages. Each message is identified by a unique message number and is displayed by telling EZRack PLC the identifying number of the message.

The EZRack PLC displays a message using one of two methods:

- Send-to-Marquee instruction
- Message controller function

The Send-to-Marquee instruction is described earlier in *Section 3.3.10* of this manual. A description of the Message Controller function appears below. Send-to-Marquee is an instruction that you use in your ladder logic. While the Send-to-Marquee instruction is more flexible (allowing you to define a register that would contain a message number, masking the message number, etc). The Message Controller function is easier to use, and for most of the applications it will be more than adequate. The rest of this chapter will focus on the Message Controller Function.

5.2 Message Controller Function

We recommend that you use either the Send-to-Marquee instruction, or the message controller feature, but not both. However, if you choose to use both methods in the same program, make sure that the messages are properly triggered and that the two methods are not fighting to send messages at the same time.

The message controller in the EZRack PLC consists of the following:
(Register/Discrete address in parentheses)

1. A Message Database
2. Message-Number System Register (SR20)
3. Message-Enable System Discrete (SD5)
4. Select-Baud-Rate System Discrete (SD6)
5. Message-Number-Not-Found System Discrete (SD7)
6. Message-Controller-Busy System Discrete (SD8)

Message Database

Message Number	Message
10	Temp Low
11	Attention!
22	Hopper Low
40	Temp High

Message Database

The Message Database holds all messages that you may want to display. Each message has a unique identifying number. To display a message, the corresponding message number is moved to message register SR20.

SR20



Message Number Register

Message Number Register

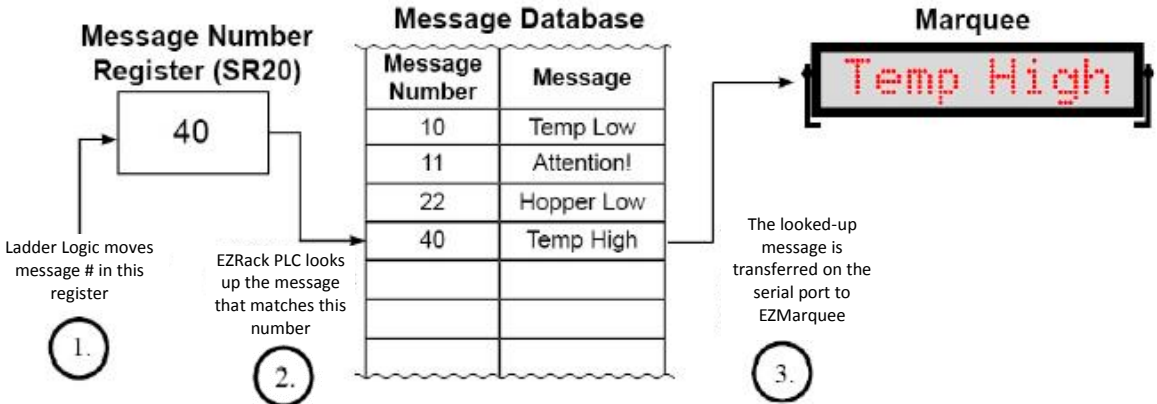
SR20 is defined as the Message Number Register. The message corresponding to the value in this register is displayed on EZMarquee, provided that SD5 is set.

System Discretes

The Message Controller function uses a few discrete system bits to manage the message display. The table below summarizes the functions of the System Discretes (SD5-SD8):

Bit	Functionality	Access	Description
SD5	Enable Bit	Read/Write	1: Enables message controller function 0: Disables message controller function
SD6	Baud Rate	Read/Write	0: (Default) 9600 Baud 1: 38400 Baud
SD7	Message Number not Found	Read Only	1: Message number in SR20 did not match any message in database (Message defined as "default" is sent) 0: Otherwise
SD8	Message Controller Busy	Read Only	1: Message controller busy processing a message 0: Message controller function is free

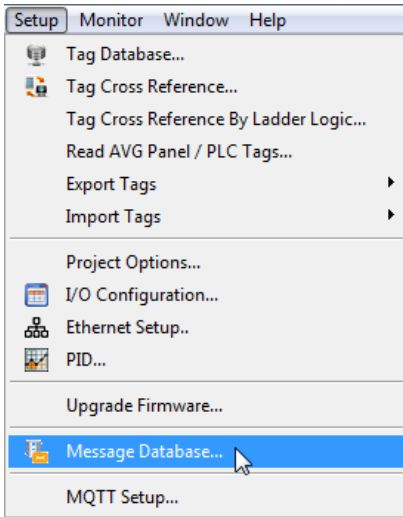
The operation of the Message Controller is shown in the diagram below:



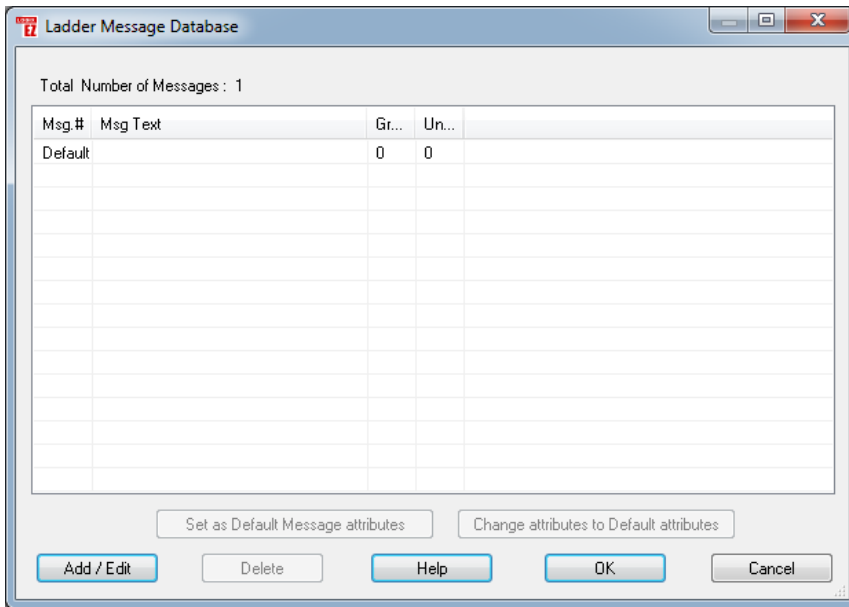
5.2.1 Message Database

As mentioned earlier, the Message Database holds all the messages to be displayed. Click onto the Setup Menu and select Message Database to define all your messages. Messages can have embedded variables which enable you to display PLC register values as part of the message.

To access the Message Database, begin by clicking onto the Setup Menu.

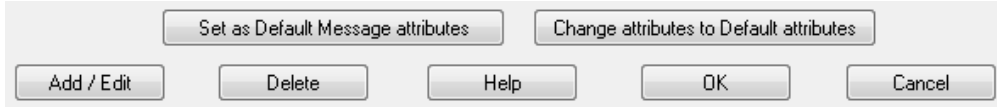


Once inside the Setup Menu, select Message Database and the following dialog box will appear.



The Message Database Lists all the programmed messages stored in the EZ Rack PLC. The default message is a message that is displayed if the value in the message-number register does not match any message number programmed. The default message is blank to start with. You may define the text of the default message.

The functions of all of the buttons in the Message Database window are as follows:



Click the **Add/Edit** button to edit the selected message or to add a new one.

Click the **Delete** button to delete the selected message.

Click the **Help** button to open context-sensitive Help.

Click the **OK** button to accept the changes and close the Message Database dialog box.

Click the **Cancel** button to cancel any changes and close the Message Database dialog box.

The **Set as Default Message attributes** button allows you to conveniently define default attributes for a message. A message has several properties or attributes as shown in the Add New Message dialog box on the previous page. To setup defaults, select the message whose attributes should be considered as default, and click this button. Once defined, all fields of the Edit dialog box will be automatically filled with the default values the next time you add a new message, saving you time. You can change the default attributes any time.

When you click the **Change attributes to Default attributes** button you can select multiple messages and change the attributes of all the selected messages to those defined as default.

Add/Edit

Message Number

Each message in the Message Database has a unique number assigned to it. The numbers need not be contiguous-- allowing you flexibility when organizing your messages. The maximum number of messages allowed in the message database is 999. The messages can be numbered from 1 to 65535. Enter a number between 1 and 65535 in this field. When you click on the Add/Edit button, the following dialog box will appear.

Marquee Address

EZMarquees can be networked using an RS422 network. The EZRack PLC can send a message to one unit, a group of units, or to all units on a marquee network.

Each EZMarquee has a DIP Switch selectable Group Number (1 or 2) and Unit Number (1 through 8). Please refer to the table below for use of Marquee address fields:

To Send Messages To	Select This	Group & Unit Number
A specific EZMarquee	Specific Unit	User-programmed group (1 or 2) and Unit Numbers (1-8). This must match with the DIP Switch setting on EZMarquee.
All units in a Group	Specific Group	User-Programmed Group Number (Unit Number = 0)
All units in Network	Broadcast	Group=0, unit=0

Display Message At Position

Center Line Column

Clear Message on line

Display Message at Position

In this group, you define where on the display the message should start. The table below describes the choices.

Select	Description
Center	Center the message on EZMarquee
Default	Don't send any positioning information with the message (Message will start at the position where last message ended)
At position	Start message at user-programmed Line and Column numbers

Clear Message check box: Check this box if you would like to clear the line before displaying the message on that line.

Select Reset Before Display Mode

Do Nothing

Select Reset Before Display Mode

You can choose certain message reset functions before displaying a new message. The choices are as follows:

Select	Description
Do Nothing	To do nothing with the previous message. The new message is appended to previously displayed message.
Clear Display, Homer Cursor	Clear the previous message and place cursor at Line 1, column 1.
Clear Display, Home Cursor, Reset	Clear the previous message and place cursor at Line 1, Column 1, and Reset EZMarquee (Reset will clear all current ASCII commands, such as Center, Blink, etc).
Clear Display, Cursor Unchanged	Clear the previous message, leave cursor unchanged.
Clear Line, Cursor at Line Start	Clear only the line and place cursor at the start of the line.

Select Message Effects

Default

Select Message Effects

This field allows you to include commands for certain message effects. The choices are described in the table below:

Select	Description
Default	No Effect
Blink Whole Message	The entire message will blink ON and OFF
Turn Off Blinking	The message will not blink
Scroll Repeatedly	The message will continuously scroll
Scroll Once	The message will scroll only once

Message Text

Char Size Color Blink

Press F7 to embed a data variable. Press CTRL+ENTER to go to next line of this message.

Message Text

In this area you Enter the actual text of the message along with its character size and color. To change the text size, you select the characters and choose the desired size from the drop down menu. To change the color, use the drop down menu to select Red, Yellow, or Green. You can also choose to

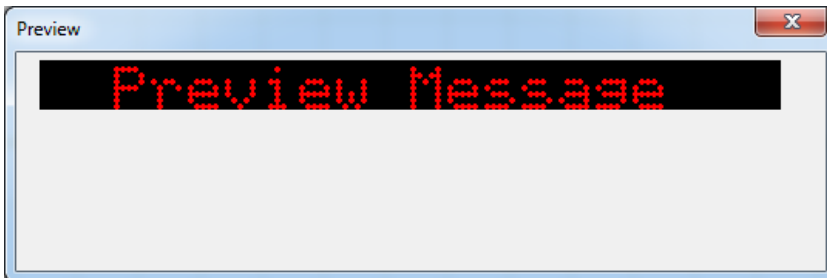
Blink selected characters of a message (to Blink the whole message, use the Blink Selected Message effect from the drop down menu).

You can embed up to 4 variables within a message. To embed a variable, press F7 at the position where you want to embed the variable and supply the information about the variable in the dialog box. You can use the key combination Ctrl+Enter to move the next line in the Edit Text box. The maximum number of characters per message is 200 (this includes any embedded attributes such as text color, text size, etc).

Preview

Preview

The Preview function allows you to see how the message will look on the marquee. Blink and scroll effects are also shown; however, these are only representations since the scroll/blink may appear differently on the actual marquee.



Finally, to add the message, click the Add New Message button. You will see the message added to the database.

5.2.2 Communication Setup

The EZRack PLC sends messages to an EZMarquee over a serial RS422 port. All communication parameters (Data Bits, Parity, and Stop Bits), except Baud Rate, are fixed between two devices.

The EZRack PLC supports two Baud Rates: 9600 (factory default), and 38,400. The EZRack PLC uses System Discrete SD6 to select between the two baud rates as follows.

Bit SD6 State	Baud Rate Selected
0	9600
1	38400

Please set SD6 to the proper value based on the Baud Rate of the EZMarquee in your initialization logic.

5.2.3 Displaying Messages



To send a message to the marquee, set the Message-Enable bit (SD5) and place the number of the message to be displayed in the Message-Number register (SR20). When you set the enable bit, the EZRack PLC will open the communication port using the Baud Rate determined by SD6 and sends the message corresponding to the message number in the SR20 register. If the message number is not found in the message database, EZRack PLC will set bit SD7 to indicate that the message number is not found and sends the “Default” message. You may monitor SD7 to detect this condition. EZRack PLC also sets the SD8 bit whenever it is busy processing a message. You should not change the Message Number register when SD8 is 1 (Busy), otherwise part of the previous message may be lost.

Once a message has been sent to the marquee, it is not sent again until one or more of the following conditions occur:

1. SD5 is disabled and enabled again.
2. The value of the register(s) embedded in the message changes.

If the embedded register value changes, the message on EZMarquee is refreshed. The table below summarizes the functions of System Discrete (SD5-SD8):

Bit	Functionality	Access	Description
SD5	Enable Bit	Read/Write	1: Enables message controller function 0: Disables message controller function
SD6	Baud Rate	Read/Write	0: (Default) 9600 Baud 1: 38400 Baud
SD7	Message Number not Found	Read Only	1: Message number in SR20 did not match any message in database (Message defined as “default” is sent) 0: Otherwise
SD8	Message Controller Busy	Read Only	1: Message controller busy processing a message 0: Message controller function is free



5.2.4 Example

Assume that the EZ Rack PLC is controlling a machine that makes parts. We need to display Production and Reject Rate, available in tags “Production Rate” (R50) and “Reject Rate” (R60) respectively, as shown in the marquee image to the left. Logic also allows for a “Machine Down” message. Machine status is available in scratch bit S100. In order to produce this example, begin by adding your messages to the Message Database. You can do so by performing the following steps:

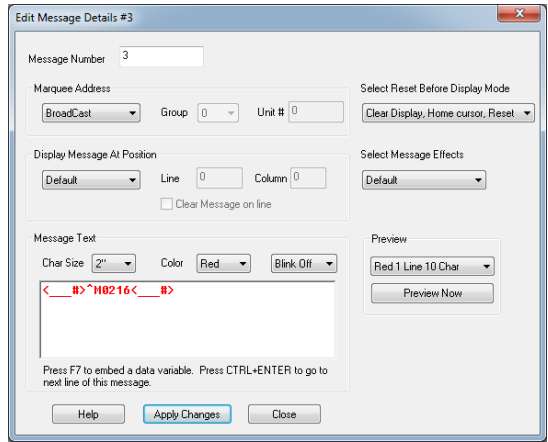
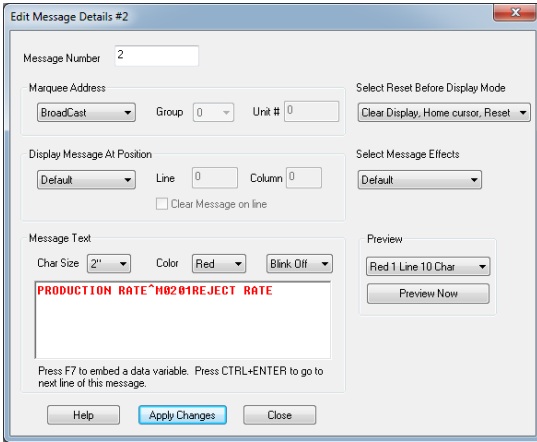
1. Click onto the Setup Menu and select Message Database.

2. Click the Add/Edit button to open the Add New Message # dialog box and set the parameters as shown in the image to the left. We created a message “Machine Down” as message number 1.

Once you’ve set the parameters as shown to the left, click onto the Add New Message/Apply changes button. The Message Database will appear as shown in the image to the below.

Total Number of Messages : 2			
Msg.#	Msg Text	Gr...	Un...
Default		0	0
1	Machine Down	0	0

3. The Production/Rejection rate message has a static part and a dynamic part. We create these as two messages. The static message is created as message #2 and is sent only once. The Dynamic part is created as message #3, and is sent repeatedly. Repeat the instructions in step 2 and set up the parameters in Messages 2 and 3 as shown in the two examples on the next page.



As shown in the dialog boxes above, use F7 to embed variable data. Variable Data appears as “<_#>” in the Message Database and in the Message Text box. You can click in the text box onto “<_#>” to edit the embedded variable. Please refer to the EZMarquee manual for message syntax and details. We’ve provided a table of Valid ASCII commands at the end of this section.

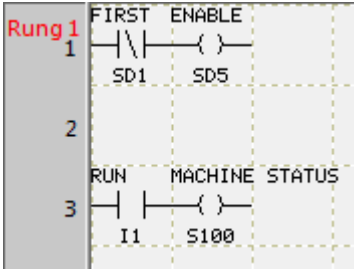
Total Number of Messages : 4

Msg.#	Msg Text	Gr...	Un...
Default		0	0
1	Machine Down	0	0
2	PRODUCTION RATE^M0201REJEC...	0	0
3	<_#>^M0216<_#>	0	0

Once you’ve completed those first three steps, the Message Database should appear as shown to the left.

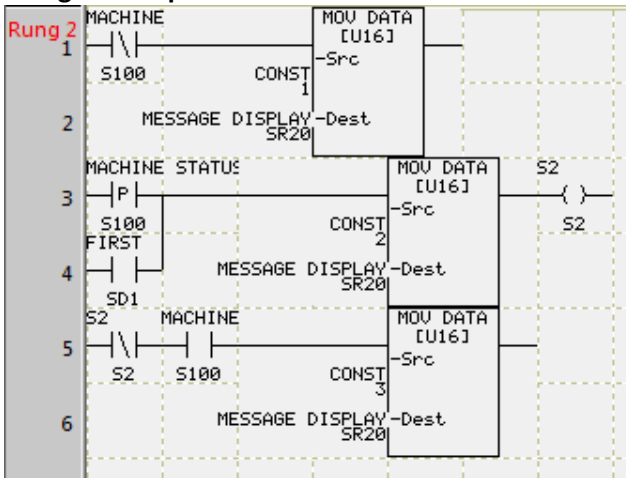
Now that you’ve set up the Message Database, you’ll need to design the ladder logic in order to complete this example project. You only need to design 3 Rungs of ladder logic. To design these Rungs, perform the following steps.

Rung 1: Enable Marquee & Check Status



The FIRST SCAN contact sets the Marquee Enable SD5. Input 1 controls the machine status.

Rung 2: Marquee Control

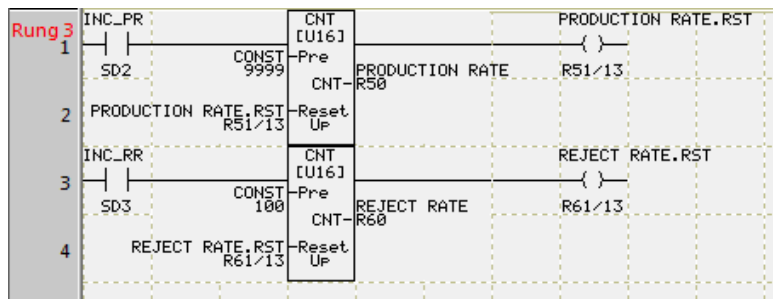


Line 1: If Machine Status (S100) is OFF, set the Marquee Message Register to SR20 to a value of 1. This will display the “Machine Down” (Message 1) on the Marquee.

Line 2: When the machine switches from stop to run, set the Marquee Message Register SR20 to a value of 2. This will display the “Production Rate” and “Reject Rate” (Message 2) on the marquee.

Line 3: Display and update the production rate value and the reject rate value on the Marquee.

Rung 3: Production & Reject Rates



This rung simulates production and reject rates.

The table below summarizes the Valid ASCII Commands in EZMarquee. For information on message syntax and details, please refer to the EZMarquee User Manual.

Valid ASCII Commands	
^Agguuuu	Selecting Unit and Group Number
^En	Resetting Display
^Hrrcc	Cursor Positioning With Line Clearing
^Mrrcc	Cursor Positioning Without Line Clearing
^ n	Cursor Positioning at Carriage Return
^Jn	Selecting Text Wrap
^Cn	Selecting Center Mode
^dCc	Selecting Character Color
^Ln	Selecting Number of Sticks per Line
^Kn	Selecting Character Size
^Bn	Selecting Blink Mode
^Xn	Selecting Blink Delimiters On/Off
^Gbbcc	Selecting Blink On/Off Rate
^Dn<message text>^N	Display Scrolling Text
All ASCII Commands listed above are Case Sensitive	



Chapter 6: PID Loop

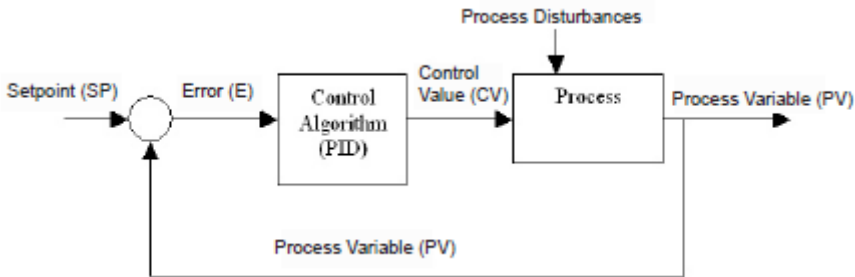
In this Chapter...

6.1 Introduction to PID	267
6.2 PID Setup.....	270
6.3 PID Monitor.....	281

6.1 Introduction to PID

Industrial Manufacturing Processes involve many variables such as temperature, pressure, flow, etc. It is important to control these variables for proper operation of the process.

There are several methods to control process variables. PID is one of the most popular control algorithms used in the industry. PID, as applied to Industrial Process Controls, stands for Proportional, Integral and Derivative control algorithm. The algorithm computes control action by using a mathematical equation which contains Proportional, Integral (Reset) and Derivative (Rate) terms. With proper choices of P, I, and D terms, a user can maintain a process value very close to the Setpoint. In addition, if the Setpoint or the process dynamics changes, the PID algorithm can bring the process back under control quickly. Selection of appropriate P, I and D coefficients is critical to the proper operation of the PID control. A block diagram of a generic process control is given below:



As shown in the figure, the user sets a target or Setpoint for the process. The system compares the actual Process Variable against the Setpoint and generates an Error value. The PID algorithm uses this error and computes a Control Variable as a function of the error. The computation function contains P, I, and D terms with user defined coefficients. The PID algorithm's goal is to minimize the error. If the Setpoint changes or the process is disturbed (resulting in a change in the Process Variable), a new error value is generated which results in a new Control Variable that should bring the Process Variable closer to the Setpoint.

PID Terminology

Before we discuss more of the details involved with the PID Loop, you should have an understanding of some of the terms used in PID.

Manufacturing Process - A process that transforms a material's properties. The transformation may involve physical or chemical changes in the material. Examples of processes are: Steam Generation, Air conditioning, Milk Pasteurization, Oil refinement, etc.

Process Variable - Materials that have physical measurable properties, such as temperature, volume, viscosity, pressure, etc. A Process variable is a measurable physical property that we want to control. For example, in the air conditioning of a building, we want to control temperature, and therefore temperature is the Process Variable.

Setpoint Value - The target or desired value of the Process Variable. The purpose of PID loop is to maintain the Process Variable as close to the Setpoint as possible.

Control Variable - The Control Variable is calculated by a control algorithm. It depends on the error and PID coefficients. (see next section for the equations used in the computations).

Error - Error equals the algebraic difference between the process variable and the setpoint. Magnitude and variation of the error depends on the process dynamics as well as on the PID coefficients. A well designed system will keep the error to a minimum value.

External Disturbance - Something that changes the equilibrium of the process. This results in a change in the control action to bring the process back into range. For example, in an air conditioned building, open doors and rainstorms are all changes that can affect the temperature.

PID on EZRack PLC

EZRack PLC products support up to 8 PID loops. For each loop the user defines several parameters (such as Setpoint, Proportional, Integral (Reset) and Derivative (Rate) Gains, Limits, etc.(further discussed in the next section). You can change most of these parameters at run time using ladder logic by using the EZRack PLC Designer Pro software in online mode.

PID Algorithms used in EZRack PLC

The EZRack PLC uses the following algorithms for PID computations:

Let

- SP_n = Setpoint at sample instance n
- PV_n = Process Variable at sample instance n
- CV_n = Control Variable at sample instance n
- K_p = Gain, Proportional term
- T_i = Reset (integral) time in seconds
- T_d = Derivative or React time, in seconds
- T_s = Sample time in seconds
- E_n = Error at sample instance n
- CV_0 = Control Variable offset

The Error is computed as follows:

$E_n = PV_n - SP_n$ for Direct Acting
 $= SP_n - PV_n$ for Reverse Acting

Then the CV_n is computed as follows:

Position Algorithm:

$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^n E_i + (T_d/T_s) \times (E_n - E_{n-1}) \right] + CV_0$$

Velocity Algorithm:

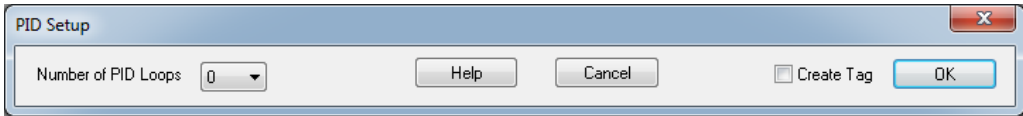
$$CV_n = K_p \times \left[E_n + (T_s/T_i) \times \sum_{i=0}^n E_i + (T_d/T_s) \times (PV_n - PV_{n-1}) \right] + CV_0$$

Note: There are options in the setup that will modify the CV computations. For example, the user can choose to use PV Square root instead of PV in error computations. Please see the PID setup where these options are discussed.

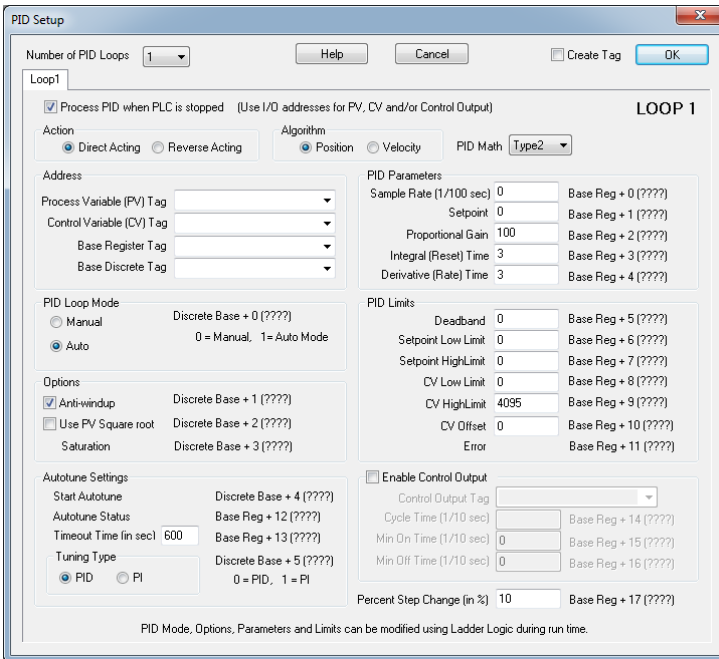
6.2 PID Setup

The following section will explain how to setup a PID loop using your EZRack PLC Designer Pro software. To access the PID Setup, perform the following steps:

1. Go to the Setup Menu and select PID. The following dialog box will appear (If you have already defined one or more loops, the image below will be different).



2. Use the drop-down arrow to select the Number of PID Loops you would like to use (you can select up to 8 PID Loops).
3. As soon as you select a number of loops other than 0, the following dialog box will appear:

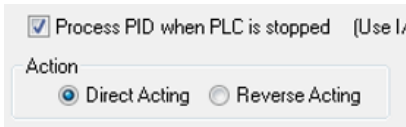


The Dialog box above allows you to define all your PID Parameters. It will show as many tabs as the number of PID loops selected. The tabs are labeled Loop1, Loop2, and so on. Each PID loop requires a contiguous block of 32 Registers and a contiguous block of 8 discrete for storing parameters and status. The blocks start at user-specified starting base addresses. In addition to the start-of-block of addresses, following tags are required: Process Variable, Control Variable and, optionally, Control Output.

The user defines the Base (or Starting) address/tag of the

Register Block. The EZRack PLC then maps the next 31 registers automatically, making a total of 32 registers per block. Out of the block of 32 registers, 17 are used currently, and the rest are reserved for future use. Similarly, the user defines a Base (or Starting) address/tag for the Discrete Block. Then the EZRack PLC will map the next 7 addresses automatically, making a total of 8 discrete in the block.

The dialog box shows which register of the block will store what for your ready reference. For example, the Sample Rate is stored in the first register of the Block (Base+0), while Deadband is in Base + 5 register. Since you know the addresses of all parameters, you can define these parameters in this dialog box, and/or dynamically define/modify these using ladder logic during runtime. The buttons and fields that appear in the PID Setup dialog box are explained below.



Process PID when PLC is Stopped - When PLC is stopped (not in Run Mode), it does NOT process ladder logic or Update I/O. However in some cases, it may be desirable to continue the PID loop even when the PLC is stopped. Use this check

box to indicate that the PID should be processed when the PLC is stopped. The default is to continue PID processing.

**Note: If you want the PID to be processed with PLC stopped, make sure to use physical addresses for the PV, and CV/Control Output tags. The reason is that when PLC is stopped no ladder executes, and ONLY PID related I/O would be updated if the “Process PID when PLC is Stopped” is checked.*

Controller Action – To determine whether the Controller Action needs set as Direct Acting or Reverse Acting, it is helpful to understand the difference between “Process Action” and “Controller Action.” First, ascertain what type of process (direct acting or reverse acting) best describes your system.

- Direct Acting Process: Control Variable and Process Variable follow the same direction i.e. Increase in Control Variable increases the Process Variable and vice versa.
 - For example: In a heating application, the more power through a heater (CV) increases the temperature (PV) or a decrease in the heater’s output will result in a decrease in temperature.
- Reverse Acting Process: The CV and PV move in opposite direction such as an increase in CV decreases the PV or vice versa.
 - For example: In an air conditioning or cooling application, more power is applied to create a reduction or decrease in the temperature.

For simple systems, after identifying the type of process that describes your system, set the Controller Action as the opposite of the process. For instance, if your system utilizes a direct acting process, the Controller Action will need set as Reverse Acting. If your system utilizes a reverse acting process, the Controller Action will need set as Direct Acting. (See below for further explanation.)

- Controller Action = Reverse Acting: The controller monitors PV. As PV decreases, the Controller will increase CV and vice versa.
 - For example: In the heating application listed above (a direct acting process), the controller would need set as Reverse Acting in order to manage the

temperature in relation to the setpoint. So that if the temperature decreases due to outside variables, the controller will increase the heater to restore the temperature back to desired level. Conversely, if the temperature increases due to outside variables, the controller will decrease the heating output to restore the temperature back to the established setpoint.

- Controller Action = Direct Acting: The controller monitors PV. As PV increases the controller will increase CV and vice versa.
 - In the example of the cooling application (or the reverse acting process), the controller would need set as Direct Acting. Therefore, if the temperature increases due to outside variables, the controller will increase the cooling power in order to restore the temperature to desired level. Conversely, if the temperature decreases due to outside variables, the controller will decrease the cooling output to restore the temperature back to the established setpoint.

EZRack PLC computes error term, based on this choice, as follows:

$$E_n = PV_n - SP_n \text{ for Direct Acting}$$

$$E_n = SP_n - PV_n \text{ for Reverse Acting}$$

Algorithm

Position Velocity

Algorithm (Position or Velocity) – EZRack PLC supports two PID algorithms, known as Position and Velocity algorithms. Select whether you would like to use a Position math equation or a Velocity math equation.

Process Variable (PV) Tag

Control Variable (CV) Tag

Process Variable (PV) Tag - Use the drop-down arrow or enter a tag address where you would like the Process Variable to be stored. You can use R or IR register types. If you use an IR type tag, then you are reading the Process Variable directly from an Input Module. If use an R-type after some scaling) using logic to the R-type register so that PID computations can use the PV.

**Note: If you would like PID to run while the PLC is stopped, you should choose an IR type tag so that the PV is updated with the actual value.*

Control Variable (CV) Tag - Use the drop-down arrow or enter a tag address where you would like the Control Variable to be stored. The CV tag has the flexibility of using R or OR registers, if you use OR, then EZRack PLC writes the CV directly to an Output Module. If you use the R type for CV tag, you will have to move the actual CV (possibly after some scaling) using ladder to an output module for control.

**Note: If you would like to PID to run when PLC is stopped, please use OR type tag for CV so that it can be updated, unless you are using Control Output.*

Base Register Tag	<input type="text"/>
Base Discrete Tag	<input type="text"/>

Base Register Tag - Base Register Tag/Address defines the starting address of a Contiguous Block of 32 registers that are used to store PID Parameters and Status information. Please see the dialog box to find the addresses of desired parameter within the block.

Base Discrete Tag - Base Discrete Tag/Address defines the starting address of a Contiguous Block of 8 discretes that are used to store PID Parameters and Status information.

PID Loop Mode	
<input type="radio"/> Manual	Discrete Base + 0 (????)
<input checked="" type="radio"/> Auto	0 = Manual, 1 = Auto Mode

PID Loop Mode - In Auto mode, the PID Loop calculates a new Control Variable value every sample period. In Manual mode, the Control Variable is controlled by user manually. The manual mode may be used for manual control of process. PID Monitor dialog box (EZ Rack PLC > PID Monitor) can be used to modify Control Variable in manual mode.

When the mode is switched from manual to auto, the integral term of the PID equation is set to the control value. This provides bumpless transfer from manual to auto.

Options	
<input checked="" type="checkbox"/> Anti-windup	Discrete Base + 1 (????)
<input type="checkbox"/> Use PV Square root	Discrete Base + 2 (????)
Saturation	Discrete Base + 3 (????)

Anti-Windup - This option inhibits integration when the control value is saturated. It controls the integral term of the PID equation when the control value is saturated. If Anti-Windup is selected, the integral term is not included when the output is saturated and the sign of the Error will cause the integral term to drive the output further into saturation. This help loops to come back into equilibrium sooner.

Use PV Square Root - If this option is selected, Square root of PV is used instead of PV in error computation.

Saturation - This line is for information only. This line shows the address of the discrete bit that would be set if the Control Variable is saturated (i.e. if the Control Variable is either 0 or 4095). You may use this in ladder logic to monitor the saturation of control variable.

Autotune Settings

Start Autotune	Discrete Base + 4 (????)
Autotune Status	Base Reg + 12 (????)
Timeout Time (in sec) <input style="width: 50px;" type="text" value="600"/>	Base Reg + 13 (????)
Tuning Type	Discrete Base + 5 (????)
<input checked="" type="radio"/> PID <input type="radio"/> PI	0 = PID, 1 = PI

Autotune Setup

The EZRack PLC can autotune PID loops, i.e. it can estimate the values for the Proportional Gain, Integral (Reset) time, and Derivative (Rate) time for PID loop. The dialog box allows you to setup the loop for autotune. EZRack PLC uses Ziegler-Nichols method to estimate the PID parameters. Following are the setup parameters for Autotune:

Note: Autotune is performed by EZRack PLC observing open loop response to a 10% step change in the control value. Before starting autotune, the process should be in a steady state. For best results, the steady state should be within 10% of the operating set point. During Autotune, watch the process variable closely for it to be within the safe limits.

Start Autotune - Shown on the dialog box for information only.

The Start Autotune discrete is at Discrete Base+4. EZRack PLC initiates autotuning of a loop when this bit transitions from 0 to 1. Autotuning of the loop is started regardless of the selected “PID Loop Mode” of the loop. Once Autotune is started, you can cancel it by setting this bit to 0. When Autotune is finished, either from success, error, or user canceled, setting this bit from 0 to 1 will return the Autotune to an idle state freeing any memory allocated for the Autotune.

Autotune Status: Shown on the dialog box for information only. During Autotune, EZRack PLC reports the status of Autotune in the register.

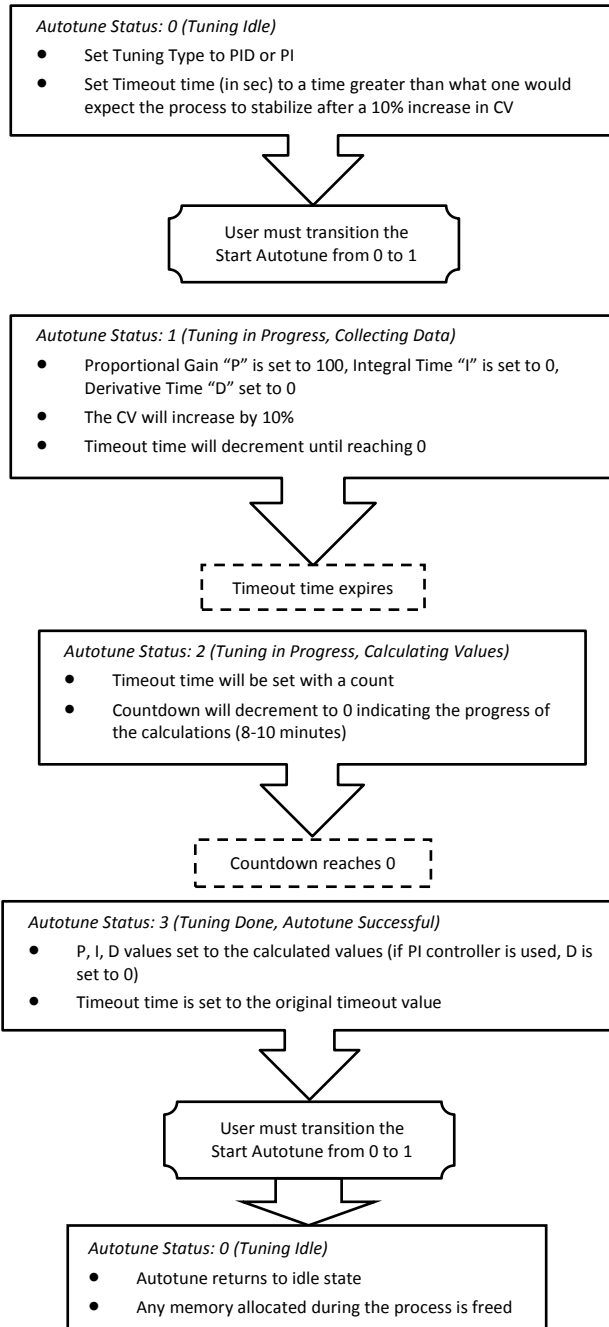
Register Value	Description
0	Tuning Idle
1	Tuning in progress, collecting data
2	Tuning in progress, calculating values
3	Tuning done, Autotune successful
4	User canceled tuning
5	Control Value could not be incremented
6	The tuning algorithm failed to fit the curve
7	Division by zero error
8	Could not determine dead time
9	One or more of P, I, or D was out of range
10	Tuning low memory error

Timeout Time (in sec): User programs Autotune timeout in seconds in this register. If EZRack PLC cannot finish autotuning within this time, the Autotune is aborted. User should program this field based on the dynamics of the process.

- When Autotune Status = 1 (collecting data), this will time down to 0 to indicate the progress of the data collection.
- While Autotune Status = 2 (calculating values), this register will be used as a decrementing counter counting down to 0 indicating the progress of the calculations

Tuning Type: User selects if PI or PID tuning is required.

Normal Autotune Sequence



PID Parameters		
Sample Rate (1/100 sec)	<input type="text" value="0"/>	Base Reg + 0 (????)
Setpoint	<input type="text" value="0"/>	Base Reg + 1 (????)
Proportional Gain	<input type="text" value="100"/>	Base Reg + 2 (????)
Integral (Reset) Time	<input type="text" value="3"/>	Base Reg + 3 (????)
Derivative (Rate) Time	<input type="text" value="3"/>	Base Reg + 4 (????)

PID Parameters

The PID Parameters consist of the Sample Rate, Setpoint, Proportional Gain, Integral (Reset) Time, Derivative (Rate) Time. The following section briefly describes each of these parameters.

Sample Rate - Enter the desired Sample Rate in this field. The Sample Rate is seconds and can be changed from 0.05 to 99.99 seconds.

**Note: All numeric fields in this dialog box use Implied Decimal points. So to enter 0.05, you simply enter 5; the EZRack PLC assumes two digits after the decimal point for most of the numeric entry fields, except where noted.*

Setpoint - Enter the Setpoint in this field. This is the Setpoint used in the PID Loop calculation. The Setpoint is the desired process level.

Proportional Gain - Enter the Proportional Gain in this field. This is the gain of the proportional term of the PID equation. The valid range is 00.00 to 99.99. Setting this to zero removes the proportional term from the PID equation.

**Note: The decimal point is implied. For example, "125" is 1.25. Default is 1.00*

Integral (Reset) Time (T_i) – The units for this time are in seconds. The Valid range is 00.00 to 6000.0. This (along K_p and T_s) controls the integral term. Setting it to zero removes the integral term from the PID equation.

**Note: The decimal point is implied. For example, "125" is 12.5 seconds. Default is 0.3.*

Derivative (Rate) Time (T_d) - Enter the Derivative Gain in this field. This along with (T_s and K_p) makes the coefficient of the derivative term. The units are in seconds. The valid range is 00.00 to 600.0. Setting this to zero removes the derivative term from the PID equation.

**Note: The decimal point is implied. For example, "125" is 12.5 seconds. Default is 0.3*

PID Limits		
Deadband	<input type="text" value="0"/>	Base Reg + 5 (????)
Setpoint Low Limit	<input type="text" value="0"/>	Base Reg + 6 (????)
Setpoint HighLimit	<input type="text" value="0"/>	Base Reg + 7 (????)
CV Low Limit	<input type="text" value="0"/>	Base Reg + 8 (????)
CV HighLimit	<input type="text" value="4095"/>	Base Reg + 9 (????)
CV Offset	<input type="text" value="0"/>	Base Reg + 10 (????)
Error		Base Reg + 11 (????)

PID Limits

The PID Limits consist of Deadband, Setpoint Low Limit, Setpoint High Limit, CV Low Limit, CV High Limit, CV Offset and Error. The following section briefly describes each of these.

Deadband - Enter the Deadband value in this field. This value is compared with the error value at loop update. If the absolute value of the error is less than the Deadband value, then the error is considered as zero for PID computations.

Setpoint Low Limit - Enter the lower limit of your desired setpoint in this field. If the setpoint is below this value, then it will be set to the value you've entered in this field.

Setpoint High Limit - Enter the higher limit of your desired setpoint in this field. If the setpoint is above this value, then it will be set to the value you've entered in this field.

Control Value (CV) Low Limit - Enter the lower limit of the Control Value in this field. If the CV is below this value, then it will be set to the value you've entered in this field. Default is 0.\

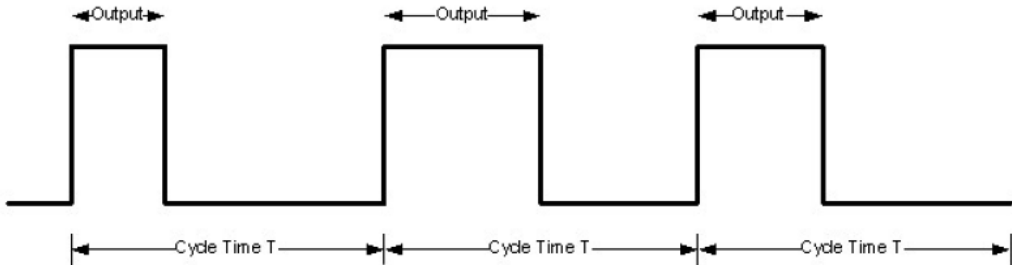
Control Value (CV) High Limit - Enter the higher limit of the Control Value in this field. If the CV is below this value, then it will be set to the value you've entered in this field. Default is 4095.

CV Offset - This is the constant offset that is added to the control variable. So, even when the Error is zero, the Control Variable equals offset.

Error – Shown on the dialog box for information only. EZRack PLC used this register to store Error value.

Control Output

EZRack PLC allows you to control a Digital output using PID control. The digital output provides a pulse out put on selected output address. The width of the pulse (within the cycle time) is proportional to the control value, as illustrated below:



Output is ON for time proportional to Control Value
 $\text{Output ON time} = (\text{CV}/4096) * T$

The following fields are programmed for the Digital Control Output: Enable Control Output: Check box to enable Digital Control Output. If the check box is unchecked, no digital output is provided.

<input checked="" type="checkbox"/> Enable Control Output		
Control Output Tag		
Cycle Time (1/10 sec)		Base Reg + 14 (????)
Min On Time (1/10 sec)	0	Base Reg + 15 (????)
Min Off Time (1/10 sec)	0	Base Reg + 16 (????)

Control Output Tag: Enter the discrete output address (O type) to provide Digital Control Output from the PID loop. The output module can be of any type (DC, AC or Relay type).

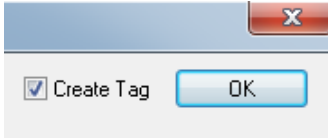
Cycle Time: Enter the Cycle time for the control output in tenths of a second. While selecting cycle time, keep in mind the load type that the output would be driving. For EM relays, we suggest that keep this time as high as possible to extend relay life.

Min Duty Cycle: This field is for display only. It is computed from the CV Low Limit $((\text{CV_LowLimit}/4096)*100)$ and expressed in percentage. As the name suggest, the output will remain on for minimum time even if the computed control value falls below the CV Low Limit.

Max Duty Cycle: This field is for display only. It is computed from the CV High Limit $((\text{CV_HighLimit}/4096)*100)$ and expressed in percentage. As the name suggest, the output will remain on for this maximum time even if the computed control value is above the CV high Limit

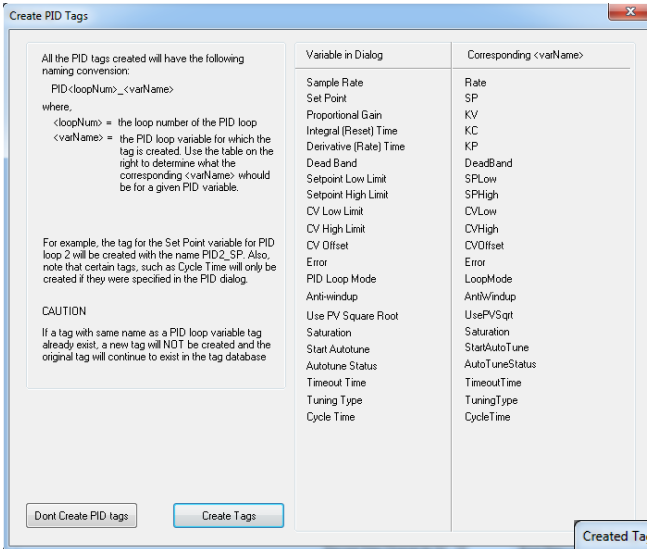
Creating PID Tags

EZRack PLC can automatically create tags corresponding to all the PID loop related variables (such as Sample Rate, SetPoint etc). To do so, perform the following steps:



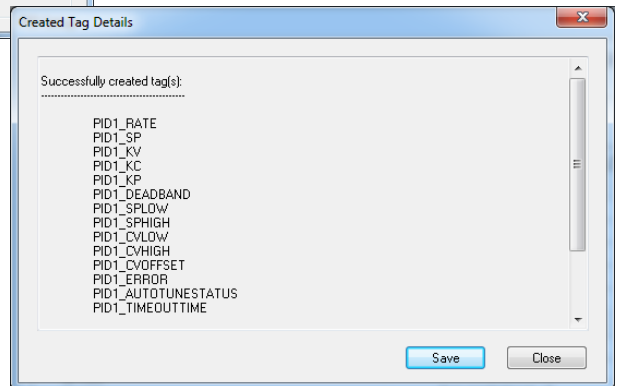
1. Check the Create Tags checkbox (located beside the OK button) in the PID dialog.

2. Click the OK button. The following dialog box appears:



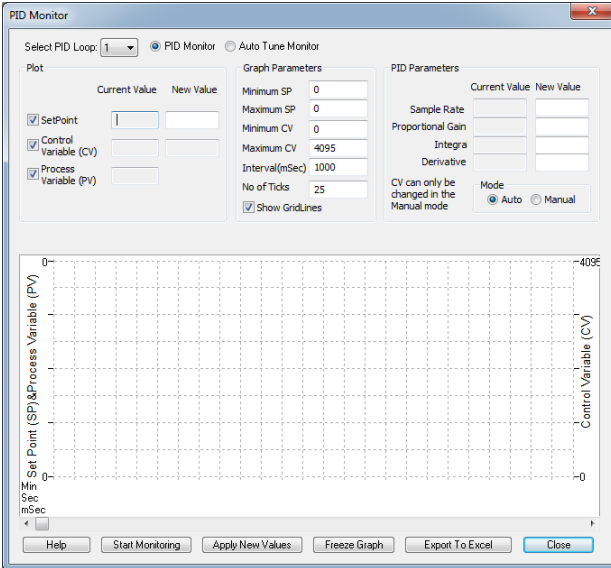
This dialog box tells you the naming convention that will be used for creating the PID loop tags. Note that all the tag names are fixed and denote the PID loop number the tag is associated with the variable the tag represents. Also, tags representing certain variables are only created if they were specified in the PID Loop (example, the tag representing Cycle Time).

3. Click the Create Tags button. At this point all the tags are created and the results are displayed in a dialog as shown below. Note that if a tag already exists, that tag will not be created and it would be reported in this dialog. By clicking the Save button, you can save this list of tags created and failed in a text file.



6.3 PID Monitor

This section will explain how to setup and use the PID Monitor function within the EZ Rack PLC Designer Pro. You can use this function to monitor and make real-time changes to your PID Loop. In order to use the PID Monitor function, you must be connected to the PLC. To begin, click to the PLC Menu and select PID Monitor (as shown to the left). The dialog box below will appear. The various fields and parameters will be explained in the following pages.



Select PID Loop - Use the drop arrow to select which PID Loop you would like to monitor (1 - 8).

Setpoint - This field displays the current value of your Setpoint. You can change the setpoint by entering a value in the New Value field and clicking the Apply New Values button at the bottom of the window.

Process Variable (PV) - This field displays the current value of the Process Variable (PV).

Control Variable (CV) - This field displays

the current value of the Control Variable (CV).

Minimum SP - Enter the Minimum Setpoint value in this field.

Maximum SP - Enter the Maximum Setpoint value in this field.

**NOTE: When selecting your values for Minimum and Maximum SP, it's a good idea to choose a number relatively close to the Process Variable. That way, when your graph is created you will be able to see more detail. The greater the range between your Minimum and Maximum SP, the less detail your graph will display. The shorter the range, the more detailed your graph will be. For this example, the Process Value is at 550, so the Maximum SP is set at 575 and the Minimum SP is set for 525, leaving a range of 50 (25 above and below) to be displayed on the graph.*

Minimum CV - Enter the Minimum Control Variable (CV) value in this field.

Maximum CV - Enter the Maximum Control Value (CV) value in this field.

Interval (mSec) - Enter the Interval value (in milliseconds) in this field.

No of Ticks - In this field, enter the Number of Ticks you would like to have displayed in the graph.

Show Grid Lines - Check this box if you would like Grid Lines to be displayed in your graph.

Sample Rate - In this field enter the Sample Rate to determine how often the PID Loop checks the process.

Proportional Gain - in this field enter the value of the Proportional Gain.

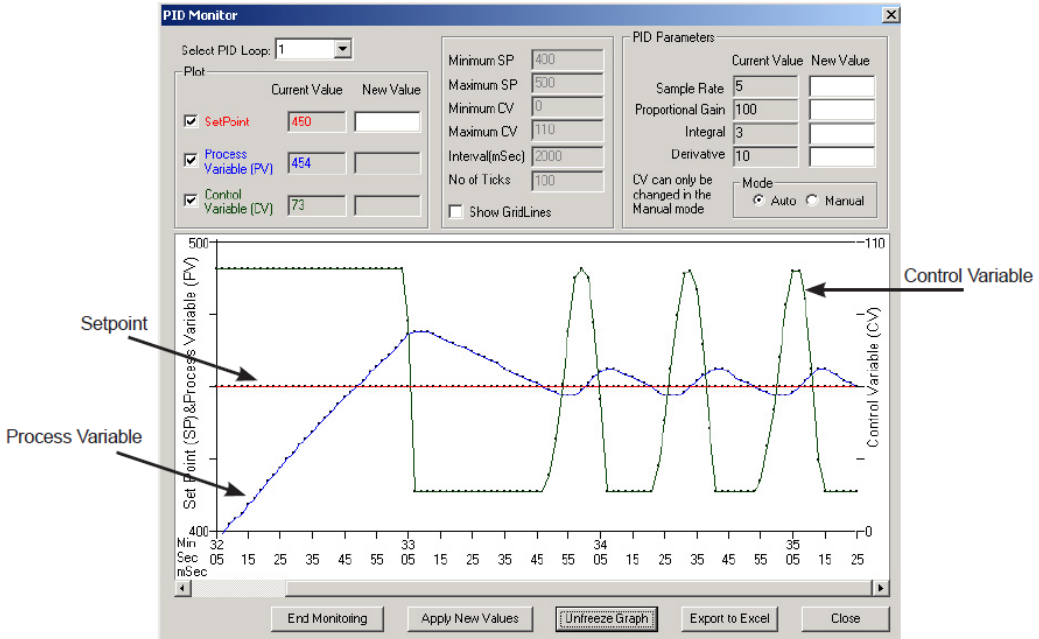
Integral - In this field enter the Integral value.

Derivative - In this field enter the Derivative value.

Mode - In this box you can choose Auto or Manual (you can only change the Control Variable in the Manual Mode).

Start Monitoring

Once all of the parameters are defined, press the Start Monitoring button (shown to the left) to begin monitoring your PID Loop. A graph will begin to appear as shown in the image below:



As you can see, the graph above has been created using the parameters explained on the previous page. The Setpoint and Process Variable were set to 450 and are represented in the graph by the line running through the middle of the graph. The Minimum SP of 400 is shown at the bottom left and the Maximum Limit of 500 is shown at the top left of the graph. The Control Variable was set to 110 and is represented on the right side of the graph. The rest of the controlling buttons for PID Monitor are explained on the next page.

Help

Start Monitoring

Apply New Values

Freeze Graph

Export To Excel

Close

End Monitoring / Start Monitoring - Press this button when you wish to stop / start the PID Monitor.

Apply New Values - Press this button once you have changed some of the parameters in PID Monitor and would like to begin monitoring those changes.

Freeze Graph - Press this button if you would like to see a still picture of the graph in its current state.

Export to Excel - Press this button to send all of the data within the graph to an Excel spreadsheet (you must have the Excel software installed onto your computer).

Close - Press this button stop the monitoring process and close the PID Monitor window.



Chapter 7: EZRack Communication (Modbus, ASCII, etc.)

In this Chapter...

- 7.1 Supported EZRack Communications 286
 - 7.1.1 EZRack Serial Communications 286
 - 7.1.2 EZRack Ethernet Communications 286
- 7.2 Modbus Communications 287
 - 7.2.1 Setup EZRack as an Ethernet Modbus Master 288
 - 7.2.2 Setup EZRack as an Ethernet Modbus Slave 293
 - 7.2.3 Setup EZRack as a Serial Modbus Master (Modbus RTU) 294
 - 7.2.4 Setup EZRack as a Serial Modbus Slave (Modbus RTU) 300
 - 7.2.5 Modbus Tips and Troubleshooting 303
- 7.3 ASCII Communication 305
 - 7.3.1 Setup EZRack to Send Out ASCII Communications 307
 - 7.3.2 Setup EZRack to Receive ASCII Communications 308

7.1 Supported EZRack Communications

The EZRack PLC supports multiple different options for communication with other devices. These communications originate from the Serial or Ethernet port. Some communications need to be setup in the EZRack Designer Pro and others are always available. Please refer to the lists below for information about setup and use of these communications.

7.1.1 EZRack Serial Communications

The EZRack currently supports the following Serial Communications:

- **AVG EZRack Protocol** – This protocol is used for HMI communication and does not need to be setup. The HMI can immediately talk to EZRack over the EZ-CBL or equivalent cable.
- **Modbus RTU Protocol** – This protocol can be used for any third party communication. The EZRack PLC does need to be setup to be a Modbus Master. No setup is needed for the EZRack to act as a Modbus Slave. To communicate over Modbus RTU a RS422 / RS485 cable is needed, please see *Section 7.2* for more information.
- **ASCII Protocol** – This is the most basic binary commutation where all information is sent over in ASCII format. This communication must be setup in the EZRack Designer Pro. For more information please see *Section 7.3* for more information.

7.1.2 EZRack Ethernet Communications

The EZRack currently supports the following Serial Communications:

- **AVG EZRack TCP/ IP Protocol** – This protocol is used for HMI communication and does not need to be setup. The HMI can immediately talk to EZRack over Ethernet.
- **Modbus TCP/IP Protocol** – This protocol can be used for any third party communication. The EZRack PLC does need to be setup to be a Modbus Master. No setup is needed for the EZRack to act as a Modbus Slave. Please see *Section 7.2* for more information.
- **IIoT/MQTT Protocol** – This protocol is the mainly used for Industrial Internet of Things communication but can be used with any device that supports MQTT protocol. Please see *Chapter 8* for more information.
- **EtherNet/IP Protocol** – This protocol can be used to communicate to any device which uses EtherNet/IP communication. Please see *Chapter 9* for more information on how to use and setup EtherNet/IP.

7.2 Modbus Communications

EZRack PLC provides connectivity to other devices over Modbus RTU and Modbus TCP/IP protocol. You can use EZRack PLC either as a Modbus Master/Client or a Modbus Slave/Server.

In this document we will use Modbus Master and Modbus Client synonymously. Similarly, Modbus Slave and Modbus Server would be used synonymously.

When used as a Modbus Master/Client, EZRack PLC communicates and exchanges data with other Modbus Slaves/Servers. When used as a Modbus Slave, the EZRack PLC can respond to Modbus commands from a Master. The EZRack can be used both as Modbus Master and Slave at the same time if using Modbus TCP/IP. For Modbus RTU only 1 connection can be made a time.

The Ethernet port on the EZRack PLC is used for Modbus TCP/IP connection. Please see section 7.2.1 and 7.2.2 for more information on how to setup the EZRack PLC for Modbus TCP/IP communication.

The Serial port on EZRack PLC is used for the Modbus RTU connection. Please see section 7.2.3 and 7.2.4 for more information on how to setup the EZRack PLC for Modbus RTU communication.

Please see the next page for Modbus Master Instruction Basics.

Modbus Master Instruction Basics

The image shows a configuration window for a Modbus Master Instruction. The window is divided into several sections with various input fields and dropdown menus. Red callout boxes with arrows point to specific fields, providing instructions on how to use them.

Callouts:

- Top Left:** Select whether using Modbus RTU or Modbus TCP/IP. (Points to the Communications dropdown menu)
- Top Right:** For Modbus TCP/IP enter the Modbus Slaves IP Address. (Points to the IP address field)
- Middle Right:** For Modbus RTU enter the Modbus Slaves ID. (Points to the Slave ID Constant field)
- Middle Left:** Select the Modbus operation this instruction will do. The options including Read or Write, Coils or Registers, One or Many. (Points to the Modbus Command dropdown menu)
- Middle Right (Lower):** For Register Communication select Byte Order. (Points to the Byte Order radio buttons)
- Middle Left (Lower):** Use the offset to select the address you will be communicating to. If you use offset 5 then the address that will be read is 300005. (Points to the Offset field)
- Bottom Left:** Enter how many consecutive registers or coils written or read from the Slave Device. (Points to the Data Length field)
- Bottom Right (Upper):** PLC Address is the starting location in the EZRack PLC where written or read information is stored. (Points to the PLC ADDRESS dropdown menu)
- Bottom Right (Lower):** Control and Error are the EZRack registers with the Modbus Master Instruction status information. (Points to the Control and Error dropdown menus)

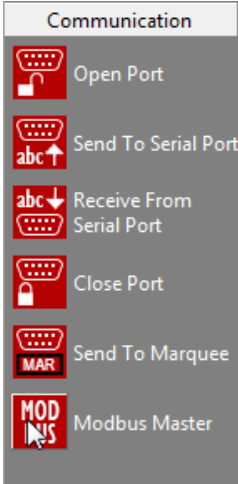
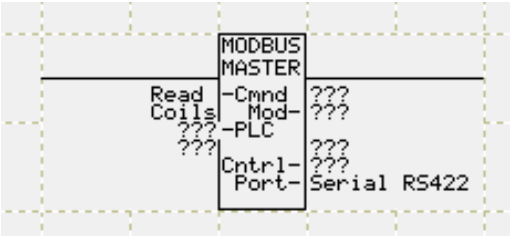
Form Fields:

- Communications:** Serial RS422 (Modbus RTU)
- IP:** 0 . 0 . 0 . 0
- Slave ID:** Tag Name (empty), Constant: 1 (1-247)
- Modbus Command:** Read Input Registers (04)
- Byte Order:** High Byte, Low Byte (selected)
- Offset:** 5 (1-65535) [Address: 300005]
- Data Length:** 1 (1-100)
- PLC ADDRESS:** (dropdown menu)
- CONTROL:** (dropdown menu)
- ERROR:** (dropdown menu)
- Timeout Time:** 30 (1-255) [tenths of a second]
- OK:** (button)

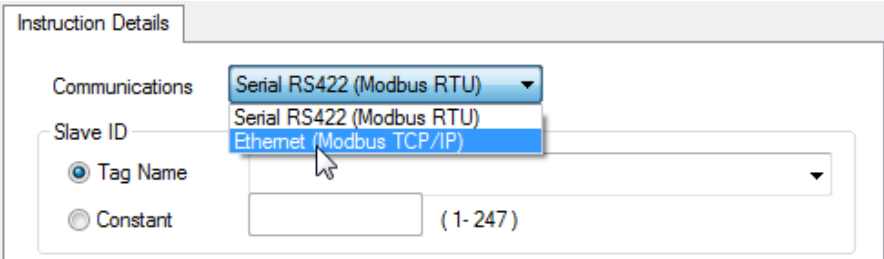
7.2.1 Setup EZRack as an Ethernet Modbus Master

The EZRack PLC can act as a Modbus Master to communicate to any third party device that supports Modbus TCP/IP communication. The EZRack currently supports up to 4 simultaneous connections at a time for read / write operations. To setup the EZRack please follow the directions below.

- 1. In an open project select the Modbus Master instruction from the Instructions Menu or the Operator Bar.

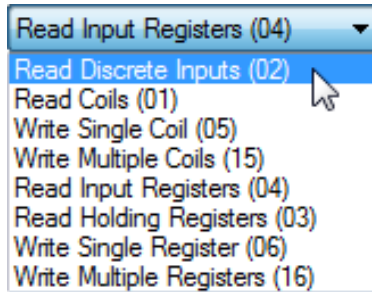


- 2. Add the instruction to your ladder logic. Then double click on it to open the setup dialogue.
- 3. In the Instruction Details use the Communications drop down and select Ethernet (Modbus TCP/IP). This will disable the slave ID option and bring up the IP address input area.



- 4. In the IP address input please put in the IP address of the Modbus Slave/Server. For example 10.1.200.100.



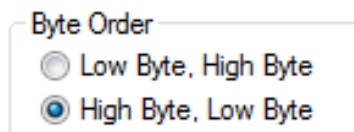


5. Next in the Modbus Command and Address Offset section select the Modbus Command to use. The table below summarizes what address range each command can write to or read from.

Modbus Command	Code	Modbus Address Range*
Read Coils	01	000001 – 065535 (Offset 1 – 65535) No more than 1024 Coils at a time
Read Discrete Inputs	02	100001 – 165535 (Offset 1 – 65535) No more than 1024 Coils at a time
Read Holding Registers	03	400001 – 465535 (Offset 1 – 65535) No more than 100 Holding Registers at a time
Read Input Register	04	300001 – 365535 (Offset 1 – 65535) No more than 100 Holding Registers at a time
Write Single Coil	05	000001 – 065535 (Offset 1 – 65535) Only one at a time
Write Single Register	06	400001 – 465535 (Offset 1 – 65535) Only one at a time
Write Multiple Coils	15	000001 – 065535 (Offset 1 – 65535) No more than 1024 Coils at a time
Write Multiple Registers	16	400001 – 465535 (Offset 1 – 65535) No more than 100 Holding Registers at a time

**(only Offset entered; type is implied by the command)*

For registers you further change the Byte Order of the data by using the Byte Order selection.



6. In the Address Offset Input the address you will communicate to.

Address Offset

Tag Name

Constant Offset (1 - 65535) [Address]

The address offset can be tag based but it always formulated as an offset based on the used command. Below are a few example:

Example 1:

To read one coil at address 200, you will select Modbus Command **Read Coils (01)**. Then in the Constant Offset input the value of 200. The Address area will now show 000200.

Example 2:

To write multiple registers at addresses 400005 – 4000020, you will select Modbus Command **Write Multiple Registers (16)**. Then in the Constant Offset input the value of 5. The Address area will now show 400005.

7. The Data Length is only available if reading or writing multiple coils/registers. The data length can be tag based or you can put in a constant value. For each Modbus Command the maximum data length will be shown to the side of the Constant Input location.

Data Length

Tag Name

Constant (1 - 1024)

8. Next the PLC Address must be input. This is the location where the value will either be written (if reading from Slave) or read from (if writing to Slave).

PLC Address

If multiple coils/registers are being written or read then the PLC Address is the starting address. For example if reading 10 registers from the Slave and the PLC Address tag address is R100 then the values will be written to R100, R101, R102... and R109. *Note: These tags will not be auto created, therefore the Auto Addressing will not ignore them and it could be possible they are used in another tag.*

9. Finally for the Modbus Master Instruction enter a register tag for the Control and Error. There is also an option to increase or decrease the time it takes before the communication will timeout. The tables below give values and descriptions for control and error codes.

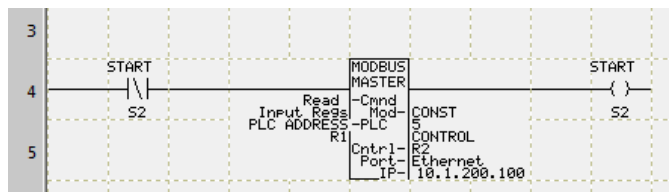
Control

Error

Control Bit Number	Status when set
B0 (LSB)	Modbus serial Enable
B1	Waiting on reply
B2	Reply processed
B3	Not used
B4	Invalid length for starting address

ERROR CODE	Error	Description
01	Illegal Function	The function code (command code) in the Modbus Master command is not understood by the Slave.
02	Illegal Data Address	The Modbus Master command tried to access an address not available in the Modbus slave device.
03	Illegal Data Value	The Modbus Master Instruction sent a value not acceptable to the slave.
04	Slave Device Failure	An error occurred in slave device, while the slave was trying to perform action requested by Modbus Master.
05	Timeout	A reply was never received from the slave (the communication link Between the Master and the Slave may be disconnected.)
07	Checksum Error	Error in check sum of the reply
08	Slave ID Failure	The slave id in the master command message does not match the slave id Returned in the reply message from the Slave.
09	Port not open error	The Port on EZRack PLC is not opened for Modbus Master Instruction

10. Now that the Modbus Master instruction is created, a contact needs to be placed in front of the instruction. The Modbus Master instruction is only executed once when power is applied. Therefore if you would like to have the instruction constantly repeat, place a normally closed contact in front of the Modbus Master Instruction and a normally open coil after.



7.2.2 Setup EZRack as an Ethernet Modbus Slave

The EZRack PLC is always configured to act as a Modbus Slave. If the EZRack gets a valid Modbus command via TCP/IP it will reply with the requested information. There is no setup needed on the EZRack PLC. Please consult the Modbus Memory Map Table below to know which tags to request for the information you want.

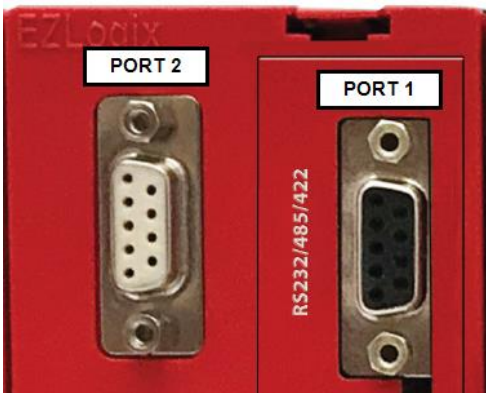
Modbus Memory Map

EZRack PLC Type	Range	Modbus Address	Modbus Type
O – Discrete Outputs	O1 – O128	00001 – 00128	DISCRETE
S – Discrete Internals	S1 – S1024	01001 – 02024	DISCRETE
SD – System Discrete	SD1 – SD16	03001 – 03016	DISCRETE
I – Discrete Inputs	I1 – I128	10001 – 10128	DISCRETE
IR – Input Registers	IR1 – IR64	300001 – 300064	WORD
R – Register Internals	R1 – R16384	400001 – 416384	WORD
OR – Output Registers	OR1 – OR64	450001 – 450064	WORD
SR – System Registers	SR1 – SR20	451001 – 451020	WORD

7.2.3 Setup EZRack as a Serial Modbus Master (Modbus RTU)

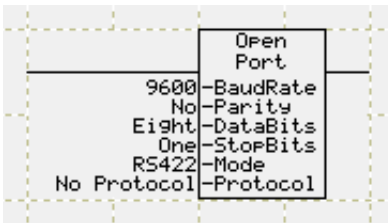
The EZRack PLC can act as a Modbus Master to communicate to any third party device that supports Modbus RTU communication. To setup the EZRack please follow the directions below.

For Modbus RTU communication you will need to use Port 1 of the EZRack PLC and you will need a RS422 or RS485 cable. Please refer to the chart below for pin out information.

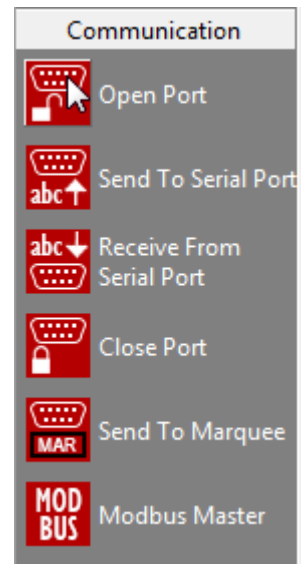


PIN CONFIGURATION	
Pin Number	Function
1	SD -
2	TXD
3	RXD
4	RD -
5	GND
6	SD +
7	CTS
8	RTS
9	RD +

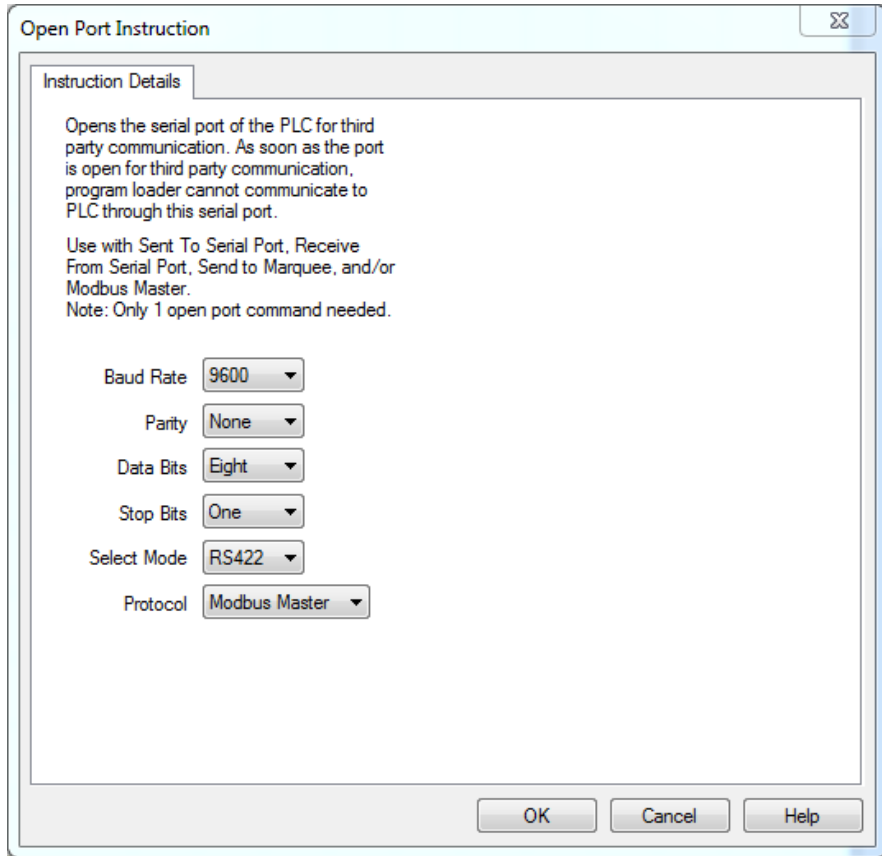
1. In an open project select the Open Port Command from the Instructions Menu or the Operator Bar.



2. Add the instruction to your ladder logic. Then double click on it to open the setup dialogue.

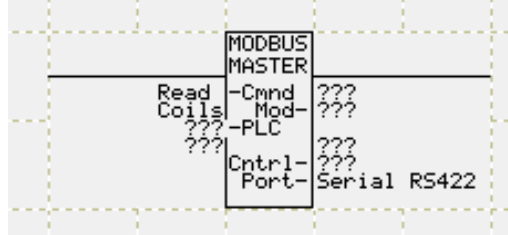


3. In the Open Port Instruction please make sure the Baud Rate, Parity, Data Bits and Stop Bits match the configuration of your other device.



4. Next please select RS422 or RS485 based on which cable you are using. *Note: RS232 does not work for Modbus Master or Modbus Slave.*
5. Finally make sure that the selected protocol is Modbus Master. ***Note: As soon as the Open Port Command is used the PLC will no longer be able to communicate over Port 1 (the primary CPU port).***
6. Next press OK and your Port 1 is now available to be used to communicate to your Modbus Slave.

- Next select the Modbus Master instruction from the Instructions Menu or the Operator Bar.



- Add the instruction to your ladder logic. Then double click on it to open the setup dialogue.
- In the Instruction Details make sure the Communications option is **Serial RS422 (Modbus RTU)**.

Communications: Serial RS422 (Modbus RTU) ▼

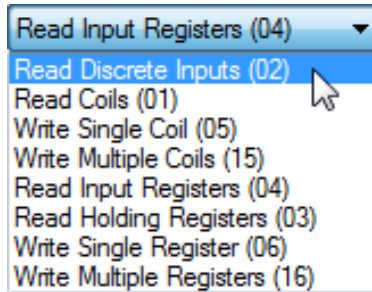
Slave ID

Tag Name

Constant

1 (1-247)

- Next for the Slave ID input the ID number of the slave you wish to communicate to. This option can also have a tag so you can change the slave you communicate to during operation. *Note: Only 1 Modbus RTU communication can happen at a time. Therefore please wait till the Modbus Communication ends before starting another one.*

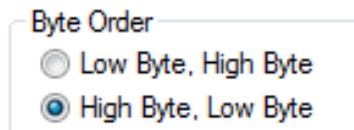


11. Next in the Modbus Command and Address Offset section select the Modbus Command to use. The table below summarizes what address range each command can write to or read from.

Modbus Command	Code	Modbus Address Range*
Read Coils	01	000001 – 065535 (Offset 1 – 65535) No more than 1024 Coils at a time
Read Discrete Inputs	02	100001 – 165535 (Offset 1 – 65535) No more than 1024 Coils at a time
Read Holding Registers	03	400001 – 465535 (Offset 1 – 65535) No more than 100 Holding Registers at a time
Read Input Register	04	300001 – 365535 (Offset 1 – 65535) No more than 100 Holding Registers at a time
Write Single Coil	05	000001 – 065535 (Offset 1 – 65535) Only one at a time
Write Single Register	06	400001 – 465535 (Offset 1 – 65535) Only one at a time
Write Multiple Coils	15	000001 – 065535 (Offset 1 – 65535) No more than 1024 Coils at a time
Write Multiple Registers	16	400001 – 465535 (Offset 1 – 65535) No more than 100 Holding Registers at a time

**(only Offset entered; type is implied by the command)*

For registers you further change the Byte Order of the data by using the Byte Order selection.



12. In the Address Offset Input the address you will communicate to.

Address Offset

Tag Name

Constant Offset [Address]

The address offset can be tag based but it always formulated as an offset based on the used command. Below are a few example:

Example 1:

To read one coil at address 200, you will select Modbus Command **Read Coils (01)**. Then in the Constant Offset input the value of 200. The Address area will now show 000200.

Example 2:

To write multiple registers at addresses 400005 – 4000020, you will select Modbus Command **Write Multiple Registers (16)**. Then in the Constant Offset input the value of 5. The Address area will now show 400005.

13. The Data Length is only available if reading or writing multiple coils/registers. The data length can be tag based or you can put in a constant value. For each Modbus Command the maximum data length will be shown to the side of the Constant Input location.

Data Length

Tag Name

Constant (1 - 1024)

14. Next the PLC Address must be input. This is the location where the value will either be written (if reading from Slave) or read from (if writing to Slave).

PLC Address

If multiple coils/registers are being written or read then the PLC Address is the starting address. For example if reading 10 registers from the Slave and the PLC Address tag address is R100 then the values will be written to R100, R101, R102... and R109. *Note: These tags will not be auto created, therefore the Auto Addressing will not ignore them and it could be possible they are used in another tag.*

15. Finally for the Modbus Master Instruction enter a register tag for the Control and Error. There is also an option to increase or decrease the time it takes before the communication will timeout. The tables below give values and descriptions for control and error codes.

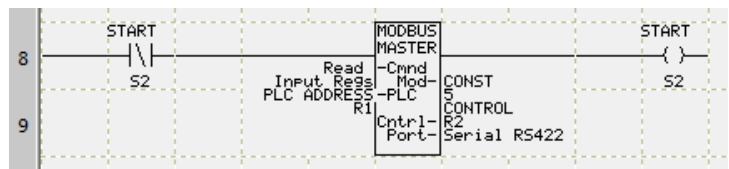
Control

Error

Control Bit Number	Status when set
B0 (LSB)	Modbus serial Enable
B1	Waiting on reply
B2	Reply processed
B3	Not used
B4	Invalid length for starting address

ERROR CODE	Error	Description
01	Illegal Function	The function code (command code) in the Modbus Master command is not understood by the Slave.
02	Illegal Data Address	The Modbus Master command tried to access an address not available in the Modbus slave device.
03	Illegal Data Value	The Modbus Master Instruction sent a value not acceptable to the slave.
04	Slave Device Failure	An error occurred in slave device, while the slave was trying to perform action requested by Modbus Master.
05	Timeout	A reply was never received from the slave (the communication link Between the Master and the Slave may be disconnected.)
07	Checksum Error	Error in check sum of the reply
08	Slave ID Failure	The slave id in the master command message does not match the slave id Returned in the reply message from the Slave.
09	Port not open error	The Port on EZRack PLC is not opened for Modbus Master Instruction

16. Now that the Modbus Master instruction is created, a contact needs to be placed in front of the instruction. The Modbus Master instruction is only executed once when power is applied. Therefore if you would like to have the instruction constantly repeat, place a normally closed contact in front of the Modbus Master Instruction and a normally open coil after.



7.2.4 Setup EZRack as a Serial Modbus Slave (Modbus RTU)

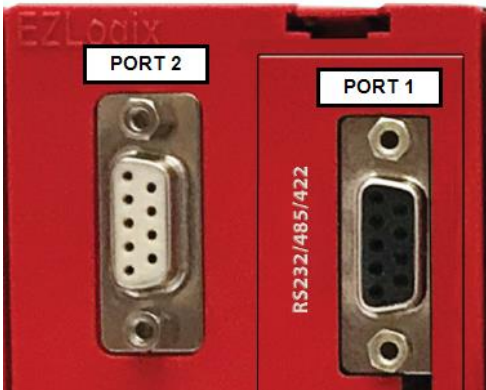
The EZRack PLC is always configured to act as a Modbus Slave but to communicate over Modbus RTU to the EZRack PLC, Serial Port 1 needs to be open and a valid Slave ID needs to be assigned to the EZRack PLC. To open Serial Port 1 follow the directions below. Once the Serial Port is open please consult the Modbus Memory Map Table below to know which tags to request for the information you want.

Modbus Memory Map

EZRack PLC Type	Range	Modbus Address	Modbus Type
O – Discrete Outputs	O1 – O128	00001 – 00128	DISCRETE
S – Discrete Internals	S1 – S1024	01001 – 02024	DISCRETE
SD – System Discrete	SD1 – SD16	03001 – 03016	DISCRETE
I – Discrete Inputs	I1 – I128	10001 – 10128	DISCRETE
IR – Input Registers	IR1 – IR64	300001 – 300064	WORD
R – Register Internals	R1 – R16384	400001 – 416384	WORD
OR – Output Registers	OR1 – OR64	450001 – 450064	WORD
SR – System Registers	SR1 – SR20	451001 – 451020	WORD

Open Serial Port 1

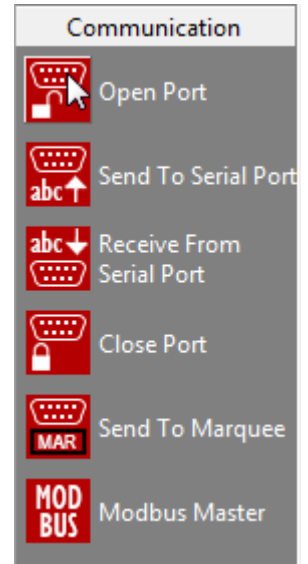
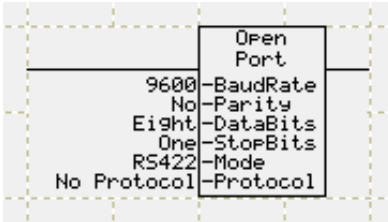
For Modbus RTU communication you will need to use Port 1 of the EZRack PLC and you will need a RS422 or RS485 cable. Please refer to the chart below for pin out information.



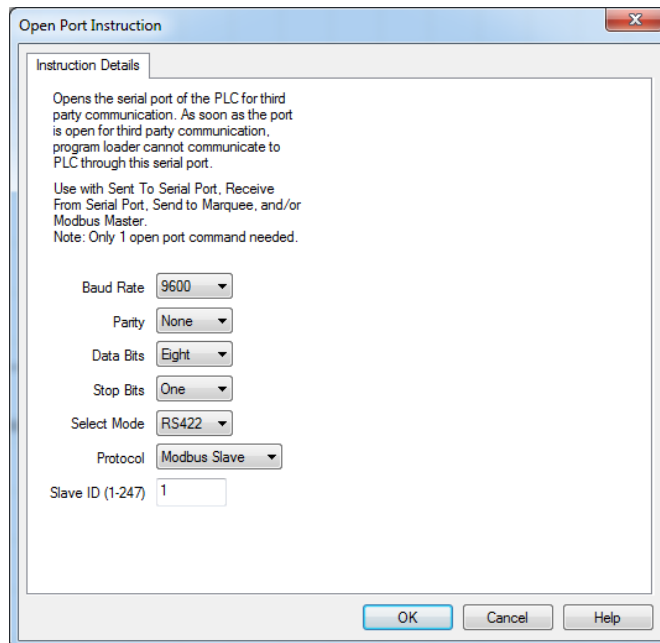
PIN CONFIGURATION	
Pin Number	Function
1	SD -
2	TXD
3	RXD
4	RD -
5	GND
6	SD +
7	CTS
8	RTS
9	RD +

Direction to Open Serial Port 1

1. In an open project select the Open Port Command from the Instructions Menu or the Operator Bar.



2. Add the instruction to your ladder logic. Then double click on it to open the setup dialogue.
3. In the Open Port Instruction please make sure the Baud Rate, Parity, Data Bits and Stop Bits match the configuration of your other device.



4. Next please select RS422 or RS485 based on which cable you are using. *Note: RS232 does not work for Modbus Master or Modbus Slave.*
5. Next select Modbus Slave for the protocol.
6. Finally enter the Slave ID that will be assigned to the EZRack PLC.

Note: As soon as the Open Port Command is used the PLC will no longer be able to communicate over Port 1 (the primary CPU port).

7. Next press OK and your Port 1 is now available to be used to communicate to your Modbus Master.

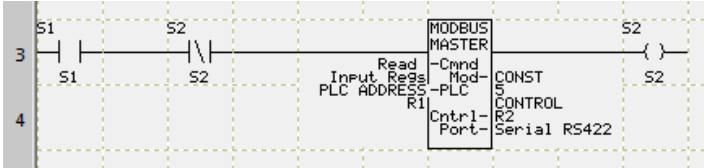
7.2.5 Modbus Tips and Troubleshooting

When using Modbus RTU or Modbus TCP/IP keep in mind that the Modbus Master instruction will only execute once upon power being applied to it. Also please refer to the table below for basics of how to use multiple instructions.

Communication Type	Max Concurrent Running Instructions	Total Max	Connection
Modbus RTU	1	Unlimited*	RS422/RS485
Modbus TCP/IP	4	Unlimited*	Ethernet

**While the number of connections total is unlimited, please make sure instructions are down before another instructions starts. Please refer to examples below.*

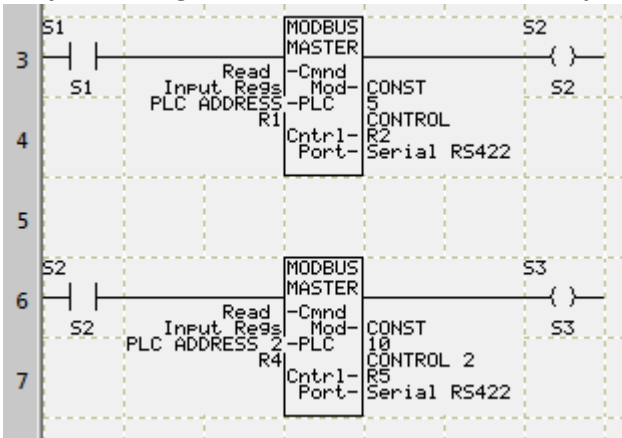
Repeating Modbus Master Instruction:



This example shows a way to repeatedly execute Modbus Master Instruction. S1 will enable the instruction. The Modbus Master instruction is repeatedly

executed as long as S1 is true. (Modbus Master Instruction executes once every time the instruction is enabled; to execute it again, the instruction should be disabled and then enabled.)

Only 1 Running Modbus Master at 1 Time Example:

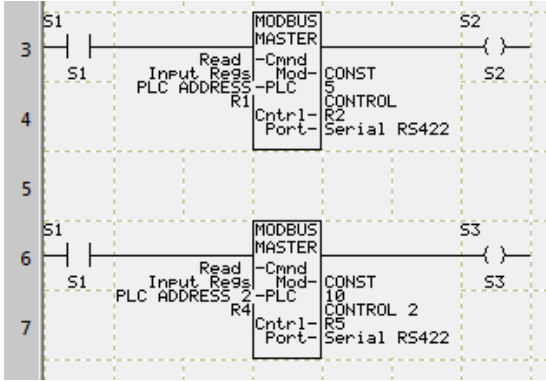


This example shows two Modbus instructions. When Normally open contact S1 is true, first instruction gets enabled, and communication to addressed slave starts. S2 will become true when S1 is true AND the Modbus instruction completes its operation.

By placing S2 before second Modbus instruction we ensure that the second instruction does not start until the first is completed. In this example S1 should

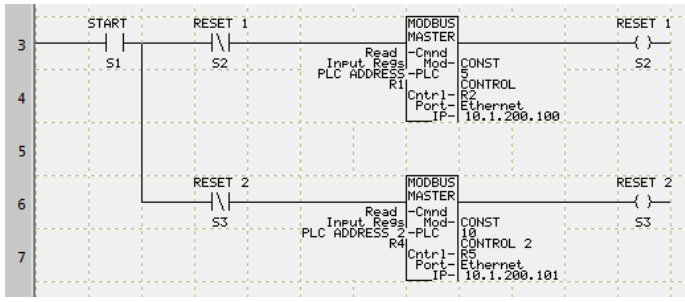
remain on until the second instruction is complete. Otherwise, when S1 turns off, S2 will also turn off, and consequently, second Modbus master instruction may not complete.

Only 1 Running Modbus Master at 1 Time Example (INCORRECT):



This example shows the **INCORRECT** use of the Modbus instructions. Two instructions are enabled simultaneously, resulting in unpredictable behavior for Modbus RTU. This example will function correctly for Modbus TCP/IP.

Constant Modbus TCP/IP Communication Example:



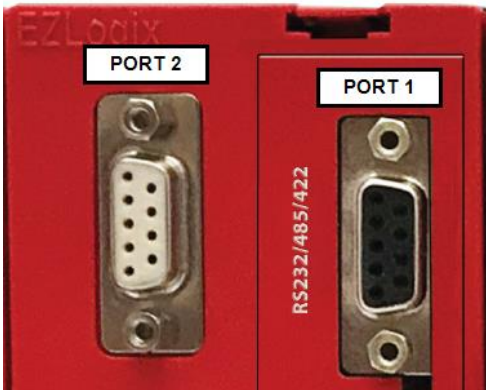
This examples shows how to constantly update Modbus information from multiple slaves when using Modbus TCP/IP. Here both Modbus Master instructions will run once S1 is ON and they will update as fast as they can. *Note: If used for Modbus RTU this will result in unpredictable behavior.*

Troubleshooting Tips

Com Type	EZRack is?	Problem	Solution
RTU & TCP/IP	Master	Illegal Function	Please make sure the Slave supports this function.
RTU & TCP/IP	Master	Illegal Data Address	Please make sure the needed address exist.
RTU & TCP/IP	Master	Illegal Data Value	Please make sure the read value is supported by EZRack PLC
RTU & TCP/IP	Master	Slave Device Failure	Please make sure the Slave is functioning correctly.
RTU	Master or Slave	Timeout	Please make sure the open port setting match those of the other device.
TCP/IP	Master or Slave	Timeout	Please make sure that the correct IP address are set and both devices are connected to Ethernet Hub (or you can use a crossover cable).
RTU	Master or Slave	Checksum Error	Please make sure the open port setting match those of the other device.
RTU	Master or Slave	Slave ID Failure	Please make sure the open port settings match those of the other device. And the correct Slave ID is used for this connection.
RTU	Master	Port not open error	Please make sure you have used an Open Port Instruction before the Modbus Master Instruction.

7.3 ASCII Communication

EZRack PLC provides connectivity to other devices over ASCII protocol. You can send and receive information from the serial port over RS232, RS422 and RS485. To use the Send and Receive from the serial port you need to first open the port using the Open Port Command. ASCII communication uses Serial Port 1 with a RS232, RS422 or RS485 cable, please refer to the table below for pin out information.

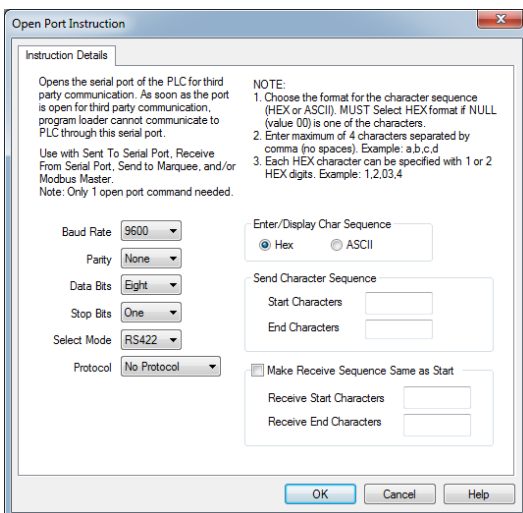


PIN CONFIGURATION	
Pin Number	Function
1	SD -
2	TXD
3	RXD
4	RD -
5	GND
6	SD +
7	CTS
8	RTS
9	RD +

Open Port Command

Open Port command is described in *section 3.3.10* of this manual. Here we repeat this briefly.

Below is the Open Port Instruction dialog box.



The following attributes will need to be set in this dialog box for the Modbus Network you are connecting to.

1. Baud Rate
2. Parity
3. Data bits
4. Stop bits
5. Select Mode "RS232, RS422 or RS485"
6. For Protocol Select "None"

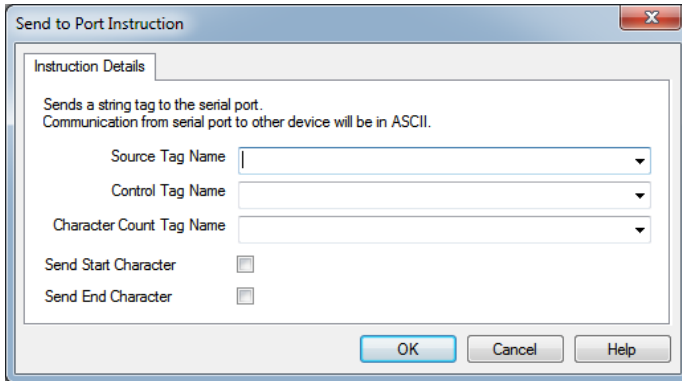
Note: As soon as the Open Port Command is used the PLC will no longer be able to communicate over Port 1 (the primary CPU port).

Enter Optional Parameters:

1. Select how the Char Sequence is inputted (Hex or ASCII).
2. Enter Send Start Characters in the Start Characters field (up to 4 characters).
3. Enter Send End Characters in the End Characters field (up to 4 characters).
4. Enter Receive Start Characters in the Start Characters field (up to 4 characters).
5. Enter Receive End Characters in the End Characters field (up to 4 characters).

Adding Send To and Receive From Port Instructions

To add the Send to Port and Receive From Port instructions, perform the following steps:



1. Select or Add an ASCII tag that contains the string to be sent in the Source Tag field using the drop down list (for a Receive instruction: the String that will receive the characters from the serial port in the Destination Tag field).

2. Select an integer register used by the instruction for status in the Control Register Tag field using the drop down list. The following table describes the control bits in the register:

Bit Number	Function
Bit 0 (lsb)	Enable (0 = Disabled, 1 = Port is Open AND Instruction is Enabled (Power flows to instruction))
Bit 1	Serial transmission done (1= function (transmit or receive) done, 0=not done)

Other bits of the register are used for internal purposes and change state during transmission/receiving.

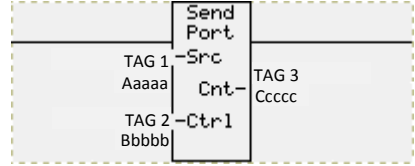
3. Select or Add an integer register that displays the number of characters transferred from the source tag to the serial output buffer in the Character Count Tag field using the drop down list (for a Receive instruction: the Number of characters transferred from the serial port to the destination tag).
4. Check either Send Start Character or Send End Character box if needed.

7.3.1 Setup EZRack to Send Out ASCII Communications

To send ASCII information out you need to use the Send to Serial Port instruction.

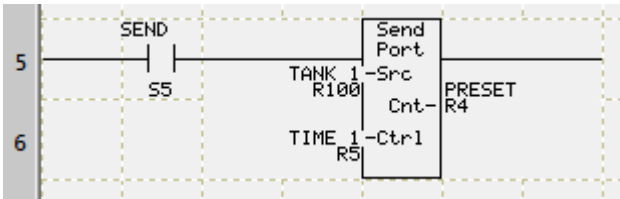
Send to Serial Port:

When power flows through this element, the Send to Serial Port instruction will send an ASCII string present in Src at memory location Aaaaa to the Serial Port. The control and character count used for sending the ASCII string is specified by Cnt at memory location Ccccc and Ctrl at memory location Bbbbb, respectively.



This instruction can only send out the specified ASCII string if the corresponding serial port has been already opened by the Open Port instruction in advance. If the serial port has not been initiated, the Send to Serial Port instruction will not send the ASCII string to the specified port.

Start and End characters can also be sent along with the ASCII string being sent out from the Src register. You can specify Start and/or End characters to be included along with the ASCII string. The starting and ending characters are specified in the Open Serial Port Instruction.



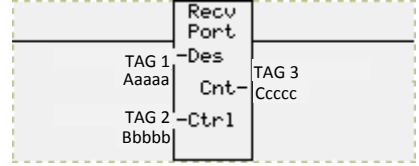
In the example above, if S5 is ON (and the Port is Open), the Send Port command would send the ASCII string as per programmed parameters. If the port is not yet open, the instruction will do nothing, and the Enable Bit in the control register will remain 0, even if the S5 is on.

7.3.2 Setup EZRack to Receive ASCII Communications

To receive ASCII information you need to use the Receive from Serial Port instruction.

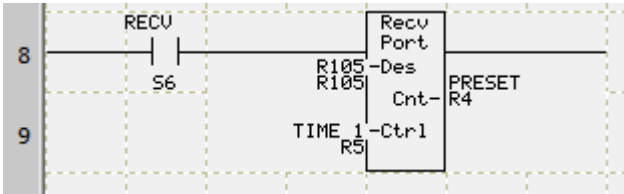
Receive From Serial Port:

When power flows through this instruction, the Receive From Serial Port instruction will receive an ASCII string from the serial port and store it in Dest at memory location Aaaaa. The control and character count used for receiving the ASCII string is specified by Cnt at memory location Ccccc and Ctrl at memory location Bbbbb, respectively.



This instruction can only receive the specified ASCII string if the corresponding serial port has been already opened by the Open Port instruction in advance. If serial port has not been initiated, the Receive from Serial Port instruction will not receive the ASCII string.

Start and End characters can also be received along with the ASCII string being received. You can specify Start and or End characters to be verified when received along with the ASCII string. The starting and ending characters are specified in the Open Serial Port Instruction.



In the example above, if S6 is ON (and the Port is Open), the Send Port command would receive the ASCII string as per programmed parameters. If the port is not yet open, the instruction will do nothing, and the Enable Bit in the control register will remain 0, even if the S6 is on.



Chapter 8: IIoT (Industrial Internet of Things)

In this Chapter...

8.1 IIOT (Industrial Internet of Things)	310
8.1.1 MQTT	311
8.2 MQTT Essentials.....	313
8.2.1 Basic Concepts	313
8.2.2 MQTT More Details and Examples	315
8.3 Basic MQTT Setup on EZRack PLC.....	324
8.4 Broker Setup	326
8.5 EZRack PLC IIoT (MQTT) Example	328
8.6 MQTT HIVEMQ Essentials	332
8.7 EZ-IIoT Subscriber Utility.....	337
8.7.1 Install EZ-IIoT Subscriber Utility	337
8.7.2 EZ-IIoT Subscriber Utility Setup	338
8.7.3 EZ-IIoT Subscriber Utility Functions	340
8.7.4 EZ-IIoT Subscriber Utility Best Practices	349

8.1 IIOT (Industrial Internet of Things)

The EZRack PLC supports the Industrial Internet of Things. The EZRack PLC comes with a built in instruction that allows the user to publish data to secure offsite MQTT Cloud Broker. This capability allows the EZRack PLC to provide data for analysis to improve efficiency, troubleshoot problems, and do preventative maintenance. This section explores in more depth what IIoT (Industrial Internet of Things) means, how the EZRack PLC supports IIoT through the MQTT protocol, and finally looks at how to setup the EZRack PLC to do MQTT communication.

What is IIoT (Industrial Internet of Things)?

The Industrial Internet of Things (IIoT) focuses on the interconnectivity and utilization of powerful data in a manufacturing environment. IIoT enables the acquisition and accessibility of important plant data at far greater speeds, security and reliability. IIoT incorporates machine learning and big data technology, harnessing the sensor data, machine-to-machine communication and automation technologies that have existed in industrial settings for years. The driving philosophy behind the IIoT is that smart machines are better than humans at accurately, consistently capturing and communicating data.

EZRack PLC built in IIoT and MQTT protocol support acts as a "bridge" between existing operational technology within a plant, for example factory machines, and plant database networks, so valuable data can be shared reliably and securely to improve plant productivity and efficiency.

How EZRack PLC Support "Edge-Gateway" Communications and IIoT?

The EZRack PLC operates as an "Edge-of-Network" or "Edge-Gateway" device with direct connectivity to external devices such as sensors, RTDs, analog inputs, etc. and easy to setup secure communications with other networks such as Modbus TCP/IP. Through the use of the MQTT protocol it can publish up to 80 tags of data per EZRack PLC CPU, thus providing a subscriber pertinent real time data from these external devices. The use of the MQTT protocol allows for great interoperability since it is becoming an industry standard. It also allows for great security through the broker. It must also be noted that with the EZRack PLC, a "security breach" to access the machine is not of any concern since there is no backwards flow of data. That is data is only ever published from the PLC. It will never accept any data or commands back from any server, broker or client.

8.1.1 MQTT

What is MQTT?

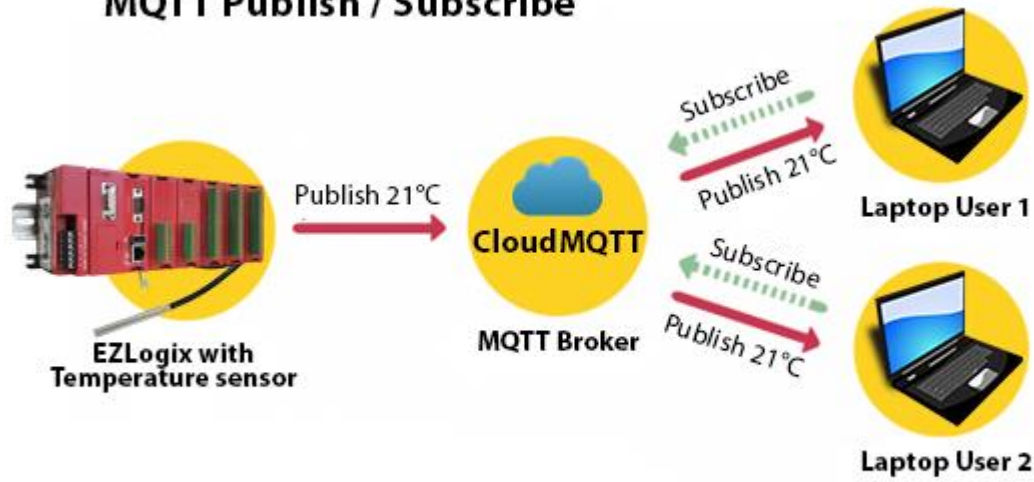
MQTT which stands for message queuing telemetry transport, is a standard Client Server publish/subscribe messaging transport protocol that is quickly becoming the leading messaging protocol for the Industrial Internet of Things (IIoT).

How does MQTT work?

The MQTT protocol works on a publish/subscribe (pub/sub) pattern. This is different from a traditional client-server model in that the machine (PLC) does not directly communicate to the server. The Pub/Sub pattern decouples a client that is publishing data (sending messages) from the client that is subscribing to the data (receiving messages). For this pattern the sender of messages is called the publisher and the receiver of messages is called the subscriber.

This Pub/Sub pattern essentially creates a barrier between the publisher and subscriber in that they do not know about the existence of the other. The broker who is known by both is the link between them. The broker can filter all the messages and distribute them to the subscriber that is supposed to receive them. Multiple subscribers can be receiving messages from the broker at the same time but getting different data. This allows for separating out access so only pertinent data is received to selected individuals or “subscribers”. The graphic below shows how Pub/Sub works.

MQTT Publish / Subscribe



1 Subscribe to topic: "temperature"

2 Publish to topic: "temperature"

What is the current functionality of EZRack PLC?

The EZRack PLC currently works as a publisher of data in the MQTT pattern. It is very flexible in that it works with any broker that the customer would like. The EZRack PLC connects to the broker with a username and password for security and then can publish up to 80 tags, also known as topics, at settable intervals.

Why does EZRack PLC as an Edge-Gateway device use MQTT for its communication?

The MQTT protocol is becoming the industry standard communication protocol for IIoT. More importantly, the MQTT protocol, provides a “bridge” between existing operational technology within a plant, for example factory machines, and plant database networks so valuable data can be shared reliably and securely to improve plant productivity and efficiency.

Misconceptions of IIoT and MQTT.

1. Implementation is extremely costly.

The EZRack PLC, with base rack, CPU and power supply all included is at an extremely attractive price of \$248 and has IIoT MQTT protocol built in, among many other features including data-logging, ladder logic and function blocks, auto-tuned PID and much more...

2. Better to wait for an Industry Consensus.

MQTT is becoming the industry standard therefore a consensus around it is developing in the manufacturing and process sectors. But even if that wasn't true, MQTT is a light and versatile protocol which can allow for communication with many different machines and plant devices.

3. Is implementing IIoT really worth it?

IIoT connectivity allows “management” to see plant performance thereby allowing the plant to optimize and track their production and efficiency. Furthermore, the implementation of IIoT can help with offsite troubleshooting and offsite analysis of production data.

4. Adding IIoT will be complicated.

The EZRack PLC has IIoT MQTT protocol built in and therefore it is a very easy setup process. Also if in the future you wish to add IIoT to an existing system no major changes are needed. Please see the MQTT Essentials section to understand how simple IIoT MQTT protocol is.

8.2 MQTT Essentials

This section explores the basics of MQTT and how it functions. For easy setup guide please see *Section 8.3, 8.4 and 8.5*.

8.2.1 Basic Concepts

Referenced from <http://mqtt.org/> and <http://mosquitto.org/man/mqtt-7.html>

Publish/Subscribe

The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. Many clients may subscribe to the same topics and do with the information as they please. The broker and MQTT act as a simple, common interface for everything to connect to. This means that if you have clients that dump subscribed messages to a database, for example Twitter, or even a simple text file, then it becomes very simple to add new sensors or other data input to a database, Twitter or so on.

Topics/Subscriptions

Messages in MQTT are published on topics. There is no need to configure a topic, publishing on it is enough. Topics are treated as a hierarchy, using a slash (/) as a separator. This allows sensible arrangement of common themes to be created, much in the same way as a filesystem. For example, multiple computers may all publish their hard drive temperature information on the following topic, with their own computer and hard drive name being replaced as appropriate:

```
sensors/COMPUTER_NAME/temperature/HARDDRIVE_NAME
```

Clients can receive messages by creating subscriptions. A subscription may be to an explicit topic, in which case only messages to that topic will be received, or it may include wildcards.

Quality of Service

MQTT defines three levels of Quality of Service (QoS). The QoS defines how hard the broker/client will try to ensure that a message is received. Messages may be sent at any QoS level, and clients may attempt to subscribe to topics at any QoS level.

Higher levels of QoS are more reliable, but involve higher latency and have higher bandwidth requirements.

- 0: The broker/client will deliver the message once, with no confirmation.
- 1: The broker/client will deliver the message at least once, with confirmation required.
- 2: The broker/client will deliver the message exactly once by using a four step handshake.

Retained Messages

All messages may be set to be retained. This means that the broker will keep the message even after sending it to all current subscribers. If a new subscription is made that matches the topic of the retained message, then the message will be sent to the client. This is useful as a "last known good" mechanism. If a topic is only updated infrequently, then without a retained message, a newly subscribed client may have to wait a long time to receive an update. With a retained message, the client will receive an instant update.

8.2.2 MQTT More Details and Examples

Referenced from <http://www.hivemq.com/blog/mqtt-essentials/> and <http://mosquitto.org/>. A full explanations of how MQTT function can be found in *section 8.6*.

MQTT History

MQTT was invented by Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom, now Cirrus Link) back in 1999, when their use case was to create a protocol for minimal battery loss and minimal bandwidth connecting oil pipelines over satellite connection. They specified the following goals, which the future protocol should have:

- Simple to implement
- Provide a Quality of Service Data Delivery
- Lightweight and Bandwidth Efficient
- Data Agnostic
- Continuous Session Awareness

These goals are still the core of MQTT, while the focus has changed from proprietary embedded systems to open Internet of Things use cases. Another thing that is often confused about MQTT is the appropriate meaning of the abbreviation MQTT. It's a long story, the short answer is that MQTT officially does not have an acronym anymore, it's just MQTT.

OASIS Standard

Around 3 years after the initial publication, it was announced that MQTT should be standardized under the wings of OASIS, an open organization with the purpose of advancing standards. On October 29th 2014 [MQTT was officially approved as OASIS Standard](#). MQTT 3.1.1 is now the newest version of the protocol.

Definition of Client/Broker

Client

When talking about a client it almost always means an MQTT client. This includes publisher or subscribers, both of them label an MQTT client that is only doing publishing or subscribing. (In general a MQTT client can be both a publisher & subscriber at the same time). A MQTT client is any device from a micro controller up to a full-fledged server that has a MQTT library running and is connecting to an MQTT broker over any kind of network. This could be a really small and resource constrained device that is connected over a wireless network and has a library strapped to the minimum or a typical computer running a graphical MQTT client for testing purposes, basically any device that has a TCP/IP stack and speaks MQTT over it.

Broker

The counterpart to a MQTT client is the MQTT broker, which is the heart of any publish/subscribe protocol. Depending on the concrete implementation, a broker can handle up to thousands of concurrently connected MQTT clients. The broker is primarily responsible for receiving all messages, filtering them, decide who is interested in it and then sending the message to all subscribed clients. Another responsibility of the broker is the authentication and authorization of clients. And at most of the times a broker is also extensible, which allows to easily integrate custom authentication, authorization and integration into backend systems. Especially the integration is an important aspect, because often the broker is the component, which is directly exposed on the internet and handles a lot of clients and then passes messages along to downstream analyzing and processing systems. All in all the broker is the central hub, which every message needs to pass.

Note: A broker has only 1 message per topic therefore for data acquisition a server client (cloud storage) or any such devices with data storage capability needs to be used. They will subscribe to the broker and store all the messages seen.

Quality of Service Expanded

Higher levels of QoS are more reliable, but involve higher latency and have higher bandwidth requirements.

0: The broker/client will deliver the message once, with no confirmation. Since there is no confirmation the message might not be delivered if connection is bad. This is often called “fire and forget” and provides the same guarantee as the underlying TCP protocol.

1: The broker/client will deliver the message at least once, with confirmation required. Will send message till confirmation received so possible that multiples of the message can exist.

2: The broker/client will deliver the message exactly once by using a four step handshake. Will always have exactly one of the message delivered. It is the slowest quality of service level. Currently not supported by the EZRack PLC.

The client chooses the maximum QoS it will receive. For example, if a message is published at QoS 2 and a client is subscribed with QoS 0, the message will be delivered to that client with QoS 0. If a second client is also subscribed to the same topic, but with QoS 2, then it will receive the same message but with QoS 2. For a second example, if a client is subscribed with QoS 2 and a message is published on QoS 0, the client will receive it on QoS 0.

Best Practice

The following should provide you some guidance if you are also confronted with this decision. Often this is heavily depending on your use case.

Use QoS 0 when ...

- You have a complete or almost stable connection between sender and receiver. A classic use case is when connecting a test client or a front end application to a MQTT broker over a wired connection.
- You don't care if one or more messages are lost once a while. That is sometimes the case if the data is not that important or will be send at short intervals, where it is okay that messages might get lost.
- You don't need any message queuing. Messages are only queued for disconnected clients if they have QoS 1 or 2 and a persistent session.

Use QoS 1 when ...

- You need to get every message and your use case can handle duplicates. The most often used QoS is level 1, because it guarantees the message arrives at least once. Of course your application must be tolerating duplicates and process them accordingly.
- You can't bear the overhead of QoS 2. Of course QoS 1 is a lot faster in delivering messages without the guarantee of level 2.

Use QoS 2 when ...

- It is critical to your application to receive all messages exactly once. This is often the case if a duplicate delivery would do harm to application users or subscribing clients. You should be aware of the overhead and that it takes a bit longer to complete the QoS 2 flow. Currently not supported by the EZRack PLC.

Queuing of QoS 1 and 2 messages

All messages sent with QoS 1 and 2 will also be queued for offline clients, until they are available again. But queuing is only happening, if the client has a persistent session (durable connection).

Topics

A topic is a UTF-8 string, which is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).

WildCards (Topics)

For topic navigation there exist wildcards. Two wildcards are available, + or # for use in topics. These allow for easier access to different ranges of topics.

+ can be used as a wildcard for a single level of hierarchy. An example of its use:

sensors/+/temperature/+

can be used as a wildcard for all remaining levels of hierarchy. This means that it must be the final character in a subscription. An example of its use:

sensors/machine/temperature/#

Example +

As another example, for a topic of "a/b/c/d", the following example subscriptions will match:

a/b/c/d +/b/c/d a+/c/d a+/+/d +/+/+/+

The following subscriptions will not match:

a/b/c b+/c/d +/+/+

Example

With a topic of "a/b/c/d", the following example subscriptions will match:

a/# a/b/# a/b/c/# +/b/c/#

Topic Best Practices

So these were the basics about MQTT message topics. As you can see, MQTT topics are dynamically and give great flexibility to its creator. But when using these in real world applications there are some challenges you should be aware of.

Don't use a leading forward slash

It is allowed to use a leading forward slash in MQTT, for example `/myhome/groundfloor/livingroom`. But that introduces an unnecessary topic level with a zero character at the front. That should be avoided, because it doesn't provide any benefit and often leads to confusion.

Don't use spaces in a topic

A space is the natural enemy of each programmer, they often make it much harder to read and debug topics, when things are not going the way, they should be. So similar to the first one, only because something is allowed doesn't mean it should be used. UTF-8 knows many different white space types, it's pretty obvious that such uncommon characters should be avoided.

Keep the topic short and concise

Each topic will be included in every message it is used in, so you should think about making them short and concise. When it comes to small devices, each byte counts and makes really a difference.

Use only ASCII characters, avoid non printable characters

Using non-ASCII UTF-8 character makes it really hard to find typos or issues related to the character set, because often they cannot be displayed correctly. Unless it is really necessary we recommend avoid using non ASCII character in a topic.

Embed a unique identifier or the ClientId into the topic

In some cases it is very helpful, when the topic contains a unique identifier of the client the publish is coming from. This helps identifying, who send the message. Another advantage is the enforcement of authorization, so that only a client with the same ClientId as contained in the topic is allowed to publish to that topic. So a client with the id `client1` is allowed to publish to `client1/status`, but not permitted to publish to `client2/status`.

Don't subscribe to

Sometimes it is necessary to subscribe to all messages, which are transferred over the broker, for example when persisting all of them into a database. This should not be done by using a MQTT client and subscribing to the multi level wildcard. The reason is that often the subscribing client is not able to process the load of messages that is coming its way. Especially if you have a

massive throughput. The recommended solution is to implement an extension in the MQTT broker.

Don't forget extensibility

Topics are a flexible concept and there is no need to preallocate them in any kind of way, regardless both the publisher and subscriber need to be aware of the topic. So it is important to think about how they can be extended in case you are adding new features to your product. For example when your smart home solution is extended by some new sensors, it should be possible to add these to your topic tree without changing the whole topic hierarchy.

Use specific topics, instead of general ones

When naming topics it is important not to use them like a queue, for example using only one topic for all messages is an anti pattern. You should use as specific topics as possible. So if you have three sensors in your living room, you should use topics `myhome/livingroom/temperature`, `myhome/livingroom/brightness` and `myhome/livingroom/humidity`, instead of sending all values over `myhome/livingroom`.

Persistent session / Durable connections

When a client connects to a MQTT broker, it needs to create subscriptions for all topics that it is interested in in order to receive messages from the broker. On a reconnect these topics are lost and the client needs to subscribe again. This is the normal behavior with no persistent session. But for constrained clients with limited resources it would be a burden to subscribe again each time they lose the connection. So a persistent session saves all information relevant for the client on the broker. The session is identified by the `clientId` provided by the client on connection establishment (more details).

So what will be stored in the session?

- Existence of a session, even if there are no subscriptions
- All subscriptions
- All messages in a Quality of Service (QoS) 1 or 2 flow, which are not confirmed by the client
- All new QoS 1 or 2 messages, which the client missed while it was offline
- All received QoS 2 messages, which are not yet confirmed to the client

That means even if the client is offline all the above will be stored by the broker and are available right after the client reconnects.

How to start/end a persistent session?

A persistent session can be requested by the client on connection establishment with the broker. The client can control, if the broker stores the session using the clean `Session` flag. If the

clean session is set to true then the client does not have a persistent session and all information are lost when the client disconnects for any reason. When clean session is set to false, a persistent session is created and it will be preserved until the client requests a clean session again. If there is already a session available then it is used and queued messages will be delivered to the client if available.

Best practices

When you should use a persistent session and when a clean session?

Persistent Session

- A client must get all messages from a certain topic, even if it is offline. The broker should queue the messages for the client and deliver them as soon as the client is online again.
- A client has limited resources and the broker should hold its subscription, so the communication can be restored quickly after it got interrupted.
- The client should resume all QoS 1 and 2 publish messages after a reconnect.

Clean session

- A client is not subscribing, but only publishing messages to topics. It doesn't need any session information to be stored on the broker and publishing messages with QoS 1 and 2 should not be retried.
- A client should explicitly not get messages for the time it is offline.

How long are messages stored on the broker?

An often asked question is how long is a session stored on the broker. The easy answer is until the clients comes back online and receives the message. But what happens if a client does not come online for a long time? The constraint for storing messages is often the memory limit of the operating system. There is no standard way on what to do in this scenario. It totally depends on the use case and the broker.

Retained Messages

A retained message is a normal MQTT message with the retained flag set to true. The broker will store the last retained message and the corresponding QoS for that topic. Each client that subscribes to a topic pattern, which matches the topic of the retained message, will receive the message immediately after subscribing. For each topic only one retained message will be stored by the broker.

The subscribing client can identify if a received message was a retained message or not, because the broker sends out retained messages with the retained flag still set to true. A client can then decide on how to process the message.

So retained messages can help newly subscribed clients to get a status update immediately after subscribing to a topic and don't have to wait until a publishing clients send the next update.

In other words a retained message on a topic is the last known good value, because it doesn't have to be the last value, but it certainly is the last message with the retained flag set to true.

It is important to understand that a retained message has nothing to do with a persistent session of any client. Once a retained message is stored by the broker, the only way to remove it is explained below.

Send a retained message

Sending a retained message from the perspective of a developer is quite simple and straightforward. You just need to set the retained flag of a MQTT publish message to true. Each client library typically provides an easy way to do that.

Delete a retained message

There is also a very simple way for deleting a retained message on a topic: Just send a retained message with a zero byte payload on that topic where the previous retained message should be deleted. The broker deletes the retained message and all new subscribers won't get a retained message for that topic anymore. Often deleting is not necessary, because each new retained message will overwrite the last one.

Why and when you should use Retained Messages?

A retained message makes sense, when newly connected subscribers should receive messages immediately and shouldn't have to wait until a publishing client sends the next message. This is extremely helpful when for status updates of components or devices on individual topics. For example the status of device1 is on the topic `myhome/devices/device1/status`, a new subscriber to the topic will get the status (online/offline) of the device immediately after subscribing when retained messages are used. The same is true for clients, which send data in intervals, temperature, GPS coordinates and other data. Without retained messages new subscribers are kept in the dark between publish intervals. So using retained messages helps to provide the last good value to a connecting client immediately.

Last Will and Testament

When a client connects to a broker, it may inform the broker that it has a will. This is a message that it wishes the broker to send when the client disconnects unexpectedly. The will message has a topic, QoS and retain status just the same as any other message. EZRack PLC currently does not support Wills.

When will a broker send the LWT message?

According to the MQTT 3.1.1 specification the broker will distribute the LWT of a client in the following cases:

- An I/O error or network failure is detected by the server.
- The client fails to communicate within the Keep Alive time.
- The client closes the network connection without sending a DISCONNECT packet first.
- The server closes the network connection because of a protocol error.
- We will hear more about the Keep Alive time in the next post.

Best Practices – When should you use LWT?

LWT is ideal for notifying other interested clients about the connection loss. In real world scenarios LWT is often used together with retained messages, in order to store the state of a client on a specific topic. For example after a client has connected to a broker, it will send a retained message to the topic client1/status with the payload “online”. When connecting to the broker, the client sets the LWT message on the same topic to the payload “offline” and marks this LWT message as a retained message. If the client now disconnects ungracefully, the broker will publish the retained message with the content “offline”. This pattern allows for other clients to observe the status of the client on a single topic and due to the retained message even newly connected client now immediately the current status.

8.3 Basic MQTT Setup on EZRack PLC

The EZRack PLC MQTT Publish instructions is looked at in *Section 3.3.16*. But before the instruction can be used the MQTT Broker information needs to be configured. To do this please go to **Setup > MQTT Setup....** The needed information for this setup is:

Information Type	Description	Example
Domain Name	This is the broker URL. Used to find your broker that you have configured.	m12.cloudmqtt.com
Port Number	Port number that your broker uses.	16581
Client ID	Individual connection ID. Needs to be different for every client otherwise will encounter problems. Can be random.	ee097f5c-fa36-4929-9414-fad17b3df3bd
User Name	Your configured username for EZRack PLC connection to broker. Should be different for every client.	
Password	Your configured password for EZRack PLC connection to broker. Should be different for every client.	

Instruction to setup MQTT:

1. Go to **Setup > MQTT Setup....** You will see the following dialog box appear.

2. Use the Domain Name Lookup with the Domain Name from the broker to find the Broker IP Address.

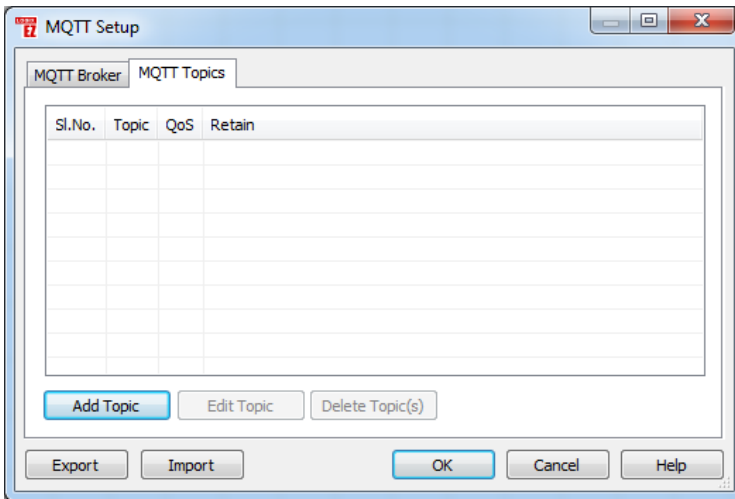
3. Enter the port number from the broker.

4. Select your keep alive interval if wanted. See *section 8.6* for more information.

5. Enter a unique client ID or generate one using the Generate

Unique Id button.

6. Enter the user name and password for your broker.
7. Go to the MQTT topics.



8. In the MQTT Topics use the Add Topic button to create the prefixes for your tags. The publish instruction will publish the tagname as a topic but if you want to have more topic information create the prefix here. For example:

Note: After this topic an "/" is appended

Topic: EZRack PLCPLC/Machine1

TagName: Speed

Published Topic: EZRack PLCPLC/Machine1/Speed

9. Now in your ladder logic add the IIoT (MQTT) Publish instruction and configure it. For configuration options please see Section 3.3.16.

8.4 Broker Setup

The EZRack PLC can work with any third party broker. It has been tested and used extensively with the CloudMQTT broker. This section will go through some important information about setup of your broker.

CloudMQTT has a free plan for testing purposes. Please see below for setup instructions.

Broker Setup Basics

1. For any broker you can go to their website and create an account. For the CloudMQTT broker you go to <https://www.cloudmqtt.com/>.



2. Then the plans section will give you information on the different plans available and their cost. The documentation provides information about how MQTT works. Support is the CloudMQTT Tech Support. Finally the Control Panel is what you use to create the MQTT connection.
3. After going to Control Panel, please create an account or login to an account.
4. In the account create a new CloudMQTT Instance.

Create new CloudMQTT Instance

PayPal not enabled. Please [enable PayPal](#) if you want to subscribe to a paid plan

Name

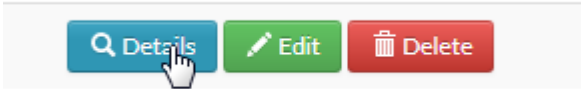
Data center

Plan

To learn more about the different plans please visit: www.cloudmqtt.com/plans.html

5. Enter a Name, select the Data Center and then for the free plan use the Cute Cat plan.

6. Once the Instance is create click on details to find the information needed to subscribe to this broker.



7. The Instance Info is the information that is needed for both the EZRack PLC Designer Pro and EZ-IIoT Subscriber Utility.

Instance info

Server

User

Password

Port

Manage Users section. You just need to provide the username and password.

Manage Users

10. Finally you can create ACL rules which govern what each user can access. This allows for management and distribution of topics to the correct people.

ACLs

Note:

- You have to set a acl rule for a custom user before it can access anything
- Use # for multi level wildcard ACL
- Use + for single level wildcard ACL

For API docs look at [HTTP API](#)

8. This information provides the details for this connections where:

9. You can also add more users in the

EZRack PLC	Instance Info
Domain Name	Server
Port Number	Port
Client ID	N.A.
User Name	User
Password	Password

11. You have now configured your broker and it can be used with the EZRack PLC and the EZ-IIoT Subscriber Utility.

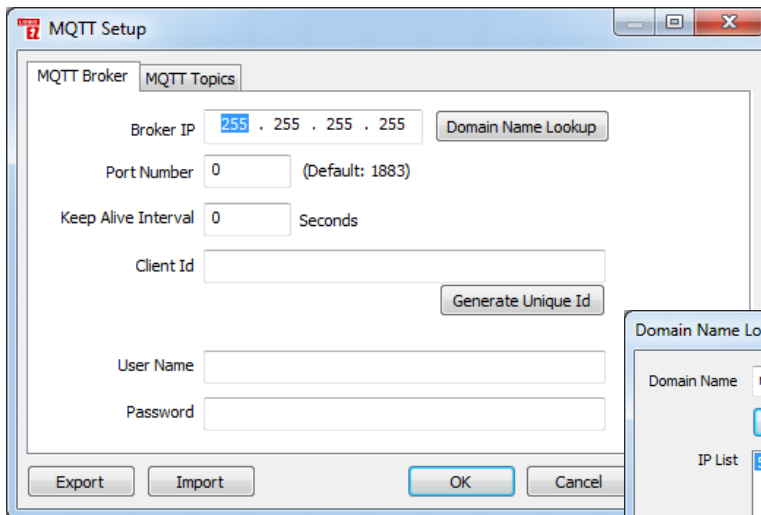
8.5 EZRack PLC IIoT (MQTT) Example

This sections shows the creation of an IIoT (MQTT) Publish instruction from start to finish in a project. It requires that the user has created a broker and has broker information.

Used Broker Information:

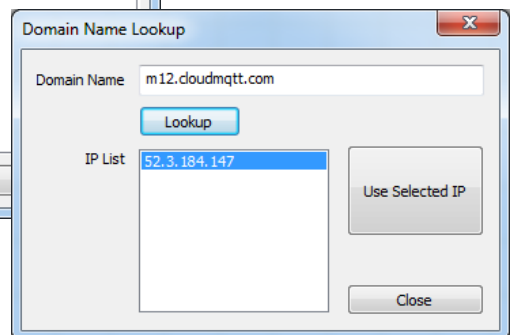
Information Type	Information
Domain Name	m12.cloudmqtt.com
Port Number	16581
Client ID	Test-ID0001
User Name	TEST
Password	AVG123

1. In a open project go to **Setup > MQTT Setup...**



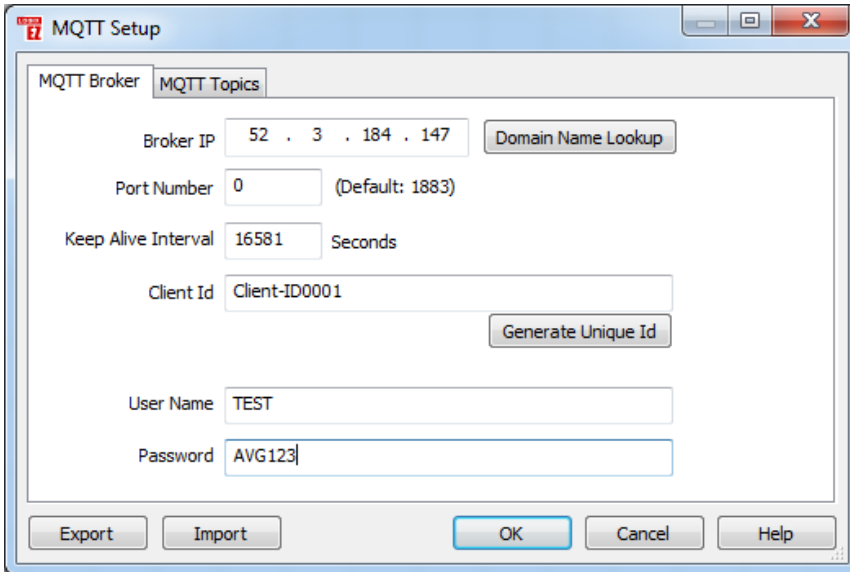
2. Click on Domain Name Lookup.

3. Enter the domain name and press Lookup. This will find the domain's IP address. Once found press Use Selected IP.



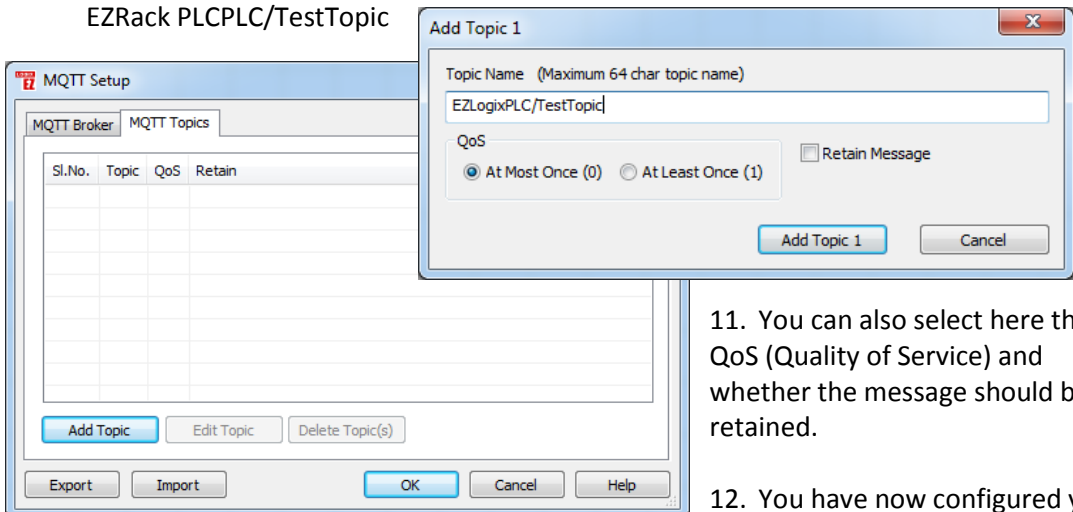
4. The Broker IP will now have been entered.
5. Next input the port number (16581).
6. For this example we keep the Keep Alive Interval at 0.
7. Enter the Client ID or generate an Unique one.
8. Finally add your broker username and password.

9. The final result should look something like this.



10. Now go to the MQTT Topics. Use the Add Topic to add a topic, for example:

EZRack PLCPLC/TestTopic



11. You can also select here the QoS (Quality of Service) and whether the message should be retained.

12. You have now configured your MQTT connection. Next you need to add the IIoT (MQTT) Publish instruction.



13. In the sidebar select the IIoT (MQTT) Publish instruction and add it to your logic. Double click on the instruction to bring up the configuration dialog.

IIoT (MQTT) Publish Instruction

Instruction Details

Broker and Topic

Publish to Broker: 52.3.184.147:16581 MQTT Setup

Topic

EZLogixPLC/TestTopic

Retain Message: No, QoS: At Most Once

Publish

Publish Type: On Rising Edge of Event Tag

Event/Enable Tag

Publish Time-interval: 0 Minute

Publish Status Tag

Status value definitions:

00: Normal operation (No Errors) 02: Connect failure
64: Done 04: Publish failure

Select Tags

For string tags > 40 char, only 40 char would be included in the value.

Decimal Places for Floating Point Tags: 5

Available Tags:

Name	Address	Type
PUBLISH TAG	R1	UNSIGNED_INT_16

>>

<<

Selected Tags: (0/10)

Name	Address	Type

Delete Tag(s)

Move Tag Up

Move Tag Down

OK Cancel Help

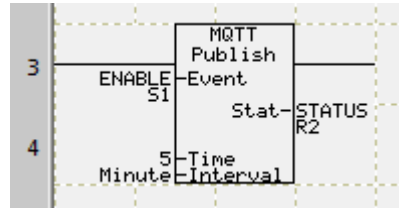
14. Under publish select the type of publishing you would like. For this example it will be At Regular Time Intervals (When Enable Tag is High).

15. Now add an Enable Tag, set the Publish Time-interval to 5 Minutes, and add an Status Tag.

16. Finally move the publish tag to the selected tag area. Final result will look like this:

Where this instruction will publish the Publish Tag to the broker every 5 minutes when the Enable (S1) tag is ON.

The published topic will be:
EZRack PLCPLC/TestTopic/PUBLISH TAG



Published value will include a timestamp and the current value of PUBLISH TAG (R1).

IIoT (MQTT) Publish Instruction

Instruction Details

Broker and Topic

Publish to Broker: 52.3.184.147:16581 MQTT Setup

Topic: EZLogixPLC/TestTopic

Retain Message: No, QoS: At Most Once

Publish

Publish Type: At Regular Time Intervals (When Enable Tag is High)

Event/Enable Tag: ENABLE

Publish Time-interval: 5 Minute

Publish Status Tag: STATUS

Status value definitions:

- 00: Normal operation (No Errors)
- 02: Connect failure
- 64: Done
- 04: Publish failure

Select Tags

For string tags > 40 char, only 40 char would be included in the value.

Decimal Places for Floating Point Tags: 5

Available Tags:

Name	Address	Type

>>

<<

Selected Tags: (1/10)

Name	Address	Type
PUBLISH TAG	R 1	UNSIGNED_INT_16

Delete Tag(s)

Move Tag Up

Move Tag Down

OK Cancel Help

8.6 MQTT HIVEMQ Essentials

Most of this section has been taken from <http://www.hivemq.com/blog/mqtt-essentials/> and is their MQTT essentials blog posts. It has been condensed here for to describe the basics of MQTT and how it functions.

Pub/Sub Pattern

As already mentioned the main aspect in pub/sub is the decoupling of publisher and receiver, which can be differentiated in more dimensions:

- Space decoupling: Publisher and subscriber do not need to know each other (by ip address and port for example)
- Time decoupling: Publisher and subscriber do not need to run at the same time.
- Synchronization decoupling: Operations on both components are not halted during publish or receiving

In summary publish/subscribe decouples publisher and receiver of a message, through filtering of the messages it is possible that only certain clients receive certain messages. The decoupling has three dimensions: Space, Time, and Synchronization.

Scalability

Pub/Sub also provides a greater scalability than the traditional client-server approach. This is because operations on the broker can be highly parallelized and processed event-driven. Also often message caching and intelligent routing of messages is decisive for improving the scalability.

Message Filtering

So what's interesting is, how does the broker filter all messages, so each subscriber only gets the messages it is interested in?

Option 1: Subject-based filtering

The filtering is based on a subject or topic, which is part of each message. The receiving client subscribes on the topics it is interested in with the broker and from there on it gets all message based on the subscribed topics. Topics are in general strings with an hierarchical structure, that allow filtering based on a limited number of expression.

Option 2: Content-based filtering

Content-based filtering is as the name already implies, when the broker filters the message based on a specific content filter-language. Therefore clients subscribe to filter queries of messages they are interested in. A big downside to this is, that the content of the message must be known beforehand and cannot be encrypted or changed easily.

Option 3: Type-based filtering

When using object-oriented languages it is a common practice to filter based on the type/class of the message (event). In this case a subscriber could listen to all messages, which are from type Exception or any subtype of it.

There are some drawbacks to consider. The decoupling of publisher and subscriber, which is the key in pub/sub, brings a few challenges with it. You have to be aware of the structuring of the published data beforehand. In case of subject-based filtering, both publisher and subscriber need to know about the right topics to use. Another aspect is the delivery of message and that a publisher can't assume that somebody is listening to the messages he sends. Therefore it could be the case that a message is not read by any subscriber.

Distinction from Message Queues

So there are many confusions about MQTT, its name and if it is implemented as a message queue or not. We will try to bring light into the dark and explain the differences. In our last post we already pointed out that the name MQTT comes from an IBM product called MQseries and has nothing to do with "message queue". But regardless of the name, what are the differences between MQTT and a traditional message queue?

A message queue stores message until they are consumed

When using message queues, each incoming message will be stored on that queue until it is picked up by any client (often called consumer). Otherwise the message will just be stuck in the queue and waits for getting consumed. It is not possible that message are not processed by any client, like it is in MQTT if nobody subscribes to a topic.

A message will only be consumed by one client

Another big difference is the fact that in a traditional queue a message is processed by only one consumer. So that the load can be distributed between all consumers for a particular queue. In MQTT it is quite the opposite, every subscriber gets the message, if they subscribed to the topic.

Queues are named and must be created explicitly

A queue is far more inflexible than a topic. Before using a queue it has to be created explicitly with a separate command. Only after that it is possible to publish or consume messages. In MQTT topics are extremely flexible and can be created on the fly.

MQTT Connection Information

Below is all basic information that is necessary to connect to a MQTT broker from a MQTT client.

ClientId

The client identifier (short ClientId) is an identifier of each MQTT client connecting to a MQTT broker. As the word identifier already suggests, it should be unique per broker. The broker uses it for identifying the client and the current state of the client. If you don't need a state to be hold by the broker, in MQTT 3.1.1 (current standard) it is also possible to send an empty ClientId, which results in a connection without any state. A condition is that clean session is true, otherwise the connection will be rejected.

Clean Session

The clean session flag indicates the broker, whether the client wants to establish a persistent session or not. A persistent session (Clean Session is false) means, that the broker will store all subscriptions for the client and also all missed messages, when subscribing with Quality of Service (QoS) 1 or 2. If clean session is set to true, the broker won't store anything for the client and will also purge all information from a previous persistent session.

Username/Password

MQTT allows to send a username and password for authenticating the client and also authorization. However, the password is sent in plaintext, if it isn't encrypted or hashed by implementation or TLS is used underneath. We highly recommend to use username and password together with a secure transport of it. In brokers like HiveMQ it is also possible to authenticate clients with an SSL certificate, so no username and password is needed.

Will Message

The will message is part of the last will and testament feature of MQTT. It allows to notify other clients, when a client disconnects ungracefully. A connecting client will provide his will in form of an MQTT message and topic in the CONNECT message. If this clients gets disconnected ungracefully, the broker sends this message on behalf of the client. We will talk about this in detail in an individual post.

Keep Alive

The keep alive is a time interval, the clients commits to by sending regular PING Request messages to the broker. The broker response with PING Response and this mechanism will allow both sides to determine if the other one is still alive and reachable. We'll talk about this in detail in a future post.

Publish Functionality

After a MQTT client is connected to a broker, it can publish messages. MQTT has a topic-based filtering of the messages on the broker, so each message must contain a topic, which will be used by the broker to forward the message to interested clients. Each message typically has a payload which contains the actual data to transmit in byte format. EZRack PLC MQTT Publish sends the data in basic text with time stamp included. Below is some more information on the message attributes:

Topic Name

A simple string, which is hierarchically structured with forward slashes as delimiters. An example would be "myhome/livingroom/temperature" or "Germany/Munich/Octoberfest/people".

QoS

A Quality of Service Level (QoS) for this message. The level (0, 1 or 2) determines the guarantee of a message reaching the other end (client or broker).

Retain-Flag

This flag determines if the message will be saved by the broker for the specified topic as last known good value. New clients that subscribe to that topic will receive the last retained message on that topic instantly after subscribing.

Payload

This is the actual content of the message. EZRack PLC MQTT Publish sends the data in basic text with time stamp included.

Packet Identifier

The packet identifier is a unique identifier between client and broker to identify a message in a message flow. This is only relevant for QoS greater than zero. Setting this MQTT internal identifier is the responsibility of the client library and/or the broker.

DUP flag

The duplicate flag indicates, that this message is a duplicate and is resent because the other end didn't acknowledge the original message. This is only relevant for QoS greater than 0. This resend/duplicate mechanism is typically handled by the MQTT client library or the broker as an implementation detail.

Subscribe Functionality

Publishing messages doesn't make sense if no one ever receives the message, or, in other words, if there are no clients subscribing to any topic. A client needs to send a SUBSCRIBE message to the MQTT broker in order to receive relevant messages. A subscribe message is pretty simple, it just contains a unique packet identifier and a list of subscriptions.

Packet Identifier

The packet identifier is a unique identifier between client and broker to identify a message in a message flow. This is only relevant for QoS greater than zero. Setting this MQTT internal identifier is the responsibility of the client library and/or the broker.

List of Subscriptions

A SUBSCRIBE message can contain an arbitrary number of subscriptions for a client. Each subscription is a pair of a topic and QoS level. The topic in the subscribe message can also contain wildcards, which makes it possible to subscribe to certain topic patterns. If there are overlapping subscriptions for one client, the highest QoS level for that topic wins and will be used by the broker for delivering the message.

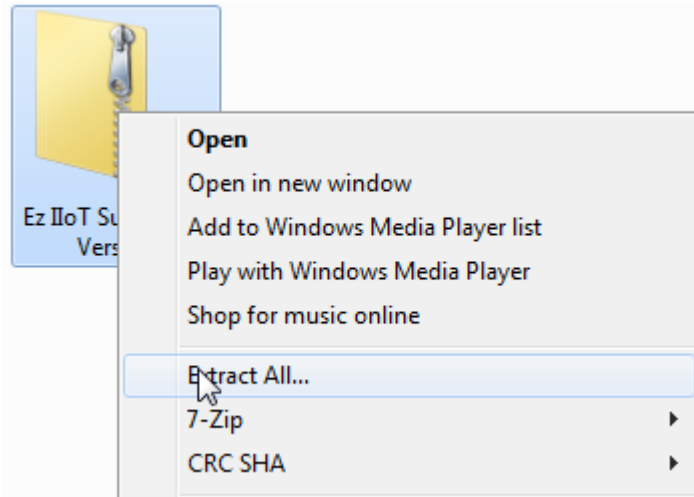
8.7 EZ-IIoT Subscriber Utility

The data EZRack PLC publishes to the broker is accessible through any third party subscriber utility but EZ Automation has created its own take on this utility. The EZAutomation subscriber utility is developed to make it very easy to see current updated information as well as store any previously published information. This utility will data log any MQTT messages that it sees when subscribed to the broker.

8.7.1 Install EZ-IIoT Subscriber Utility

The EZ-IIoT Subscriber Utility is a separate setup which can be downloaded from www.EZAutomation.com. The EZ-IIoT Subscriber Utility can be installed on any computer that the EZRack PLC Designer Pro can and at least 2 MB of free space on hard drive for installation. Follow directions below to setup the utility.

1. Download the EZ-IIoT Subscriber Utility ZIP file from the website.



2. Extract the zip folder to the location where you want to place the utility.
3. The utility will now run. Please follow directions below to setup your broker connection.

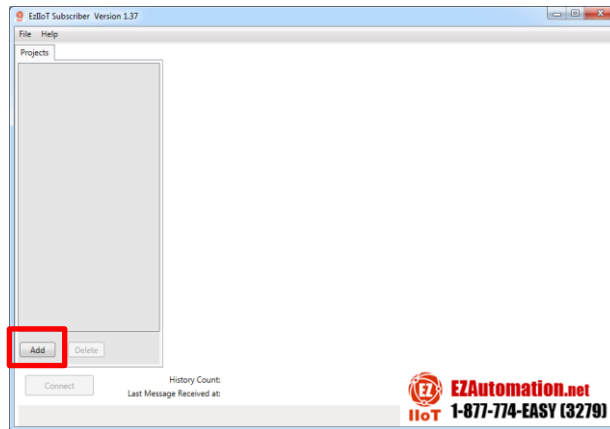
Note: The EZ-IIoT Subscriber Utility requires .NET Framework 4.5 which you might need to install from the Microsoft Windows Website.

8.7.2 EZ-IIoT Subscriber Utility Setup

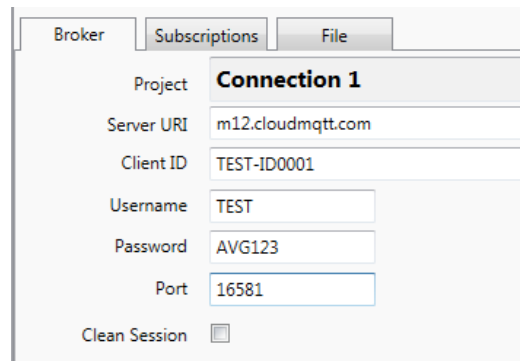
The EZ-IIoT Subscriber Utility is very easy to setup. The only information needed is listed in the table below. To setup the utility please follow the instructions below.

Information Type	Example Information
Domain Name (Server URI)	m12.cloudmqtt.com
Client ID	Test-ID0001
User Name	TEST
Password	AVG123
Port Number	16581

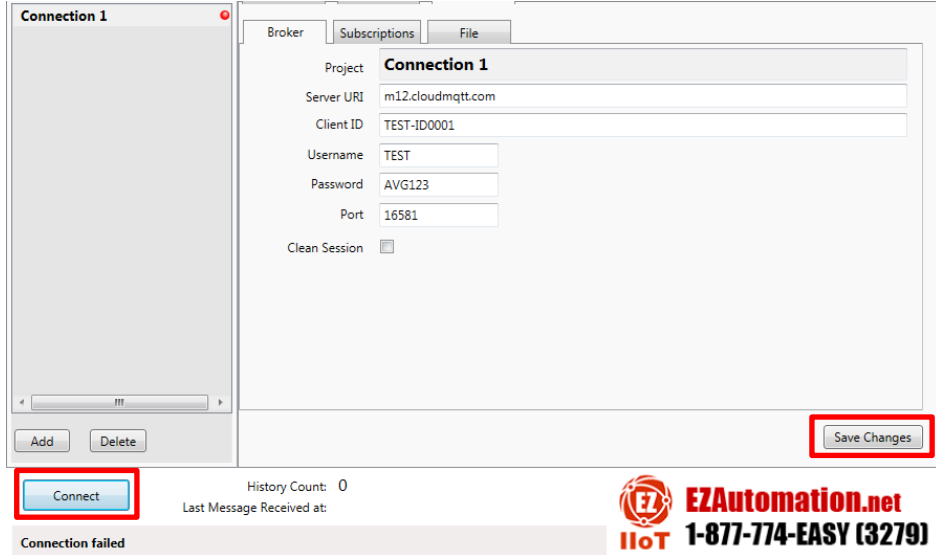
1. Open the EZ-IIoT Subscriber Utility. In the projects are click the “Add” button.



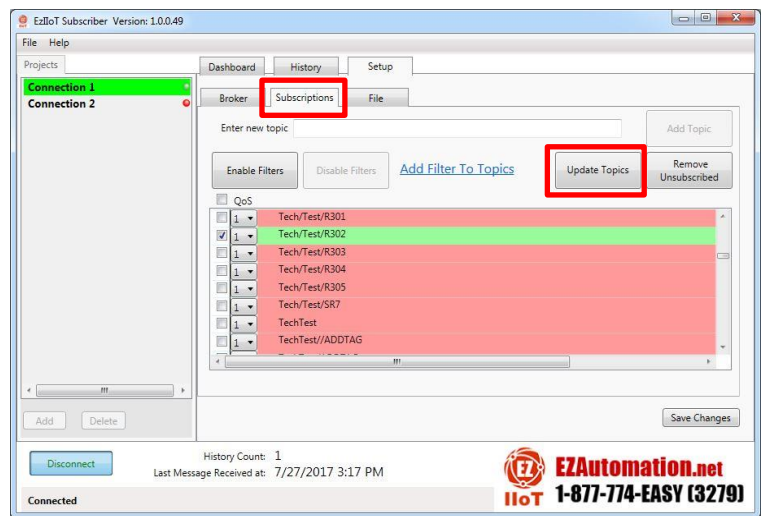
2. In the new connection enter the information from the broker. The example shown uses the example information in the table above. You can also rename the project in the Broker Setup window.



- Click the “Save Changes”. You will now have the Connect option in the information below. Use the “Connect” button to connect to your broker.



- As soon as you are connected the Project will turn green. Now in the Setup tab go to the Subscriptions tab. Click the Update Topics to get the topics you have access to. This will only retrieve topics that have been published with the 'Retain Flag' set to



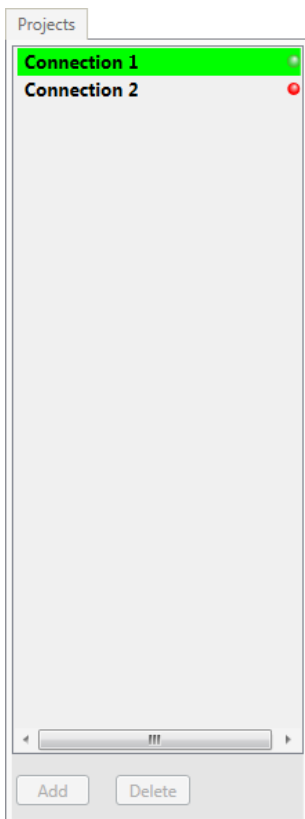
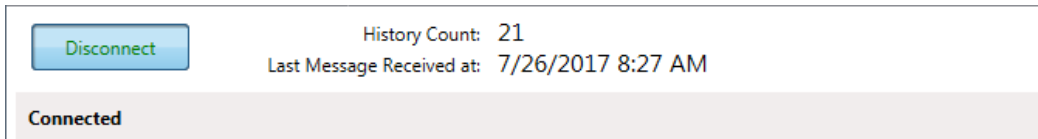
true AND have been published at least once. If this is not true you can add any topic you would like. Then select Topics you would like to subscribe to. Once select the topic is subscribed and you will now be updated in the History tab about its value. Please see the next section for the full functionality of the Utility.

8.7.3 EZ-IIoT Subscriber Utility Functions

The EZ-IIoT Subscriber Utility has 5 tabs total for its full functionality. This section will go through the 5 tabs and list its functionality. There are 2 main tabs (Dashboard, History) and 3 setup tabs (Broker, Subscriptions, and File).

Connection Status

The connection status is visible in all tabs and allows the user to connect and disconnect from the broker. It also lists the current history count and when the last message was received. If there are any errors they will also be listed here.



Projects List

The project list allows switching between all the different connection setups. Only one connection can be connected at a time. New connection can only be added when you are disconnected from the broker. The green light indicates which project/connection is actively connected to a broker.

Use the **Add** and **Delete** buttons to add and delete connections when not connected to broker.

Dashboard Tab

The dashboard is the main view screen for any Project / Connection. It allows the user to have an overview of this broker connection and monitor any important topics.

The screenshot shows the Dashboard Tab interface with the following elements and callouts:

- Tab Navigation:** A box containing 'Dashboard', 'History', and 'Setup' tabs.
- Project/Connections Name:** A box containing 'Project: **Connection 1**'.
- Topic Information (See Subscription Tab):** A box containing 'Topics Subscribed / Available: 9 / 149'.
- Add Topics to Dashboard:** A button labeled 'Add Topics to Dashboard'.
- Remove All:** A button labeled 'Remove All'.
- Topic Cards:** Two topic cards are shown: 'Topic' and 'Topic 2'. Each card has a close button (X) in the top right corner.
- Use this to add important topics to the dashboard to monitor its value and status.** A callout pointing to the 'Add Topics to Dashboard' button.
- Removes all topics from dashboard. Does not unsubscribe.** A callout pointing to the 'Remove All' button.
- Each individual topic added to dashboard will have its information box. See below for more information.** A callout pointing to a detailed topic card for 'Tech/Test/R300'.

The detailed topic card for 'Tech/Test/R300' shows:

- Topic name: Tech/Test/R300
- Close button (X)
- Value: 47
- Published: 7/26/2017 8:27 AM
- Received: 7/26/2017 8:27 AM

The screenshot shows a highlighted topic card with the following elements:

- Topic name: Topic
- Close button (X)
- Value: Value
- Published: Date and Time published
- Received: Date and Time received

Dashboard Highlighted Topics

Any topic added to the dashboard will have a box appear where the current status / value can be monitored. This box will list the topic name at the top. The last received value is the value in the middle. Finally it will list the publish time and Utility receive time at the bottom. To eliminate this topic from the dashboard use the X or the Remove All option. Eliminating the topic from the dashboard does not

unsubscribe. *Note: Each time a new message is received for this topic it will flash to indicate status change.*

History Tab

The history tab lists all the received values from all subscribed topics. Filters exist to navigate and narrow down information. Also the history can be cleared. The connection status area will list the total count of received values from all topics listed in the history tab. The history can be saved manually but it is also saved automatically (please see **Setup > File** Tab for more information).

The screenshot shows the 'History' tab interface. At the top, there are navigation buttons for 'Dashboard', 'History', and 'Setup'. A callout labeled 'Tab Navigation' points to these buttons. Below the navigation is a header area for 'Connection 1' with a callout 'Project/Connections Name'. The main area contains a table of message history. Above the table are buttons for 'Enable Filters', 'Disable Filter', and 'Add Filter To Topics'. A callout 'Enable/Disable Filter Use Add Filter to Topics to create Filter (Please see next page for more information)' points to these buttons. To the right of the table are buttons for 'Clear History' and 'Save History'. Callouts explain these: 'Clear History (does not clear saved .csv file)' and 'Manually Save Current History in new .csv file'.

Unique Id	Received At	Topic	Broker Sent At	Message	QoS	Retained Flag	Dup. Flag
1246	7/26/2017 9:51:49 AM	Tech/Test/R300	7/26/2017 9:51:46 AM	124	0	NO	NO
1247	7/26/2017 9:51:49 AM	Tech/Test/R301	7/26/2017 9:51:46 AM	124	0	NO	NO
1248	7/26/2017 9:51:49 AM	Tech/Test/R302	7/26/2017 9:51:46 AM	124	0	NO	NO
1249	7/26/2017 9:51:49 AM	Tech/Test/R303	7/26/2017 9:51:46 AM	124	0	NO	NO
1250	7/26/2017 9:51:49 AM	Tech/Test/R304	7/26/2017 9:51:46 AM	124	0	NO	NO
1251	7/26/2017 9:51:49 AM	Tech/Test/R305	7/26/2017 9:51:46 AM	124	0	NO	NO
1252	7/26/2017 9:51:49 AM	Tech/Test/SR7	7/26/2017 9:51:46 AM	1	0	NO	NO

History Information

Unique ID – Each received message will have a unique ID number per connection which can be used to reference the received message. It can be used to search in the .csv file as well.

Received At – This is the time and date that the message was received by the Utility.

Topic – The subscribed topic name.

Broker Sent At – When the publisher sent the message to the broker. Can be incorrect if publisher (EZ Rack PLC) has wrong date and time.

Message – The actual message. The utility is formatted to expect EZ Rack PLC format of messages. The EZ Rack PLC messages are formatted to include the Time Stamp of when the message was sent and then the message value. The EZ Rack PLC publish format is “TimeStamp, Value”. Example below:

Received message: 1501073628, 291

The corresponding history result is:

Broker Sent At: 7/26/2017 12:53:48

Message: 291

QoS – Quality of Service from the publisher. Set on the publisher (EZ Rack PLC) side.

Retained Flag – This will tell you if it is a currently published message or if it is a retained message. The message will say “NO” for retained flag if you are subscribed while it is published. Otherwise if the message is set on the publisher side as retained you will receive the message as soon as you subscribe. Please see example below:

Event 1:

Utility: Subscribes

Publisher: Publish Message 1 with Retain Message set

Utility: Message 1 received and has “NO” for retained flag

Event 2:

Publisher: Publish Message 2 with Retain Message set

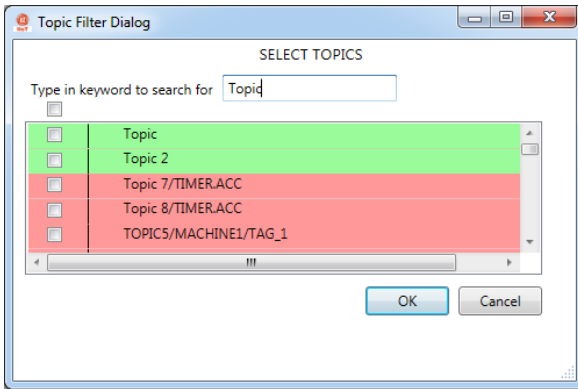
Utility: Subscribes

Utility: Message 2 received and has “YES” for retained flag

Note: Retain flag will not be “YES” unless the message was published before the user subscribed

Dup. Flag – The duplicate flag will be set to “YES” if the message has been received more than once by the Broker.

Topic Filter



The Topic Filter can be used to filter by different topics. Use the keyword selector to search for needed topics. Then select the topics you would like to see when filter is enabled. Click "OK" to finish setting up filter.

On the main screen use enable filter to see only previously selected topics.

Setup Broker

This tab is used to configure the broker information before connecting to the broker. Please see the setup instructions in the previous section for more information.

The screenshot shows a web interface for configuring a broker connection. At the top, there are three tabs: 'Dashboard', 'History', and 'Setup'. The 'Setup' tab is selected. Below the tabs, there are two sub-tabs: 'Broker' and 'File'. The 'Broker' sub-tab is selected. The main content area is titled 'Connection 1' and contains the following fields:

- Project: Connection 1
- Server URI: m12.cloudmqtt.com
- Client ID: TEST-ID0001
- Username: TEST
- Password: AVG123
- Port: 16581
- Clean Session:

At the bottom right, there is a 'Save Changes' button. The interface is annotated with several red boxes and arrows:

- 'Tab Navigation' points to the 'Setup' tab.
- 'Setup Navigation' points to the 'Broker' sub-tab.
- 'Needed Broker Information (Please see setup section)' points to the Username and Password fields.
- 'Select this if you would like to **Unsubscribe** from all topics when you **Disconnect** from the Broker' points to the 'Clean Session' checkbox.
- 'Make sure to save changes before connecting' points to the 'Save Changes' button.

Note: You cannot connect with the new settings until you save changes.

Setup Subscriptions

This tab is used to subscribe to different topics. This tab is only available when connected to the broker. You can either add a topic or subscribe to topics that already exist on the broker. Use the filter to narrow down the topics you would like to work with.

The screenshot shows the 'Setup Subscriptions' interface. At the top, there are three tabs: 'Dashboard', 'History', and 'Setup'. Below these are three sub-tabs: 'Broker', 'Subscriptions', and 'File'. A text input field labeled 'Enter new topic:' is on the right, with an 'Add Topic' button next to it. Below the input field are buttons for 'Enable Filters', 'Disable Filters', and a link 'Add Filter To Topics'. There are also 'Update Topics' and 'Remove Unsubscribed' buttons. The main area displays a table of topics. The first row is highlighted in green and contains a checked checkbox, a 'QoS' dropdown menu set to '1', and the topic name 'Tech/Test/R303'. A 'Delete' button is visible to the right of this row. A 'Save Changes' button is at the bottom right. Red callout boxes with arrows point to various elements: 'Tab Navigation' points to the top tabs; 'Setup Navigation' points to the sub-tabs; 'Enable/Disable Filter' points to the 'Enable Filters' and 'Disable Filters' buttons; 'Use Add Filter to Topics to create Filter (Please see next page for more information)' points to the 'Add Filter To Topics' link; 'Subscribe all visible topics' points to the checked checkbox in the first row.

How to Subscribe to a Topic

This close-up shows a row in the subscription table. It features a checked checkbox, a 'QoS' dropdown menu set to '1', and the topic name 'Tech/Test/R300'. The row is highlighted in green.

To subscribe just check the box next to the Topic you would like to subscribe to. You can change the Quality of Service (QoS) for communication between utility and Broker for that topic at any time by using the dropdown (QoS of 1 or 0 allowed). Also you can subscribe to all visible topics by using the check box next to QoS.

Update Topics

Update Topics

The update topics will download all topics that exist as retained messages on the broker. Only the topics that you have permission to see will be downloaded. You can also add any topics you would like at any time. This will only retrieve topics that have been published with the 'Retain Flag' set to true AND have been published at least once.

Remove
Unsubscribed

Remove Unsubscribed

The remove unsubscribed option will delete all unsubscribed topics currently visible in the Subscription window.

Delete

Delete

You can delete individual topics by right clicking on topic and selecting the delete option.

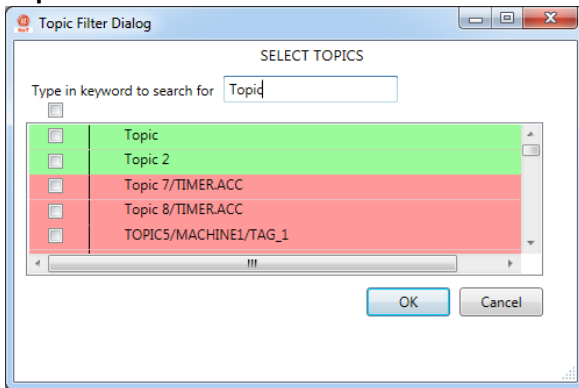
Add Topic

Add Topic

You can at any point add a topic to subscribe to by typing in the topic and pressing Add Topic.

Note: you will need to do this for any topic which does not have a retained flag since the update topics will not populate the list with these.

Topic Filter



The Topic Filter can be used to filter by different topics. Use the keyword selector to search for needed topics. Then select the topics you would like to see when filter is enabled. Click "OK" to finish setting up filter.

On the main screen use enable filter to see only previously selected topics.

Setup File

This tab is used to configure how the Utility will save the messages it has received. Here you can name the save file and change the save folder. You can also configure conditions of saving and when a new file is created.

The screenshot shows the 'Setup File' configuration window for 'Connection 1'. The window has a top navigation bar with 'Dashboard', 'History', and 'Setup' tabs. Below that is a sub-navigation bar with 'Broker', 'Subscriptions', and 'File' tabs. The main configuration area is titled 'File' and contains several sections:

- File Section:** Includes 'Base Name' (text input: TopicData) and 'Folder Name' (text input: C:\) with a 'Browse' button.
- Saving Action Section:** Contains two radio buttons: 'Automatically' (selected) and 'Manually'.
- Save To File Section:** Contains two checked checkboxes:
 - 'Append -YYMMDD-HHMM to Base Name when a new file is created.'
 - 'Create a new file after filesize exceeds 1000 Kbytes.'
- Bottom Right:** A 'Save Changes' button.

Annotations with red boxes and arrows point to various elements:

- 'Tab Navigation' points to the top navigation bar.
- 'Setup Navigation' points to the sub-navigation bar.
- A box pointing to the 'File' section says: 'Use this to set the name of the .csv and where it will be saved'.
- A box pointing to the 'Saving Action' section says: 'Select whether history is saved automatically or you need to save manually'.
- A box pointing to the 'Save To File' section says: 'These settings are used when Saving Action is set to Automatic. Use these settings to set when a new file is created and how it will be named.'
- A box pointing to the 'Save Changes' button says: 'Make sure to save changes since changes are not implemented till they are saved.'

Note: The newest data will always be saved in the Base Name .csv file. If new files are created then data is either saved in files with the date and time appended. Or if that format is not used the oldest files will be in "Base Name1.csv", second oldest in "Base Name2.csv". Also if the Base Name is open in excel, write is not possible so a new file with name "Base Name_.csv will be created.

8.7.4 EZ-IIoT Subscriber Utility Best Practices

This section will mention some common best practices when using the EZ-IIoT Subscriber Utility.

Utility Use

Recommended uses of this utility (can be used for multiple purposes at same time):

- Monitor tags – This utility can be used to monitor about 4-10 tags from the dashboard.
- Data Log – When this utility is subscribed it can be used to data log tag values for later analysis. *Note: it is stored as a .csv file.*
- Check Status – This utility can also be used to just check status of machine periodically by subscribing to see current status.
- Troubleshoot – This utility can also be used to see tag values for off-site troubleshooting capability.

CSV Files

When looking at saved history (data logging) in the CSV files the best way to view is to create a copy and then view in excel. If the CSV is open in excel the utility can write to it and will create a new file. Also note the oldest data will have a unique ID of 0 and the newest will have the highest value unique ID.

Client ID

Please make sure to use different Client IDs for each subscriber. If the same client ID is used for multiple subscribers only 1 will ever be able to connect to the Broker at a time. If the client IDs are different all subscribers up to your broker limit can connect to the Broker at the same time.

Username and Password

Each username and password can be limited to only certain topics thereby allowing users specific access to needed information. Therefore it is best to create a different username and password for each user. A username and password can be used in multiple locations to connect at the same time but it is not recommended.

EZ-IIoT Subscriber Utility Acceptable Data Format

The EZRack PLC publish format is “TimeStamp, Value”. The EZ-IIoT Subscriber Utility expects data in this format. Example below:

Received message: 1501073628, 291

The corresponding history result is:

Broker Sent At: 7/26/2017 12:53:48

Message: 291



Chapter 9: EtherNet / IP

In this Chapter...

9.1 EtherNet/IP Basics	Error! Bookmark not defined.
9.1.1 Implicit vs Explicit Messaging	Error! Bookmark not defined.
9.1.1 Explicit Messaging Details	Error! Bookmark not defined.
9.1.3 Implicit Messaging Details	Error! Bookmark not defined.
9.2 EtherNet/IP Adapter Setup	Error! Bookmark not defined.
9.2.1 EZRack PLC Setup	Error! Bookmark not defined.
9.2.2 Allen-Bradley Setup	Error! Bookmark not defined.
9.2.3 Troubleshooting	Error! Bookmark not defined.

9.1 EtherNet/IP Basics

EtherNet/IP is an industrial network protocol talks Common Industrial Protocol (CIP) over Ethernet. It is most often used with Allen-Bradley Rockwell devices and industrial equipment meant to interface with those devices. EtherNet/IP uses both of the most widely deployed Ethernet standards (Internet Protocol suite and IEEE 802 project) to define the features and functions of its transport, network, data link and physical layers. EtherNet/IP uses the CIP object model framework for its communication. The object-oriented design of CIP provides EtherNet/IP with the services and device profiles needed for real-time control applications and to promote consistent implementation of automation functions across a diverse ecosystem of products.

9.1.1 Implicit vs Explicit Messaging

EtherNet/IP supports two types of communication. First is the explicit messaging which where each communication is a separate query and response. This communication is inherently slower than the implicit communication because each “packet” requires overhead information about what you need from which device. On the other hand implicit communication is where a link between devices is established initially and from that point forward all specified information is exchanged at set intervals.

Each of these communication is used for different applications and purposes. The selection of explicit or implicit messaging often depends on the choice of device, as each may support only one messaging mode. If your application requires large amounts of data, explicit messaging is the preferred choice because bandwidth is saved, as data is only requested when necessary.

The table below gives a small overview of the uses, please refer to the next page for more information.

EtherNet/IP Connection	Explicit Message Description	Implicit Message Description
Initiates Connection (Master)	Client (Controller)	I/O Scanner (Controller)
Services Connection (Slave)	Server (Field Device)	I/O Adapter (Field Device)
Messaging type	Unconnected but can be connected	Connected
Typical Use	Diagnostic / Event data	Real-time I/O data (Control)

9.1.1 Explicit Messaging Details

Explicit messaging (client/server messaging) is best used for non-real time communication which is not time critical. In this type of messaging the client (PLC / Controller) requests information from the server (Field Device) and the server sends the requested information back to the client.

Since the client request the information via TCP/IP services, the message needs to include all the information so that the server can respond explicitly to the message. The client will basically say to server, "I need this information, with this specified formatting, please send it". The server then responds with a correctly formatted message with the information.

This configuring and monitoring ability works well for non-real time messaging as the client can send a message request anytime, and the server can respond when it is available. Due to this explicit messaging requires programming in the controller for setup. You need to request the data, add handshaking, acknowledge the data, and move the data where it's needed in the controller.

Explicit Messaging (TCP/IP)



EtherNet/IP Connection	Explicit Message Description	Color Code
Initiates Connection (Master)	Client (Controller)	Dark Blue
Services Connection (Slave)	Server (Field Device)	Red
Messaging type	Unconnected but can be connected	Yellow and Blue
Typical Use	Diagnostic/event/configuration data	Red and Green

9.1.3 Implicit Messaging Details

Implicit messaging (I/O Messaging) is used for time-critical applications such as real time control. Implicit messaging is called I/O messaging it is frequently used for remote I/O applications. This communication is much more efficient than explicit messaging since but the scanner and adapter are preconfigured to know exactly (implicitly) what to expect from this communication.

Implicit messaging basically copies a set amount of data with minimal other information into the message. Both the scanner and adapter don't need to be told much since they both know what to expect in the message and what to send back. The meaning of the data is "implied" so there is no extra stuff.

Also the setup of implicit messaging is simple and quick. The scanner only needs to be setup to know what data it should receive and send, and which EtherNet/IP device it needs to connect to. After that the data is transferred at the rate you specify, typically in the 5ms to 20ms range.

Implicit Messaging (UDP)



EtherNet/IP Connection	Implicit Message Description	Color Code
Initiates Connection (Master)	I/O Scanner (Controller)	Dark Blue
Services Connection (Slave)	I/O Adapter (Field Device)	Red
Messaging type	Connected	Yellow
Typical Use	Real-time I/O data (Control)	Red and Green

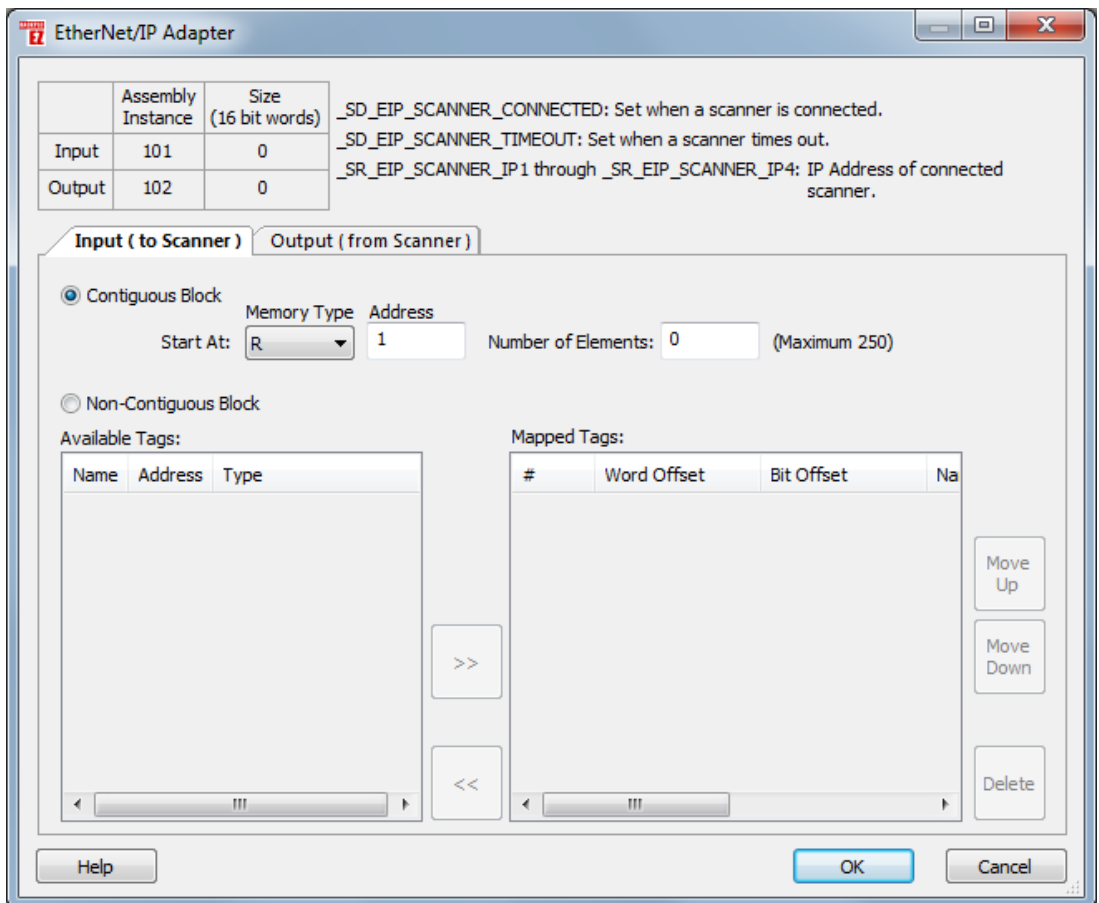
9.2 EtherNet/IP Adapter Setup

The EZRack PLC supports Ethernet/IP Adapter communication. This communication is for the EZRack to act as Adapter to external device Scanner. This section will define how to setup the EZRack PLC and how to setup an Allen-Bradley PLC to act as Scanner to the EZRack PLC.

9.2.1 EZRack PLC Setup

To setup the EZRack to communicate as an Ethernet/IP Adapter follow the direction below:

1. Go to **Setup > EtherNet/IP Adapter Setup...** You will see the following screen appear:



2. This screen allows you to setup which tags are to be used for the Adapter. You can either use a contiguous block of registers (example R1-R250) or you can select the registers you would like to send. The maximum number is restricted to 250 Input and 250 Output.

3. Important information for the setup of Scanner is the Connection Parameters of the Input / Output Assembly Instances and the Input / Output Word size. Make sure to setup this information in your Allen-Bradley PLC after you have selected all tags that you will send and receive.

System Discretes:

_SD_EIP_SCANNER_CONNECTED	SD26	Read Only	Indicates when the EtherNet/IP Adapter is connected with an EtherNet/IP Scanner.
_SD_EIP_SCANNER_TIMEOUT	SD27	Read Only	Indicates when the EtherNet/IP Adapter connection times out (after 3000 mSec)

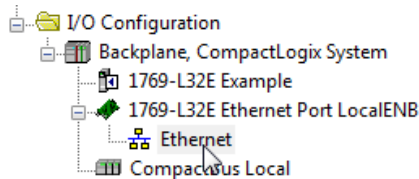
System Registers:

_SR_EIP_SCANNER_IP1	SR21	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP2	SR22	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP3	SR23	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.
_SR_EIP_SCANNER_IP4	SR24	Read Only	This will have the IP address of the EtherNet/IP Scanner which is connected to the EZ Rack PLC.

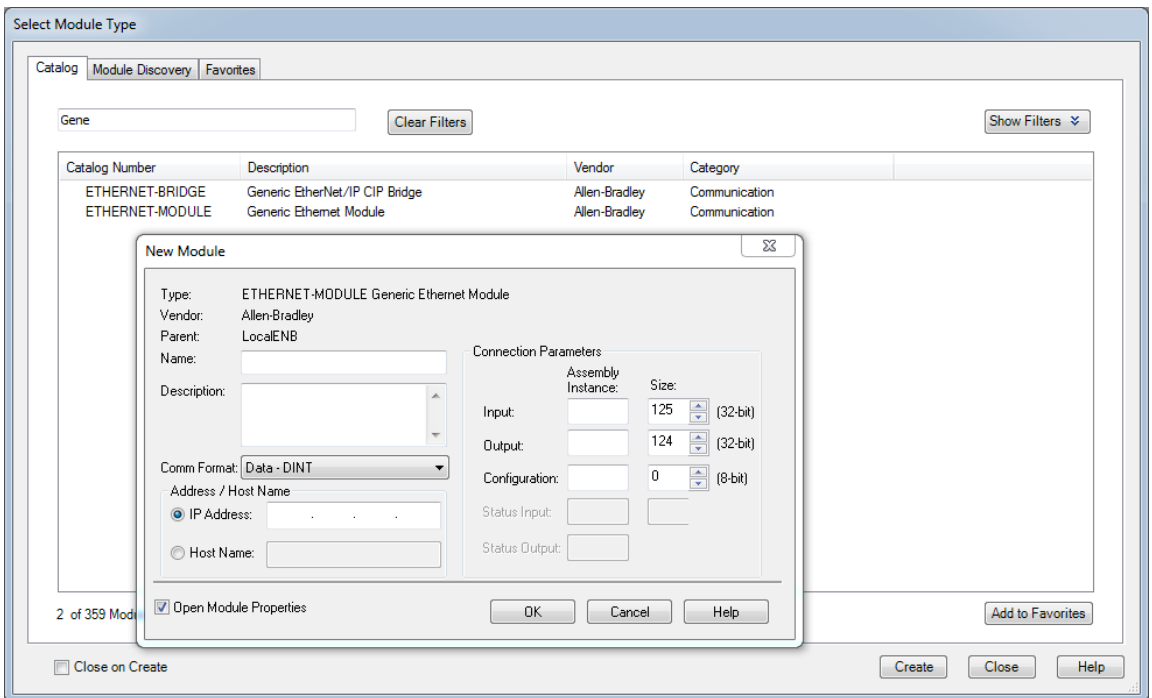
9.2.2 Allen-Bradley Setup

To setup an Allen-Bradley PLC to communicate to the EZRack PLC please use RSLogix 5000.

1. In the selected project under your Ethernet option click to Add New Module.



2. In the Select Module Type find the Generic Ethernet Module. Select this module and the following dialog will show up:



3. Add a name for your module and put in the **Comm Format as Data INT**. Also put in the IP Address of your EZRack PLC.

Vendor: Allen-Bradley
 Parent: LocalENB
 Name: EZRack_PLC
 Description:
 Comm Format: Data - INT
 Address / Host Name
 IP Address: 10 . 1 . 200 . 200

4. Copy the Connection Parameters from the EZRack PLC to the Connection Parameters of the New Module. Set the Assembly Instance Configuration to 1.

	Assembly Instance	Size (16 bit words)
Input	101	10
Output	102	10

Connection Parameters

Assembly Instance: 101 Size: 10 (16-bit)

Input: 101 Output: 102 Configuration: 1 Size: 0 (8-bit)

5. Click OK to create the module. In the Connection settings please use an RPI of 10 ms or more. A faster RPI does work but is not as reliable. You can also choose to use any of the other settings all function with the EZRack PLC.

Module Properties Report: LocalENB (ETHERNET-MODULE 1.1)

General Connection* Module Info

Requested Packet Interval (RPI): 10.0 ms (1.0 - 3200.0 ms)

Inhibit Module

Major Fault On Controller If Connection Fails While in Run Mode

Use Unicast Connection over EtherNet/IP

6. After hitting apply you have created a connection between the EZRack PLC and the Allen-Bradley PLC. As soon as both projects are transferred to their respective PLCs they will communicate together and exchange the selected tag information. *Note: Allen-Bradley tags will be in the Controller Tags area under the name of the Generic Ethernet Module.*

9.2.3 Troubleshooting

If the EZRack PLC and Allen-Bradley PLC are not communicating or having trouble communicating the Module Fault will inform you as to the problem. Please see table below for the options:

Error	Error Code (HEX)	Problem
Connection Request timed out.	0204	The Allen-Bradley PLC cannot connect to the EZRack PLC. Please make sure the IP address is correct and the EZRack PLC is on the same network as the Allen-Bradley PLC.
Invalid segment type	0315	The Connection Parameters information is incorrect. Please check and make sure both PLCs parameters match.



Chapter 10: EZRack Modules

In this Chapter...

10.1 Basic Modules	360
10.2 Specialty Modules	366
10.2.1 High Speed Counter (EZRPL-IO-HSCNT)	367
10.2.2 Resistance Temperature Detector Module (EZRPL-IO-4RTD).....	376
10.2.3 Thermocouple Modules (EZRPL-IO-4THIE)	379

10.1 Basic Modules

The EZRack PLC has many different IO modules. Most IO modules are plug and play with the only configuration needed is for the EZRack Designer Pro to be configured to communicate with them, please see *Section 2.5.8* for IO Configuration instructions. There are some modules that do need additional setup, for this please refer to the next section in this chapter. For more detailed information about each modules please see the Hardware Manual or refer to the Module Specifications that can be found online at www.EZAutomation.net. The pinouts for modules are included below for convenience.

DC Inputs / Outputs

Pin#	16DCI	16DCON	16DCOP	8DCOP-HC	Pin#
1	INPUT-1	+VS-1	+VS-1	VS+1	1
2	INPUT-2	OUTPUT-1	OUTPUT-1	OUT-1	2
3	INPUT-3	OUTPUT-2	OUTPUT-2	COM-1	3
4	INPUT-4	OUTPUT-3	OUTPUT-3	OUT-2	4
5	COM-1	OUTPUT-4	OUTPUT-4	COM-1	5
6	INPUT-5	OUTPUT-5	OUTPUT-5	VS+2	6
7	INPUT-6	OUTPUT-6	OUTPUT-6	OUT-3	7
8	INPUT-7	OUTPUT-7	OUTPUT-7	COM-2	8
9	INPUT-8	OUTPUT-8	OUTPUT-8	OUT-4	9
10	COM-2	COM-1	COM-1	COM-2	10
1	INPUT-9	+VS-2	+VS-2	VS+3	1
2	INPUT-10	OUTPUT-9	OUTPUT-9	OUT-5	2
3	INPUT-11	OUTPUT-10	OUTPUT-10	COM-3	3
4	INPUT-12	OUTPUT-11	OUTPUT-11	OUT-6	4
5	COM-3	OUTPUT-12	OUTPUT-12	COM-3	5
6	INPUT-13	OUTPUT-13	OUTPUT-13	VS+4	6
7	INPUT-14	OUTPUT-14	OUTPUT-14	OUT-7	7
8	INPUT-15	OUTPUT-15	OUTPUT-15	COM-4	8
9	INPUT-16	OUTPUT-16	OUTPUT-16	OUT-8	9
10	COM-4	COM-2	COM-2	COM-4	10

AC Inputs / Outputs

Pin#	4ACI4ACO	8ACI	8ACO	Pin#
1	4 X 2 LEDS	4 X 2 LEDS	4 X 2 LEDS	1
2				2
3				3
4				4
5				5
6				6
7				7
8				8
9				9
10				10
1	INPUT-1	INPUT-1	OUTPUT-1	1
2	INPUT-2	INPUT-2	OUTPUT-2	2
3	INPUT-3	INPUT-3	OUTPUT-3	3
4	INPUT-4	INPUT-4	OUTPUT-4	4
5	INPUT-COM	COM-1	COM-1	5
6	OUTPUT-1	INPUT-5	OUTPUT-5	6
7	OUTPUT-2	INPUT-6	OUTPUT-6	7
8	OUTPUT-3	INPUT-7	OUTPUT-7	8
9	OUTPUT-4	INPUT-8	OUTPUT-8	9
10	OUT-COM	COM-2	COM-2	10

AC/DC IO (Relay Outputs)

Pin#	4DCOP4RLO	8RLO	Pin#
1	VS+	NO-1	1
2	DCOP-1	COM-1	2
3	DCOP-2	NC-1	3
4	DCOP-3	NO-2	4
5	DCOP-4	COM-2	5
6	VS-COM	NC-2	6
7	Not Used	NO-3	7
8	NC-1	COM-3	8
9	COM-1	NO-4	9
10	NO-1	COM-4	10
1	NC-2	NO-5	1
2	COM-2	COM-5	2
3	NO-2	NC-5	3
4	NC-3	NO-6	4
5	COM-3	COM-6	5
6	NO-3	NC-6	6
7	NC-4	NO-7	7
8	COM-4	COM-7	8
9	NO-4	NO-8	9
10	Not Used	COM-8	10

Analog Inputs / Outputs

Pin#	8ANI4ANOV-16BIT	8ANI4ANOC	8ANI4ANOV	Pin#
1	OUT-1	OUT-1	OUT-1	1
2	OUT-2	OUT-2	OUT-2	2
3	OUT-3	OUT-3	OUT-3	3
4	OUT-4	OUT-4	OUT-4	4
5	COM	COM	COM	5
6	IN-1	IN-1	IN-1	6
7	IN-2	IN-2	IN-2	7
8	IN-3	IN-3	IN-3	8
9	IN-4	IN-4	IN-4	9
10	COM	COM	COM	10
1	IN-5	IN-5	IN-5	1
2	IN-6	IN-6	IN-6	2
3	IN-7	IN-7	IN-7	3
4	IN-8	IN-8	IN-8	4
5	COM	COM	COM	5
6	Not Used	Not Used	Not Used	6
7	Not Used	Not Used	Not Used	7
8	Not Used	Not Used	Not Used	8
9	VS+	VS+	VS+	9
10	VS-COM	VS-COM	VS-COM	10

Temperature Modules

Pin#	8RTD	4THIE	Pin#
1	CH-1 (+)	IN-1+	1
2	CH-1 (-)	IN-1-	2
3	COM-1	Not Used	3
4	Not Used	IN-2+	4
5	CH-2 (+)	IN-2-	5
6	CH-2 (-)	Not Used	6
7	COM-2	IN-3+	7
8	Not Used	IN-3-	8
9	Not Used	Not Used	9
10	Not Used	Not Used	10
1	CH-3 (+)	IN-4+	1
2	CH-3 (-)	IN-4-	2
3	COM-3	Not Used	3
4	Not Used	CJC- V+	4
5	CH-4 (+)	CJC-VOUT	5
6	CH-4 (-)	CJC-GND	6
7	COM-4	Not Used	7
8	Not Used	Not Used	8
9	VS+	VS+	9
10	VS-COM	VS-COM	10

Specialty Combo

Pin#	HSCNT	6DI4DO-2ANI2ANO	Pin#
1	VS+	DO-1	1
2	Not Used	DO-2	2
3	OUT-1	DO-3	3
4	Not Used	DO-4	4
5	OUT-2	COM	5
6	Not Used	DI-1	6
7	OUT-3	DI-2	7
8	Not Used	DI-3	8
9	OUT-4	DI-4	9
10	VS-COM	DI-5	10
1	CNT-1-EN	DI-6	1
2	CNT-1-RST	DI-COM	2
3	CNT-1-A	AO-1	3
4	CNT-1-B	AO-2	4
5	COM	COM	5
6	CNT-2-EN	AI-1	6
7	CNT-2-RST	AI-2	7
8	CNT-2-A	COM	8
9	CNT-2-B	VS+	9
10	COM	VS-COM	10

10.2 Specialty Modules

The EZRack PLC Designer Pro includes some modules which need more advanced setup for ease of use. Please see this section to know how to setup these modules correctly. Below there is a basic summary of the modules and their functionality.

EZRPL-IO-HSCNT

High Speed Counter Module

The HSCNT module is the module used when interfacing EZRack PLC with an encoder device. This module has two counters that count at up to 100 KHz and has settings to count 1X, 2X, or 4X Quadrature and Up or Down Counter. The module also includes a physical reset and a physical enable. Finally it has 2 outputs per counter for PLS style control.

EZRPL-IO-4RTD

Resistance Temperature Detector Module

The RTD module is one of the temperature detection modules for the EZRack PLC. This module works with the Pt100, 120 Ni, and 10 Cu / 25 Cu RTD sensor wires. This module is used at lower ranges when a more consistent temperature reading is needed. When comparing RTD to THIE, the RTD will always be more consistent with its readings at the same temperatures.

EZRPL-IO-4THIE

Thermocouple Input Module

The THIE module is one of the temperature detection modules for the EZRack PLC. This module works with the most of the different Thermocouple wires including Type J, Type K, and etc. This module is used when a larger temperature range is needed. When comparing THIE to RTD, the THIE will have a greater temperature range than the RTD.

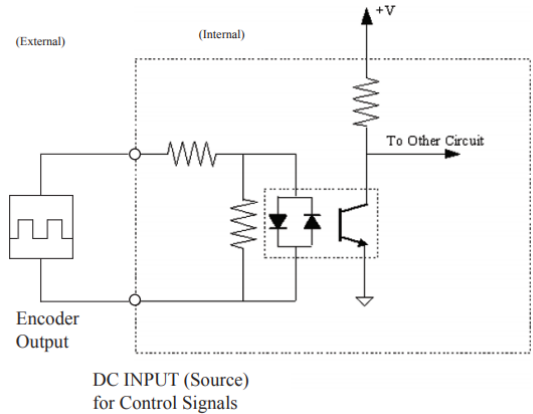
10.2.1 High Speed Counter (EZRPL-IO-HSCNT)

EZRack PLC offers High Speed counters with two 24-bit counters. The counters accept input from quadrature encoders and offer features to multiply counts by 2 or 4.

In addition, modules offer programmable set points and outputs to create high speed Programmable Limit Switch type of outputs.

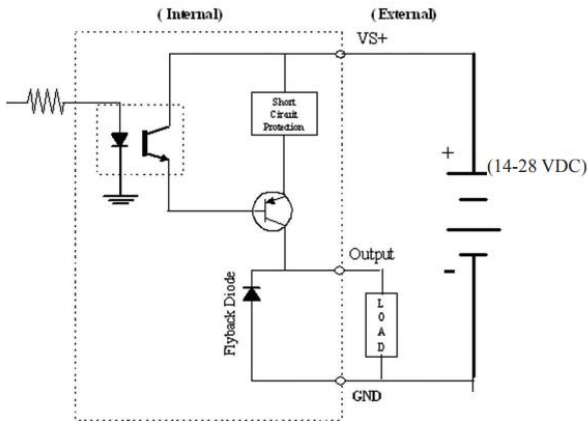
Wiring Diagram

Pinout Information			
1	VS+	11	Counter EN-1
2		12	Counter RST-1
3	Output 1	13	Counter A-1
4		14	Counter B-1
5	Output 2	15	Common
6		16	Counter EN-2
7	Output 3	17	Counter RST-2
8		18	Counter A-2
9	Output 4	19	Counter B-2
10	VS Common	20	Common



Note: Use Counter RST connection for preset signal.

Output Wiring



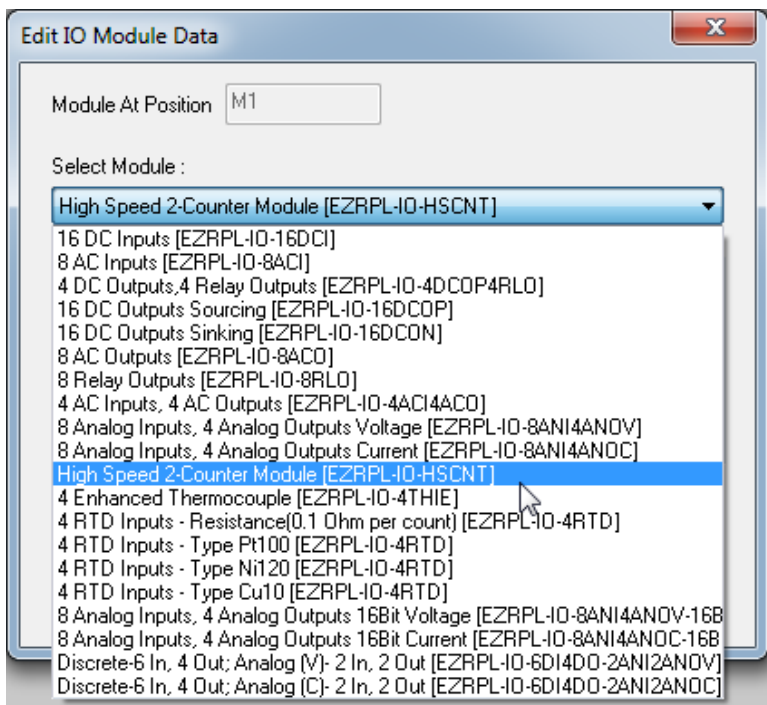
Outputs

Setpoint	Output (Wiring Diagram)	Indicator (Output Status Word)
Counter 1 Setpoint 1	Output 1	Bit 0 (LSB)
Counter 1 Setpoint 2	Output 2	Bit 1
Counter 2 Setpoint 1	Output 3	Bit 2
Counter 2 Setpoint 2	Output 4	Bit 3

HSCNT I/O Configuration

To setup the module in EZ Rack PLC, follow these steps:

1. In I/O configuration select a HSCNT module you are going to use, as shown below:



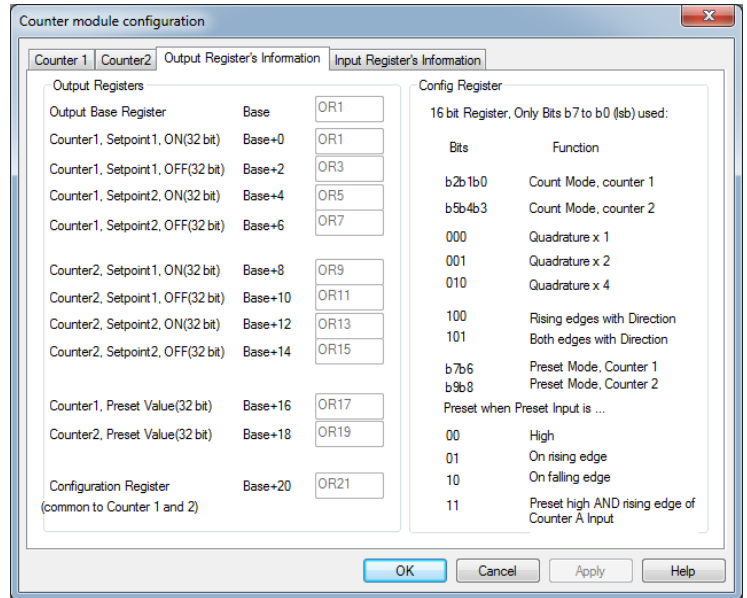
Note: You can also use the Auto Configure option.

2. The module uses 5 input registers (memory type IR). It also use 21 output registers (memory type OR). You can assign a desired starting Input Register and Output Register addresses. *Note: The maximum number of OR registers is 64 therefore up to 3 HSCNT can be used.*

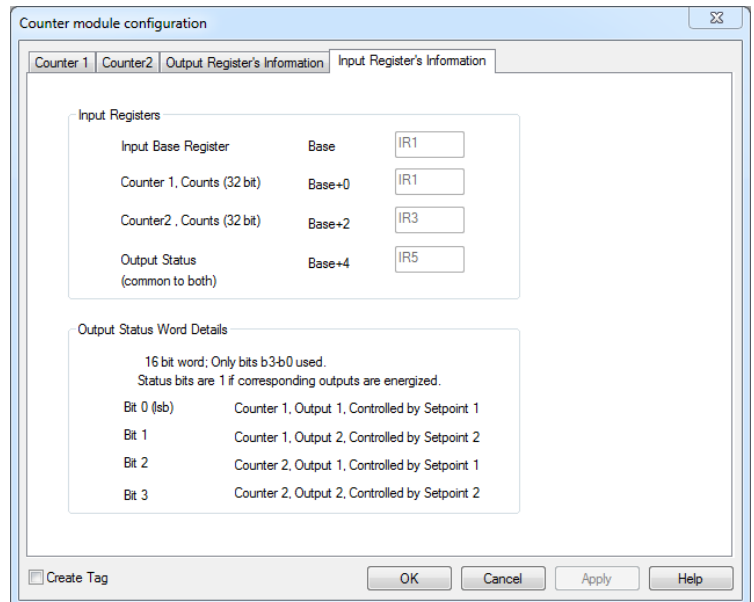
3. Next click on “Click to define setup parameters. Following dialog comes up:

4. In the Counter 1 and Counter 2 tabs you can define the Count and Preset Mode. Also you can set the set points for the Programmable Limit Switch type of outputs. And finally you can set the preset value. Note: Please see below for more information on these settings.

5. The Output Register's Information tab will tell you how to configure your module through tags. Use the corresponding tags and information to set the counter you would like to the settings you want.



6. Finally the Input Register's Information tab will tell you the functionality of the input registers.



Settings

Count Mode

The Counter Module supports 5 Counting Modes as described below. Select the desired mode. As shown in the dialog box, bits b2-b0 of the configuration register stores the count mode of the counter.

Count Mode

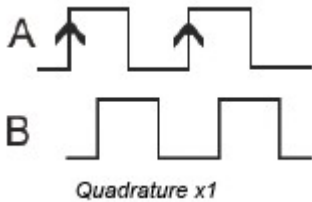
Quadrature Counting <input checked="" type="radio"/> Quadrature x1 <input type="radio"/> Quadrature x2 <input type="radio"/> Quadrature x4	PULSE and Direction Counting (Count Input on A , direction on B) <input type="radio"/> Count Rising Edges <input type="radio"/> Count Both Edges
---	---

Counter Config Register (base +20)
Bits b2 to b0 (3b) control count mode

Address Bits b2b1b0 =

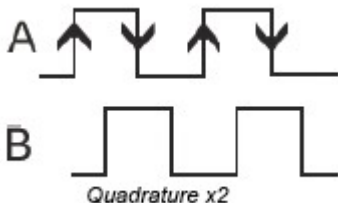
Quadrature Counting

Quadrature x1 - This mode will give 1 count for every quadrature period. Count rising edge only of signal A.



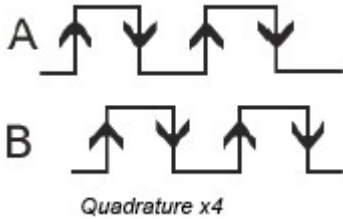
Phase relation of A & B determines the direction.

Quadrature x2 - This mode will give 2 counts for every quadrature period, giving the user twice the resolution of 1X.Count rising and falling edges of A.



Phase relation of A & B determines the direction.

Quadrature x4 - This mode will give 4 counts for every quadrature period, giving the user twice the resolution of 2X.Count both edges of A and B.



Direction is determined by the phase relation of A & B.

Pulse and Direction Counting

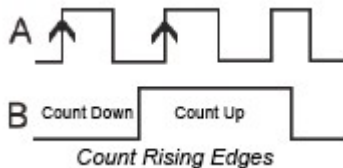
Count Rising Edges

This mode only counts Signal A. The signal from B establishes the direction.

This mode will count for the rising edge of Signal A from Encoder 1.

If direction is high, then the counter will be incremented by 1.

If direction is low, then the counter will be decremented by 1.



Count only rising edges.

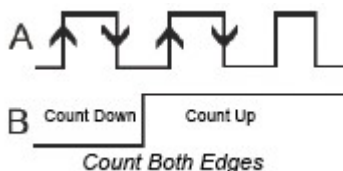
Count Both Edges

This mode only counts Signal A. The signal from B establishes the direction.

This mode will count the rising and falling edge of Signal A from Encoder 1, giving the user twice the resolution of the "Count Rising Edges" mode.

If direction is '1', then the counter will be incremented by 1.

If direction is '0', then the counter will be decremented by 1.



Count both rising and falling edges.

Set Point (1-4)

The Counter Module provides 4 programmable Limit Switch Outputs.

Set Point 1			
ON Value	<input type="text" value="0"/>	Address	<input type="text" value="OR1"/> Base+0
OFF Value	<input type="text" value="0"/>	Address	<input type="text" value="OR3"/> Base+2

Set Point 2			
ON Value	<input type="text" value="0"/>	Address	<input type="text" value="OR5"/> Base+4
OFF Value	<input type="text" value="0"/>	Address	<input type="text" value="OR7"/> Base+6

Please enter the ON & OFF values for each of the setpoints.
The dialog box shows the registers used for setpoints.
Each setpoint controls a corresponding output on the module.

E.g. Setpoint 1 controls Output 1.

Output 1 is ON when the count value is greater than or equal to the ON value, but is less than the OFF value.

Each value is a 24-bit value but takes up two 16-bit registers.

Output = ON if ON Value \leq OFF Value

Preset Value

When the preset input is triggered (see preset mode below), the value in the Value (Long) field will replace the current count of Counter 1.

The count then starts with this value. Preset is a 24 bit value, but takes up 2 16-bit registers.

Preset Value	
Address (Counter base +16)	<input type="text" value="OR17"/>
Value (Long)	<input type="text" value="0"/>

Preset Mode

As shown in the dialog box, preset mode is saved in bits b7 and b6 of the Counter configuration register. The preset mode can be set differently for both counters and each counter has its own preset/reset connection pin. Settings exist for both Counter 1 and Counter 2. Counter 1 uses bits 6, 7 and counter 2 uses bits 8, 9 of the config register. *Note: Use Counter RST connection for preset signal.*

Preset Mode

Load Preset value when Preset Input is...

High
 On rising edge
 On falling edge
 Preset High AND rising edge of Counter 1 A input

Counter config register (base+20)
Bits b7 to b6 (lsb) control preset mode

Bits b7b6 =

Preset Mode

Load Preset value when Preset Input is...

High
 On rising edge
 On falling edge
 Preset High AND rising edge of Counter 2 A input

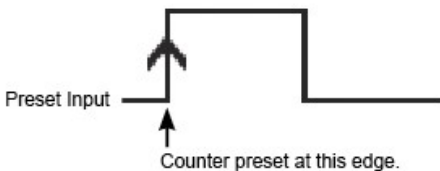
Counter config register (base+20)
Bits b9 to b8 control preset mode

Bits b9b8 =

High

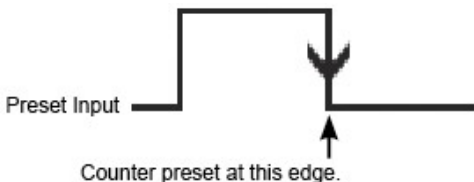
This option will set the counter to the preset value while being held high. While the preset signal is high, no new count signals will be counted.

On Rising Edge



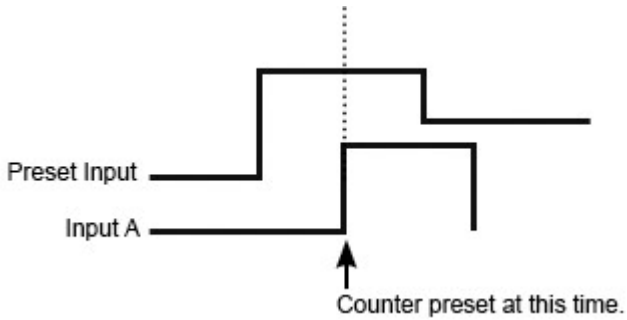
This option will preset on the rising edge of the preset signal.

On Falling Edge



This option will wait for the falling edge of the preset input to trigger a preset pulse.

Preset High AND Counter 1 A Input



This option triggers a preset pulse every time that there is a rising edge Signal A and the preset signal is high. The count stays at preset till the preset signal goes low.

10.2.2 Resistance Temperature Detector Module (EZRPL-IO-4RTD)

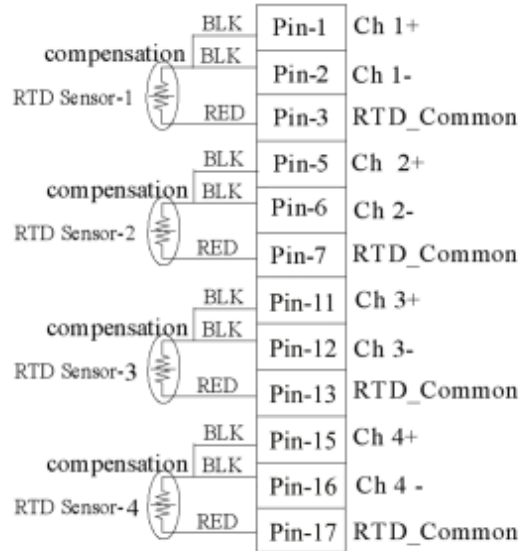
EZRack PLC IO family offers a four point RTD Input Module for connecting to popular RTDs (Resistance Temperature Detectors).

The module has following features:

1. 4 Differential Inputs that support PT100, Ni120 & Cu10 RTD's or direct resistance measurement.
2. Common mode rejection 100 Db minimum.
3. Accuracy $\pm 15\text{ppm}/^{\circ}\text{C}$
4. Resolution $\pm 0.1^{\circ}\text{C}$ (0.1 ohm per count for resistance measurement)

Wiring

Pinout Information			
1	CHAN1 +	11	CHAN3 +
2	CHAN1 -	12	CHAN3 -
3	COM-1	13	COM-3
4	Not Connected	14	Not Connected
5	CHAN2 +	15	CHAN4 +
6	CHAN2 -	16	CHAN4 -
7	COM-2	17	COM-4
8	Not Connected	18	Not Connected
9	Not Connected	19	VS +
10	Not Connected	20	VS - COM

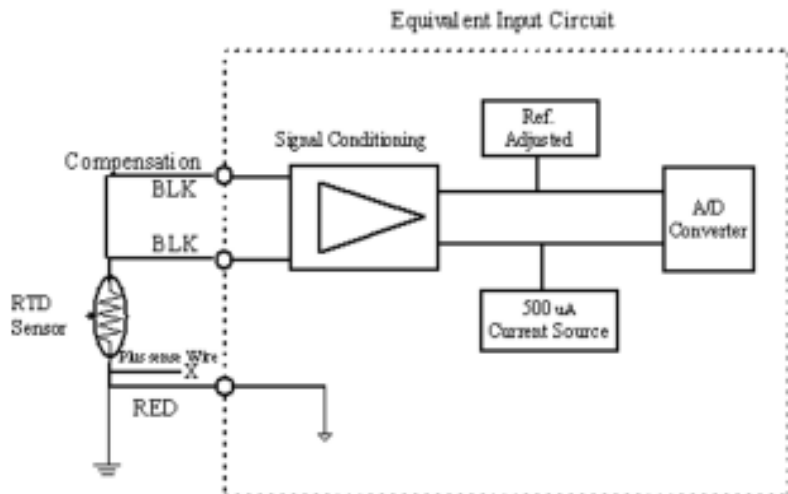


Important Note:

Keep the wires of same type and length.

Shield and drain wire cannot be used for third connection.

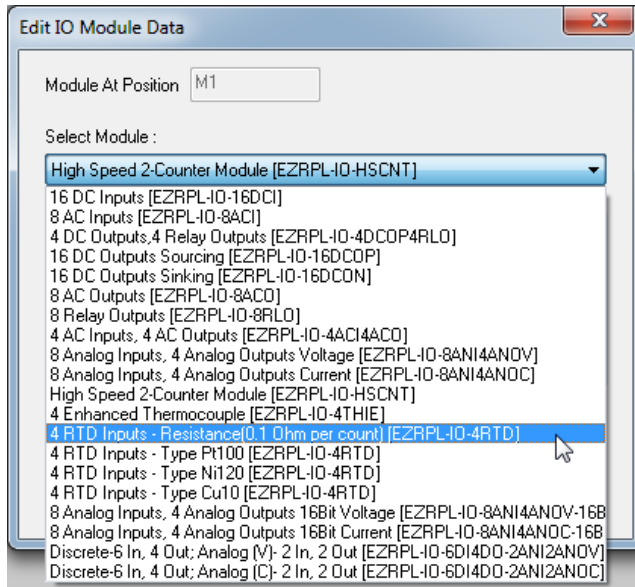
In case of four-wire RTD type, extra plus Sense wire should not be connected.



RTD I/O Configuration

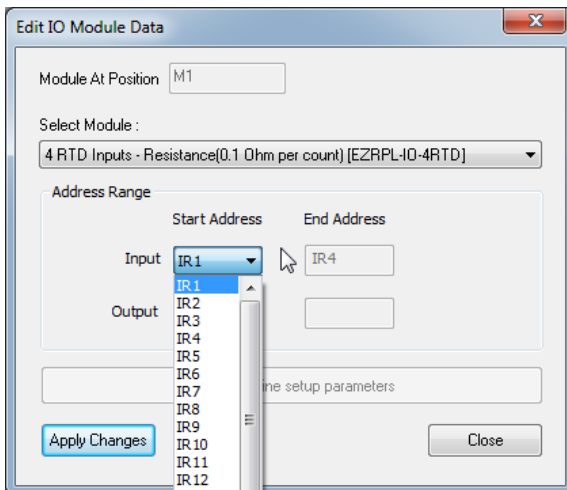
To setup the RTD module in EZRack PLC, follow these steps:

1. In I/O configuration select a 4 RTD Inputs module (EZIO-4RTD) as per the type of sensor (Pt100/Ni120/Cu10, or Resistance if you want to read resistance in ohms) you are going to use, as shown below:



Note: You can also use the Auto Configure option and then select the correct wanted RTD module.

2. The module uses 4 input registers (memory type IR). Assign a desired starting Input Register addresses:



The starting Input Register is used for Input # 1, the next for Input #2, etc. For example if the starting input address is IR1, then reading from RTD at input # 1 would be returned in IR1, from #2 in IR2, from #3 in IR3, and from #4 in IR4.

3. Click on Apply Changes button and close the Add/Edit I/O Module window.

Tag Data Types

In EZRack PLC the ladder logic accesses memory using tags. Therefore create tags for each of the Input Register (IR) associated with the module. Each tag has a data type to interpret the data values appropriately.

Please follow below given guidelines for the tag data types for the module:

Tags associated with the Input Registers (IR) must be of SIGNED_INT_16 for the RTD modules.

RTD Type selected	Use Tag data type
Pt100, Ni120, Cu10	SIGNED_INT_16
Resistance	UNSIGNED_INT_16

Temperature Values

The thermocouple module provides temperature values with one digit after the decimal point. However these values are presented as whole integer numbers, not as floating point or real numbers. The decimal point is implied. Thus if a reading from the module is 1234, it should be interpreted as 123.4, or alternately if the temperature is 234.5 degrees (C or F or K), the input register would return a value of 2345.

10.2.3 Thermocouple Modules (EZRPL-IO-4THIE)

EZRack PLC IO family offers a Thermocouple Module which can work with any type Thermocouples after setup. Please see below for information on setup.

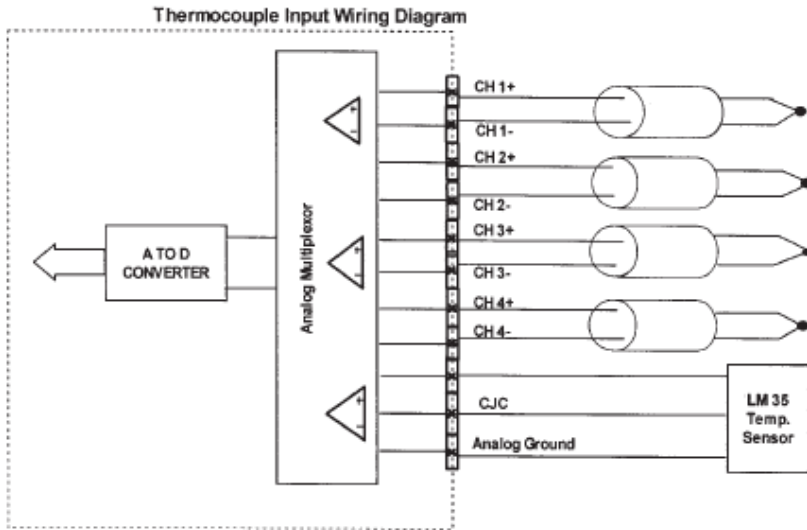
The module has following features:

1. Up to 4 thermocouple inputs with user selectable thermocouple types on each input
2. User programmable broken thermocouple detection
3. User programmable units for temperature -- Centigrade, Fahrenheit or Kelvin
4. Automatic Cold Junction Compensation (CJC) with Internal or External Sensor

Wiring

Pinout Information			
1	Input 1+	11	Input 4+
2	Input 1-	12	Input 4-
3	Not Connected	13	Not Connected
4	Input 2+	14	CJC V+
5	Input 2-	15	CJC VOut
6	Not Connected	16	CJC Ground
7	Input 3+	17	
8	Input 3-	18	
9	Not Connected	19	VS+
10	Not Connected	20	VS Common

Note: Only 3 thermocouple inputs are available when External CJC is used.



Which CJC Sensor, internal or external, to use?

The module provides automatic cold junction compensation (CJC). It can use internal or external temperature sensor for this purpose. With both sensor types, the module provides very repeatable temperature readings. You may select internal or external based on below given guidelines:

Use Internal CJC Sensor: If you need repeatable but not very accurate readings (typical +/- 6 deg C). This allows you to use 4 thermocouples with the module. Internal sensor is automatically used if external sensor is not selected for input #4.

Use External CJC Sensor: If you need repeatable as well as more accurate (max +/- 3 deg C) readings. You will need to use LM19 temperature sensor, and you can use only 3 thermocouples. *Note: To use External Sensor, select "CJC Sensor (LM19)" for Input #4 type, in the configuration dialog box. (see below).*

Type of Thermocouple & ranges supported, and error values:

Thermocouple type	Range			Value reported with Open Thermocouple or incorrect configuration if option selected is:		
	Centigrade	Fahrenheit	Kelvin	Do not report	Low Value	High Value
J	-210°C to +1200°C	-346°F to +2192°F	63°K to 1473°K	Random	-32768 (0x8000)	32767 (0x7FFF)
K	-200°C to +1372°C	-328°F to +2502°F	73°K to 1645°K	Random	-32768 (0x8000)	32767 (0x7FFF)
S	-50°C to +1768°C	-58°F to +3214°F	223°K to 2041°K	Random	-32768 (0x8000)	32767 (0x7FFF)
T	-200°C to +400°C	-328°F to +752°F	73°K to 673°K	Random	-32768 (0x8000)	32767 (0x7FFF)
E	-200°C to +980°C	-328°F to +1796°F	73°K to 1253°K	Random	-32768 (0x8000)	32767 (0x7FFF)
R	-50°C to +1768°C	-58°F to +3214°F	223°K to 2041°K	Random	-32768 (0x8000)	32767 (0x7FFF)
B	250°C to +1820°C	482°F to +3308°F	523°K to 2093°K	Random	0 (0x0000)	65535 (0xFFFF)
N	-200°C to +1300°C	-328°F to +2372°F	73°K to 1573°K	Random	-32768 (0x8000)	32767 (0x7FFF)
Ambient Temp	This selection would read module's ambient temperature					

Thermocouple Module Setup

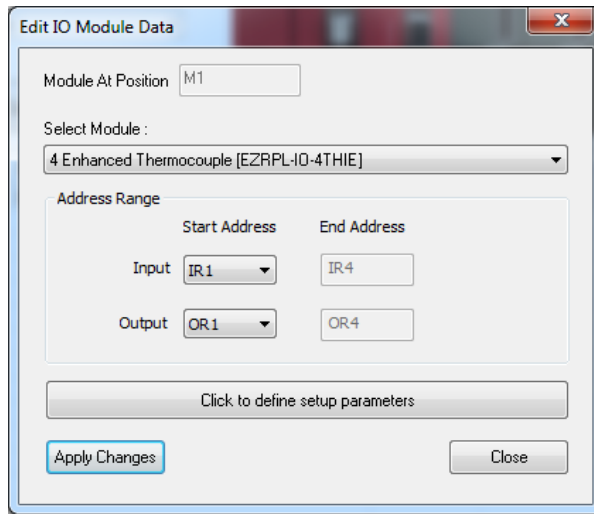
To setup the module in the EZ Rack PLC, follow these steps:

1. In I/O configuration select "4 Enhanced Thermocouple Module (EZLGX-IO-4THIE) as shown below, and assign desired starting Input and starting output addresses:

The module takes up 4 input registers (IRs) and 4 output registers (ORs):

- Input Registers return thermocouple readings.
- Output registers are used for configuration of respective thermocouple.

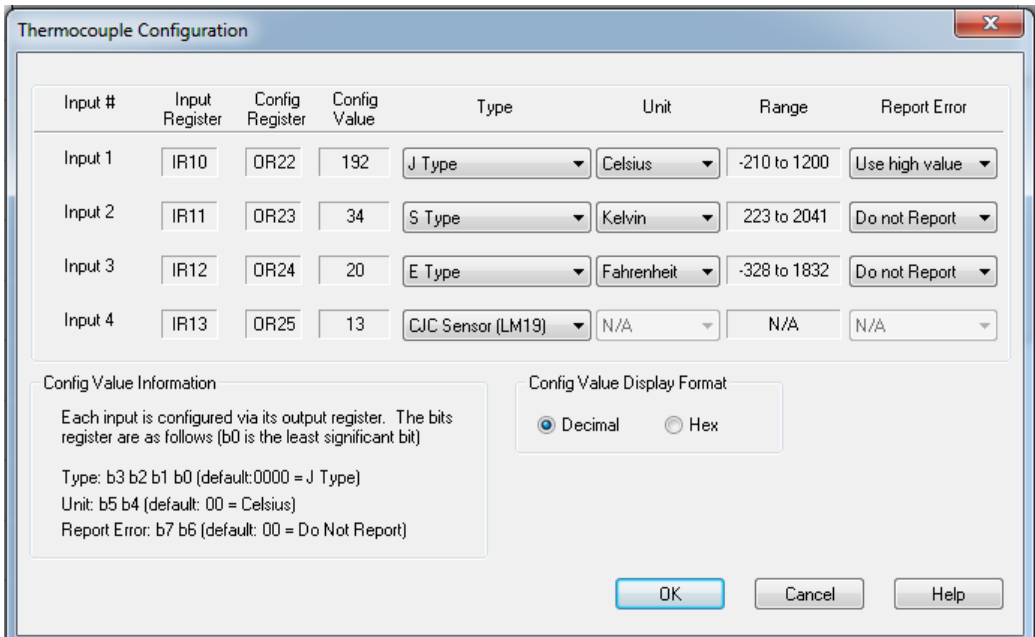
The starting Input Register is used for Input # 1, the next for Input #2, etc. For example if the starting input address is IR1, then reading from thermocouple at input #1 would be returned in IR1, from #2 in IR2, from #3 in IR3, and from #4 in IR4. Similarly Starting OR would configure Thermocouple #1, and next OR would configure Input #2, and so on.



The dialog box is titled "Edit IO Module Data". It contains the following fields and controls:

- Module At Position:
- Select Module:
- Address Range section:
 - Start Address:
 - End Address:
 - Output:
 - Output:
- Buttons: "Click to define setup parameters", "Apply Changes", and "Close".

2. Click on "Click to define setup parameters. Following dialog comes up:



The dialog box is titled "Thermocouple Configuration". It features a table with columns: Input #, Input Register, Config Register, Config Value, Type, Unit, Range, and Report Error. Below the table is a "Config Value Information" section and a "Config Value Display Format" section with radio buttons for "Decimal" and "Hex".

Input #	Input Register	Config Register	Config Value	Type	Unit	Range	Report Error
Input 1	IR10	OR22	192	J Type	Celsius	-210 to 1200	Use high value
Input 2	IR11	OR23	34	S Type	Kelvin	223 to 2041	Do not Report
Input 3	IR12	OR24	20	E Type	Fahrenheit	-328 to 1832	Do not Report
Input 4	IR13	OR25	13	CJC Sensor (LM19)	N/A	N/A	N/A

Config Value Information
 Each input is configured via its output register. The bits register are as follows: (b0 is the least significant bit)
 Type: b3 b2 b1 b0 (default: 0000 = J Type)
 Unit: b5 b4 (default: 00 = Celsius)
 Report Error: b7 b6 (default: 00 = Do Not Report)

Config Value Display Format
 Decimal Hex

Buttons: "OK", "Cancel", "Help".

For each input, select Type of Thermocouple, Unit of measurement, and Option to report error (such as due to open thermocouple, wrong configuration, etc.). These choices create a config value that is written to corresponding OR registers when the program is written to EZRack PLC. These values can also be written or modified using ladder logic, but it may be easier to use the dialog box. That is all required to setup the module.

Note: If CJC Sensor (LM19) is selected for Input 4 then only 3 thermocouple inputs are available (Input 1-3).

The programmable parameters, namely type, unit, and report error, as well as the computed config value, are described below:

Type
 Select the type of the thermocouple using this field. The possible choices are: J Type, K Type, S Type, T Type, E Type, R Type, B Type, N Type, Ambient Type. The display-only range field depends on the type of the thermocouple selected (along with the unit).

For Input #4 ONLY: If using external sensor for CJC, select “CJC Sensor (LM19)” for Input #4.

Unit
 Select the unit for measurement for each thermocouple. The possible choices are: Celsius, Fahrenheit and Kelvin. The display-only range field depends on the unit selected (along with the type of the thermocouple).

Report Error
 The Report Error function on the thermocouple module provides diagnostic capabilities to detect open or burnt thermocouple, or incorrect configuration (which can happen if the ladder logic writes an incorrect value to the config register). The following table describes the choices and the resulting actions. You can use these values in ladder logic to detect possible problems with the thermocouples or configurations.

Choice	Value Returned	
	All types EXCEPT B Type	B Type
Do Not Report	Indeterminate	Indeterminate
Use low value	-32768	0
Use high value	32767	65535

Config Value

The computed Config Value (determined by above choices) is the value written to the config (output) register. Each thermocouple input is configured via its config register. The Config Value Display Format option allows you to display (in this dialog box) the config value in either decimal or hex. The actual config value depends on the selections made for the type, unit and report error as shown in the following table:

Bits in Config Registers	Determined by	Default value
Bit 3 – Bit 0	Thermocouple type	0000 (J -type)
Bit 5 – Bit 4	Unit Selection	00 (Celsius)
Bit 7 – Bit 6	Report-Error Selection	00 (Do not report)

The values of bits for various selections can be seen from the configuration dialog box.

Example:

	Thermocouple Type	Unit Selection	Report-Error Selection	Final Value
Selected Option	S Type	Fahrenheit	Use high value	210
Binary Result	0010	01	11	0000 0000 1101 0010

Thermocouple Module Operation

The Thermocouple provides 4 inputs for thermocouples. Each input can independently be configured by writing to corresponding configuration register. The configuration can be done using the dialog box as described above. Alternately the configuration can be done by writing appropriate values in corresponding Output Registers of the module using ladder logic.

Tag Data Types

The ladder logic in EZRack PLC accesses PLC memory using tag names. Therefore create tags for each of the Input (IR) and output registers (OR), if used, associated with the module. Each tag has a data type to interpret the data values appropriately. Please follow below given guidelines for the tag data types for the module:

Tags associated with the Output Registers (OR) must be declared as UNSIGNED_INT_16

Tags associated with the Input Registers (IR) must be of SIGNED_INT_16 for all thermocouple types EXCEPT type B in which case it should be UNSIGNED_INT_16.

Interpreting Temperature Values

The thermocouple module provides temperature values with one digit after the decimal point. However these values are presented as whole integer numbers, not as floating point or real numbers. The decimal point is implied.

Thus if a reading from the module is 1234, it should be interpreted as 123.4, or alternately if the temperature is 234.5 degrees (C or F or K), the input register would return a value of 2345.

Open Thermocouple detection

Thermocouple module can detect open or burnt thermocouples. On detecting an open/burnt thermocouple, the module provides a lowest or highest possible value in corresponding Input Register. The choice of low or high value is user programmable. See configuration dialog box, and specification to see the values returned for each thermocouple type.

External CJC Sensor Diagnostics

The thermocouple module provides automatic cold junction compensation. User has an option to use internal (default) or external temperature sensor for this purpose. If external sensor is not selected in Module setup (see setup section), the internal option is automatically used.

The wiring of the 3-terminal external sensor is shown in wiring section above. The reading in input-register corresponding to input #4 can be used to detect incorrect wiring of the sensor. If the reading is above 100 or below 0, the sensor may not be wired correctly.



Chapter 11: Sparkplug B (Ignition)

In this Chapter...

- 11.1 Sparkplug B IIoT (MQTT) Basic Setup..... 387
- 11.2 Basic Ignition MQTT Modules Setup..... 393
- 11.3 Advanced Sparkplug Setup (Security and Encryption)..... 394
 - 11.3.1 Encryption and Certificate Basics..... 394
 - 11.3.2 EZRack PLC Encryption and Certificate Authority Setup:..... 396
 - 11.3.3 Ignition Encryption and Keystore Setup 397
- 11.4 Redundancy Setup (EZRack PLC and Ignition): 398
- 11.5 Store and Forward Setup: 399
 - 11.5.1 Store and Forward Time Zone Setup 400
- 11.6 Troubleshooting Sparkplug B Setup: 402

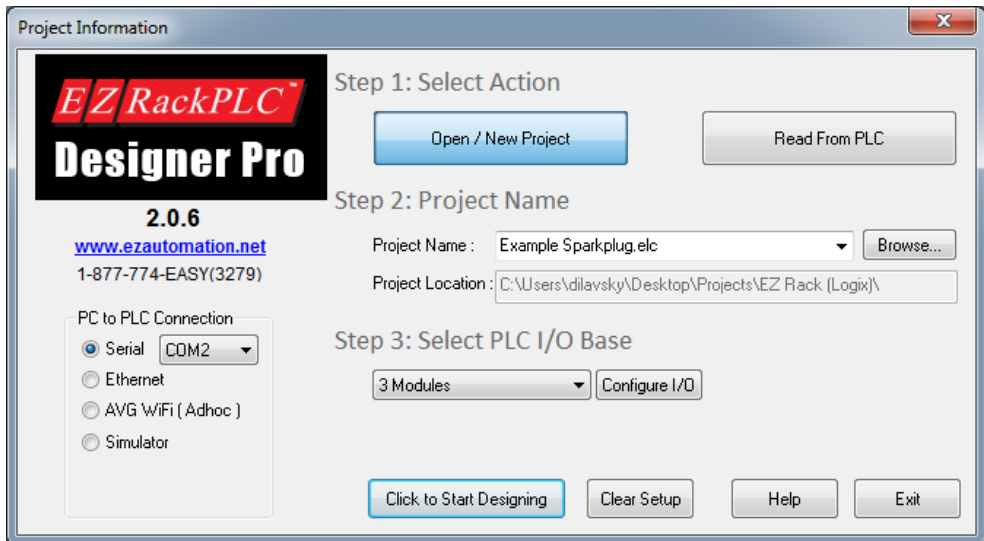
11.1 Sparkplug B IIoT (MQTT) Basic Setup

This setup guide is for using the EZRack PLC with Inductive Automation’s Ignition platform. Using this guide, the EZRack PLC will be setup to communicate over MQTT to the Ignition platform using the Sparkplug B specification. This allows the EZRack PLC’s tag to automatically show up in the Ignition software. Please follow the instructions below to create a basic project that will communicate to Ignition.

Note: This MQTT communication is for use with the Inductive Automation’s Ignition Platform with the Cirrus-Link IIoT Modules. For more information please visit <https://inductiveautomation.com/> and <http://www.cirrus-link.com/>. There is a quick setup guide for the needed Ignition modules included at the end.

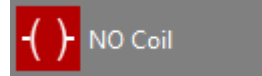
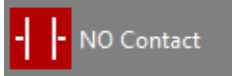
EZRack Sparkplug B Setup

1. Launch the EZRack PLC Designer Pro version 2.1.0 or higher. *Note: You will need EZRack PLC firmware version A.0.297 or higher.*
2. On the Project Information screen click the button option **“Open / New Project”**.

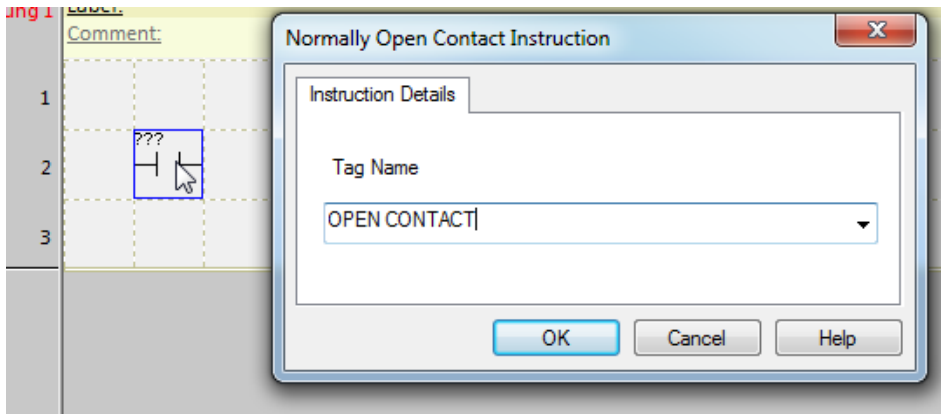


3. Select a project or enter a project name (new project names will create a new project). You can also configure your I/O if you would like, but it is not needed. Then click the button **“Click to Start Designing”**.

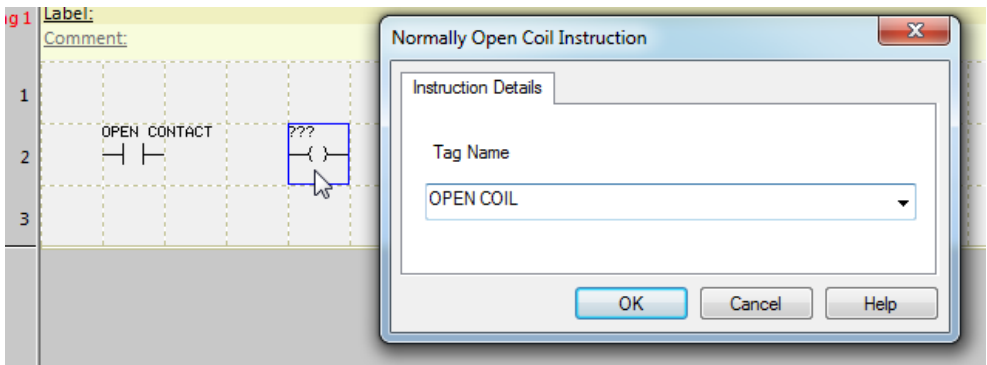
4. Once the project is open, we will add a contact and coil. In the right side bar click on the Normally Open Contact and place it down anywhere in the Main Logic area. Next select a Normally Open Coil and place it down as well.



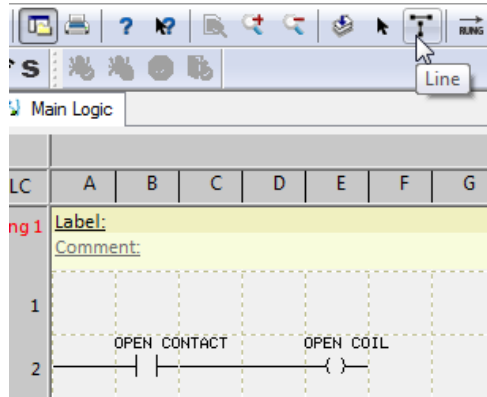
5. Next double click on the NO Contact and in the next dialogue enter a tag name, for example OPEN CONTACT. Then click OK.



6. Next do the same for the NO Coil, giving it a different name, for example OPEN COIL.



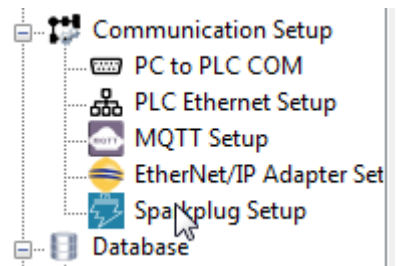
7. Finally make sure that the contacts are connected. If need be use the line option (in top toolbar) to connect the left side to the NO Contact and the NO Coil. You can also double click in the ladder logic. *Note: You do not need to connect the line all the way to the right side power rail.*



8. Now you will need to setup the sparkplug to communicate to the broker. Please make sure you have the following information from the Ignition Broker.

Information Type	Example Information
User Name to Connect to Broker	admin
Password to Connect to Broker	changeme
Domain Name (Server URI) or IP Address of Broker	10.1.200.12
Port Number (TLS or non-TLS)	1883 (TLS 8883)

9. Next open the Sparkplug Setup. It is on the Navigation window / project tree on the left side. It is also under **Setup > Sparkplug Setup**
10. In the Sparkplug dialogue box please enter a Client ID. *Note: The Primary Host ID is used for Redundant Ignition Environment, please see the section on Redundancy for more information.*



11. Next you can change the Group ID, Node ID, and Device ID. The final options in the first tab, if selected, will have the PLC provide some basic info about the CPU and USB to the Ignition platform.

12. In the Brokers tab please enter the IP address of the Broker or use the Domain Name Lookup to find the IP address. For the Domain Name Lookup just enter the Domain Name and it will input the needed IP address.

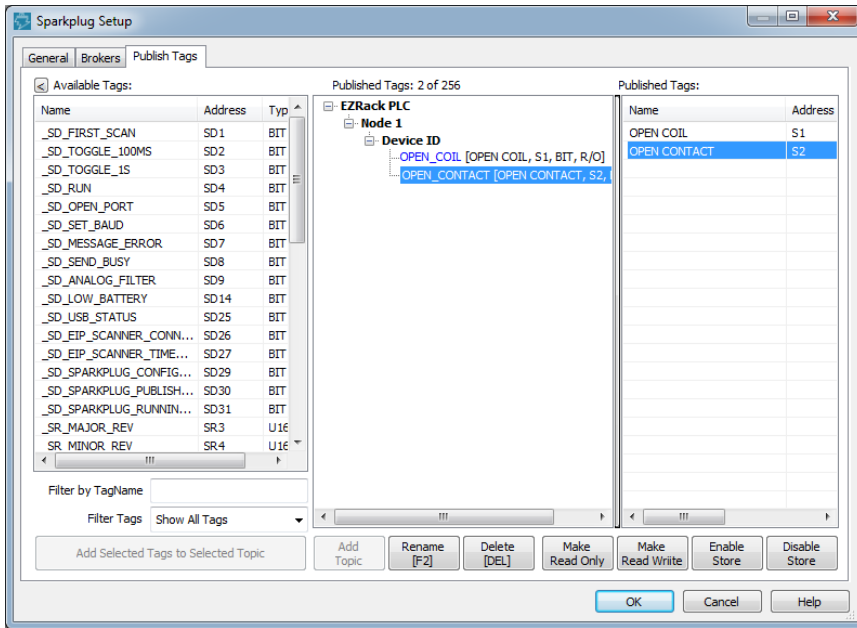
13. Please enter the Port and set the Keep Alive Interval. Finally enter the Username and Password for the Broker.

IP	Port	KeepAlive	UserName	Password	TLS	Check Host	Broker Name	CA Local File Path
10.1.200.12	1883	60	admin	changeme	None	Yes		

Note: Please refer to the security section to setup TLS security.

Note: The EZRack supports up to 4 different brokers. Please see the redundancy section for more information.

14. Finally in the Publish Tags section add the Open Contact and Open Coil tags. Then select the Open Contact and make it Read/Write. Please see below for button descriptions.

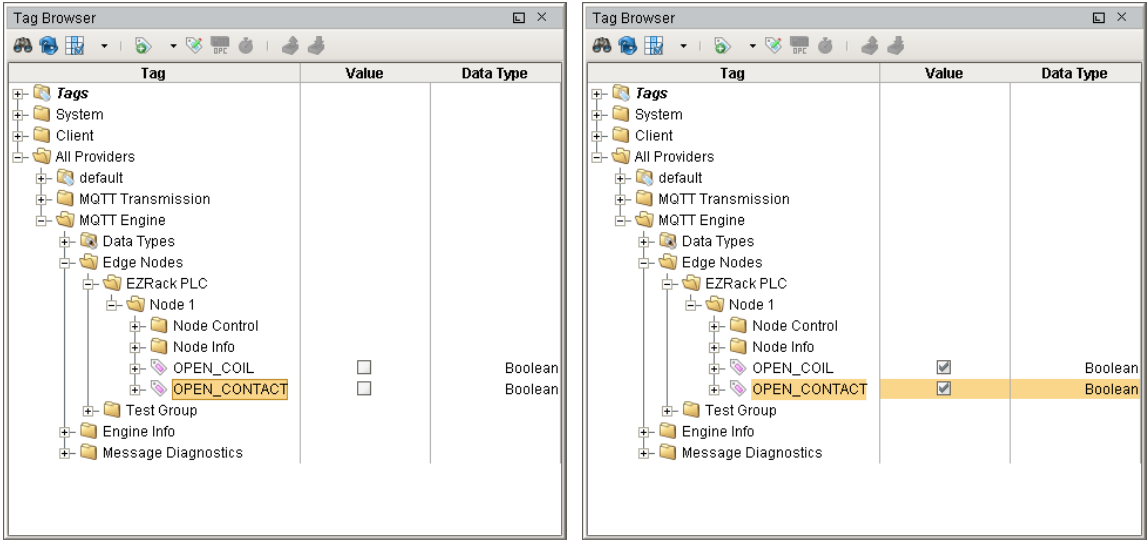


Note: You can also set the tag to be stored in case of Broker connection loss. Please see the Store and Forward section for more information.

Button descriptions:

- Add Topic:** Use this to add a topic from the Available Tags list (Table on left). If no tag select then can add a folder (topic).
 - Rename Topic / Metric:** Can rename either a topic or even a tag. Note: The name change does not affect the tag database name of the tag.
 - Remove Topic / Tags:** Removes selected topic or tag. If topic removed, it will also remove any subtopics and tags in the topic.
 - Make Tag(s) Read Only:** Makes the tag read only, this means that cannot use external means to change this tag.
 - Make Tag(s) Read Write:** Allows tags to be changed externally. For example Ignition could now use the open coil tag as a button, instead of just an indicator.
 - Enable Tag(s) Store:** Allows redundancy to work with that tag.
 - Disable Tag(s) Store:** Disables redundancy to work with that tag.
15. Now go to **File > Transfer to PLC**. Please make sure you are connected to the PLC and have the correct PC to PLC Connection.

16. Once the project has been transferred and the PLC is running you will now see the tags in your Ignition Designer software in the tag browser. After enabling Read/Write you will be able to use the OPEN_CONTACT to turn ON/OFF the OPEN_COIL.



Note: For basic Ignition setup instructions you can see the next section of this Getting Started Guide or you can visit <https://inductiveautomation.com/> and <http://www.cirrus-link.com/> for more information.

For more help on creating logic in the EZ Rack Designer Pro and other functionality please refer to the Software Manual or the Software Help File.

11.2 Basic Ignition MQTT Modules Setup

This section briefly explores how to add to Ignition the Cirrus-Link MQTT modules and where to find the needed information to setup the EZRack PLC. Please visit

<https://inductiveautomation.com/> and <http://www.cirrus-link.com/> for more information.

Cirrus Link Needed Modules

Please download the following modules from

<https://inductiveautomation.com/downloads/ignition.>

- MQTT Distributor Module
- MQTT Engine Module
- MQTT Transmission Module

Adding Needed Modules

1. Go to localhost:8088 and Log into your Ignition Software. *Note: This assumes you have already setup the Ignition platform.*
2. Go to the Configure tab and select **System > Modules**.
3. Scroll all the way down to the bottom of the page and click Install or Upgrade Module.
4. Select the wanted module and click Install. The module will be installed and Ignition will restart.
5. Repeat for all 3 needed modules.

Cirrus-Link Guide to Using the 3 Modules

<https://docs.chariot.io/display/CLD/Getting+Started%253A+Single+Ignition+Architecture>

EZRack PLC Information

The above Cirrus-Link Guides explains how to use the 3 Modules and make changes to their setup. If no changes are made then the default information needed to use with the EZRack PLC is listed below. This information is changed and modified in the MQTT distributor Module. Note: If you do modify it then both the MQTT Engine and MQTT Transmission will also need to be modified.

Information Type	Example Information
User Name to Connect to Broker	admin
Password to Connect to Broker	changeme
Domain Name (Server URI) or IP Address of Broker	IP address of the computer with Ignition Running
Port Number (TLS or non-TLS)	1883 (TLS 8883)

11.3 Advanced Sparkplug Setup (Security and Encryption)

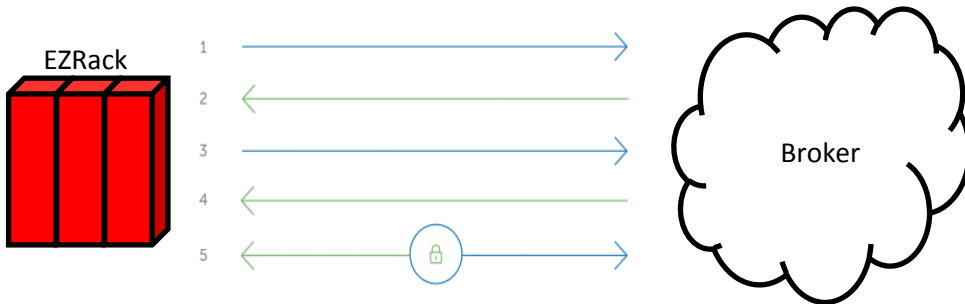
The EZRack PLC supports Encryption and Certificate security. Encryption is used to make sure that no one can listen to the data the EZRack PLC is sending out to the Broker. The certificate authentication is used to make sure that the Broker is the one you want. The EZRack PLC will also support client certificate authentication in the future, where the Broker will be able to tell that the EZRack PLC is the correct Client.

11.3.1 Encryption and Certificate Basics

When the EZRack PLC attempts to communicate with a Broker over a secure SSL connection, the PLC and the Broker establish the SSL connection using a process called an “SSL Handshake” (see diagram below). Note that the SSL Handshake is invisible to the user and happens instantaneously.

Essentially, three keys are used to set up the SSL connection: the public, private, and session keys. Anything encrypted with the public key can only be decrypted with the private key, and vice versa.

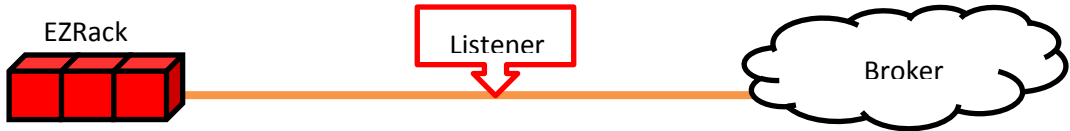
Because encrypting and decrypting with private and public key take a lot of processing power, they are only used during the SSL Handshake to create a symmetric session key. After the secure connection is made, the session key is used to encrypt all transmitted data.



When using security the process is as follows:

1. The EZRack PLC contacts the Broker and requests it identifies itself.
2. The Broker sends a copy of its SSL Certificate including the Broker’s public key.
3. The EZRack PLC checks the certificate against a list of trusted CAs and that the certificate is unexpired. If the EZRack PLC trusts the certificate, it creates, encrypts, and sends back a symmetric session key using the Broker’s public key.
4. The Broker decrypts the symmetric session key using its private key and sends back an acknowledgement encrypted with the session key to start the encrypted session.
5. The EZRack PLC and Broker now encrypt all transmitted data with the session key.

Example of how keys work and protect from spying on your connection:



Diffie–Hellman Key Exchange is the concept that explains how keys work without a Listener being able to decrypt the message. The following examples are commonly used to explain this concept.

Cryptographic Example:

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p , where p is prime, and g is a primitive root modulo p . These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to $p - 1$.

EZRack PLC and Broke agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).

EZRack PLC chooses a secret integer $a = 4$, then sends Broke $A = g^a \bmod p$
 $A = 5^4 \bmod 23 = 4$

Broke chooses a secret integer $b = 3$, then sends EZRack PLC $B = g^b \bmod p$
 $B = 5^3 \bmod 23 = 10$

EZRack PLC computes $s = B^a \bmod p$
 $s = 10^4 \bmod 23 = 18$

Broke computes $s = A^b \bmod p$
 $s = 4^3 \bmod 23 = 18$

EZRack PLC and Broke now share a secret (the number 18).

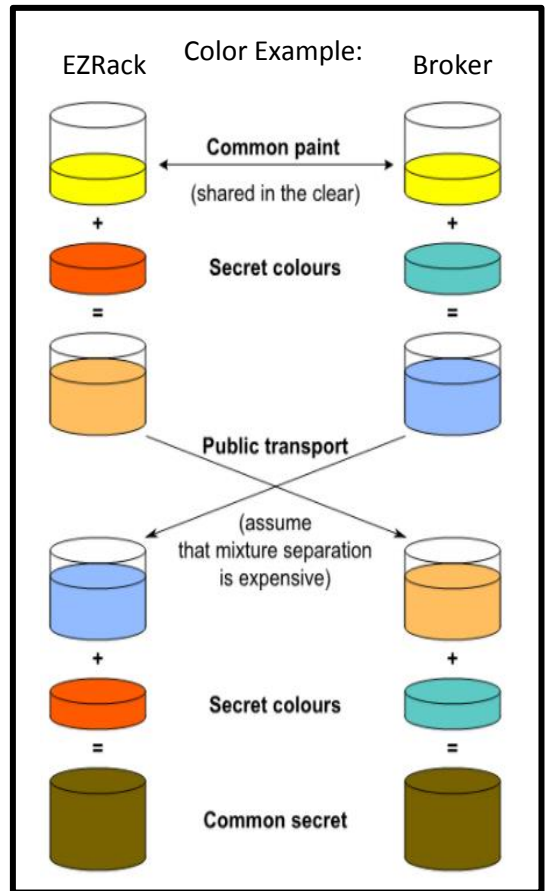
Both EZRack PLC and Broke have arrived at the same value s , because, under mod p ,

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

More specifically,

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

Note that only a , b , and $(g^{ab} \bmod p = g^{ba} \bmod p)$ are kept secret. All the other values – p , g , $g^a \bmod p$, and $g^b \bmod p$ – are sent in the clear. Once EZRack PLC and Broke compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.



11.3.2 EZRack PLC Encryption and Certificate Authority Setup:

To setup the EZRack to use encryption follow the directions below:

1. To setup encryption you will need to go to the Sparkplug Setup (**Setup > Sparkplug Setup...**) and then to the Brokers tab.

Sparkplug Setup

General Brokers Publish Tags

Update Selected Broker

Broker IP 10 . 1 . 200 . 12 Domain Name Lookup

Port Number 1883 (Default: 1883)

Keep Alive Interval 60 Seconds

User Name admin

Password changeme

TLS Version Number None Check Server Hostname

Broker Name

CA Certificate Local File Path ...

Remove CA Certificate Update Selected Broker

Brokers

IP	Port	KeepAlive	UserName	Password	TLS	Check Host	Broker Name	CA Local File Path
10.1.200.12	1883	60	admin	changeme	None	Yes		

Delete Broker(s)

OK Cancel Help

2. Now add a new broker or edit the current broker.
3. In the Port Number you will need to change it to the port used for SSL security. The Ignition SSL secure port is 8883.
4. For TLS Version Number select the newest that is supported by your broker. Most brokers should be able to use TLS version 1.2.
5. Next in the Certificate Authority (provided by broker), find and select the file in your local directory.
 - a. You can also select the checkbox next to Check Server Hostname. If you do check this make sure to enter your hostname in Broker Name field. With this box checked the Server Certificate will be checked to make sure that the hostname matches.
6. Finally click the Update Selected Broker button.
7. The EZRack will now use TLS (SSL) secure communication and check the Certificate of the broker.

11.3.3 Ignition Encryption and Keystore Setup

To setup the Ignition MQTT Distributor Module to use encryption follow the directions below:

1. Go to localhost:8088 and Log into your Ignition Software. *Note: This assumes you have already setup the Ignition platform and MQTT Modules setup.*
2. Go to the Configure tab and select **MQTT Distributor > Settings**.
3. Under General the TLS Configuration will need to be enabled. You can also at this point change the secure MQTT port if you want.

TLS Configuration	
Enable TLS	<input checked="" type="checkbox"/> Enable or Disable TLS for the MQTT Server
Secure MQTT Port	<input type="text" value="8883"/> TLS enabled MQTT Server port
Secure Websocket Port	<input type="text" value="9443"/> TLS enabled MQTT Server Websocket port
Keystore Password	<input type="text"/> Java keystore password
Java Keystore File	<input type="button" value="Choose File"/> No file chosen Java Keystore File to upload for SSL enabled MQTT

4. Next you will need to provide a Java Keystore file to the Distributor. This Keystore will include the Certificate for the Broker and the Key pair of public and private key for the Broker. Use the Choose File option and navigate to the Keystore to select it.
5. You also need to enter the Keystore Password in the field above the File.
6. Finally click the button to Save Changes. The MQTT distributor will now have TLS (SSL) Security and Authentication Enabled.

The EZRack and MQTT Distributor should now be able to communicate over the secure MQTT port.

11.4 Redundancy Setup (EZ Rack PLC and Ignition):

The EZ Rack PLC supports a redundancy backup system of Brokers. What this means is that the Broker tab allows the user to input up to 4 different Brokers. If more than one broker is setup and the EZ Rack cannot connect to the first Broker it will cycle through the rest of Brokers and try to connect to them. This makes sure that the EZ Rack will be connect even if a Broker fails.

The EZ Rack also supports the Redundant Ignition Environment. To learn more about the Ignition Redundant Environment you can visit Cirrus Link website and their [help documentation](#). This environment is setup to have a failover backup for the primary/master ignition instance. It also lets the EZ Rack know that it has moved to the backup through the Primary Host ID.

The screenshot shows the 'Brokers' tab in the EZ Rack configuration. It features a 'Client ID' input field with a 'Generate Unique ID' button to its right. Below this, the 'Primary Host ID' is set to 'Primary' and is highlighted with a red rectangular border.

To setup the Redundant Ignition Environment on the EZ Rack you will only need to input the Primary Host ID from the MQTT Engine Settings. The Primary Host ID which is used to ensure client to client communications. The only requirement is that it match exactly on both the MQTT Engine and EZ Rack configurations. Also you will need to ensure that both the Master and Backup Brokers are listed in the Brokers tab.

MQTT Engine Settings

The screenshot shows the 'Advanced Settings' tab for the MQTT Engine. Under the 'Configuration' section, the 'Enabled' checkbox is checked. Below this, the 'Primary Host ID' is set to 'Primary' and is highlighted with a red rectangular border. A descriptive note below the field reads: 'The Primary Host ID to allow connecting clients to ensure they remain connected to this application (optional)'.

Cirrus Link Documentation:

<https://docs.chariot.io/display/CLD/Advanced%3A+MQTT+Modules+in+Redundant+Ignition+Environment>

11.5.1 Store and Forward Time Zone Setup

When the EZRack PLC has to store and then forward the information, the time stamp used with the Ignition software needs to be in GMT (UTC). The EZRack Time and Date is setup to be used in your current time zone and therefore for the Store and Forward to work correctly the different from GMT needs to be inputted into the System Register `_SR_TIME_ZONE` (SR29). Please see directions below about setting this up correctly.

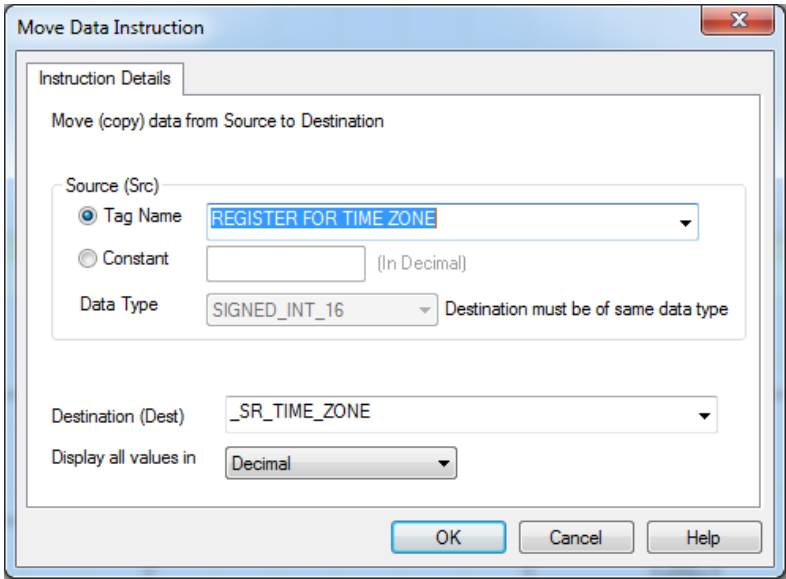
1. To setup the Time Zone, please first go to a website like <https://www.timetemperature.com/> to figure out your time zone. Some basic time zones are mentioned below.
2. Convert the hour time zone difference to minutes. For example if for Central time of -6 hours, please use -360 minutes.

Time Zone	GMT time offset (UTC)	Number to Enter into <code>_SR_TIME_ZONE</code> (SR29)
Eastern Standard Time (EST)	- 5 hours	-300
Central Standard Time (CST)	- 6 hours	-360
Mountain Standard Time (MST)	- 7 hours	-420
Pacific Standard Time (PST)	- 8 hours	-480
Middle European Time (MET)	+ 1	60
Indian Standard Time (IST)	+ 5.5	330
China Standard Time (CST)	+ 8	480
Australian Eastern Standard Time (AEST)	+ 10	600

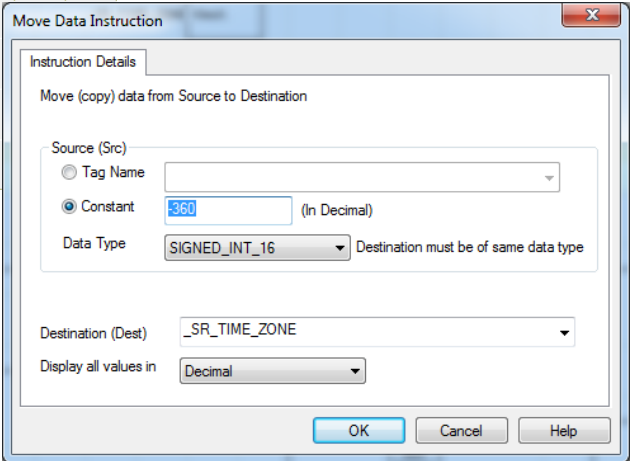
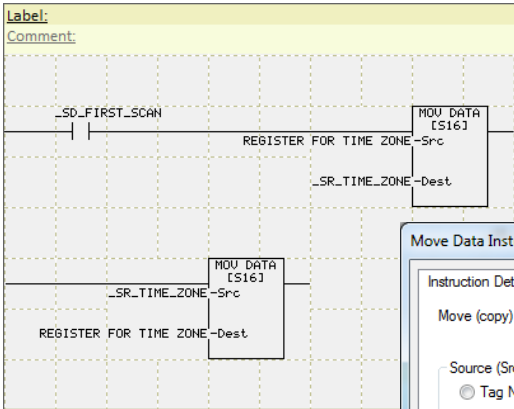
3. Next create a first scan rung with a Move Data instruction.



4. If you do not need to set this external (ex: from HMI) then you can set the Move Data Instruction to move the constant value selected above to the System Register. *Note: System registers are not retentive over a power cycle.*



- 5. If you need to set it externally, please use a Register tag to store the value and move this value into the _SR_TIME_ZONE upon power up. Please note that you should set the System Register from the HMI and then have an instruction to store the value into a register like below.



11.6 Troubleshooting Sparkplug B Setup:

This section will outline some basic EZRack PLC Sparkplug B troubleshooting tips. The EZRack PLC has some System Discretes and System Registers to help figure what might be causing your problem.

System Discretes

System Discretes Name	System Discretes	Description	Read/Write
_SD_SPARKPLUG_CONFIG_ERROR	SD29	This bit will turn on if the Sparkplug configuration has an error.	Read Only
_SD_SPARKPLUG_PUBLISH_ERROR	SD30	This bit indicates if the EZRack PLC is not able to publish a message to the broker.	Read Only
_SD_SPARKPLUG_RUNNING_STATUS	SD31	This bit is ON if the Sparkplug MQTT is connected to the Broker and publishing correctly.	Read Only

System Registers

System Register Names	System Registers	Description	Read/Write	Data Type
_SR_TIME_ZONE	SR29	This register is used to set the timestamp of messages to correspond to GMT (UTC).	Read/Write	S16
_SR_SPARKPLUG_CURRENT_BROKER_INDEX	SR30	If multiple Brokers are defined this register will indicate which broker is connected (0-3)	Read Only	US16
_SR_SPARKPLUG_STORE_FORWARD_FIFO_COUNT	SR31	When Store and Forward is enabled this will indicate how many tag updates are stored	Read Only	US16
_SR_SPARKPLUG_TASK_COUNTER	SR32	Each time that Sparkplug communication is initiated this counter will increment.	Read Only	US16

Troubleshoot Tips

Error	Suggested Solution
Configuration Error (SD29 is ON)	This suggested that your configuration on the Sparkplug setup is incorrect. Recommend creating a simple Sparkplug program and seeing if that works. Then work up to the project that caused this error.
Publish Error (SD30 is ON)	This suggest that broker is refusing EZRack MQTT messages. I would check that the EZRack has permission to publish to those topics. Also I would check the username and password of the EZRack PLC Broker sign in. Also check that Client ID is unique.
Sparkplug Not Running (SD31 is ON)	Please make sure that the Broker is accessible from the network the EZRack PLC is on. Further check the username and password of the EZRack PLC sign in. Also check that Client ID is unique. Finally if using TLS security I would make sure that the CA is current and that the PLC time makes the CA valid.

Error	Suggested Solution
Store and Forward Not Working	Please make sure that the tags you would like to store are set to store configuration. Also please make sure that you are setting the time zone correctly so that store forward slots the information in the right time slot. Note that ignition works of GMT (UTC) and not the current time zone you are in.
EZ Rack connected to Broker but Ignition is not seeing connection	Please make sure that the EZ Rack is connected to the correct Broker that the Ignition software is connected to. Also please make sure that both EZ Rack and Ignition have correct permissions with Broker (ACLs).
Non-TLS connection works but TLS does not	Please make sure that the for TLS security you have included a valid Certificate Authority and that the Time on the EZ Rack PLC has been set so the CA is valid.