

# **ControlLogix Level 1**

## **Maintenance and Troubleshooting**

**EthernetSupport.com**

*"Customized Automation Training"*



Copyright (c) 1999 Ricky Bryce.

Permission is granted to copy, distribute verbatim copies of this document, commercially or non-commercially with no front cover texts, or back cover texts. Changing this document is not allowed.

14MAR07

Disclaimer:

This document is written in the hope that you can utilize for your own education to gain knowledge of PLC systems (should you decide to utilize this document).

Although I believe the information in this document to be accurate, it is YOUR responsibility to verify this information before implementing it in any way, especially when damage to personnel or equipment could result.

By continuing to read this document, you agree to hold no one who writes, modifies, or distributes this document liable in any way (even negligence).

Due to the wide variety of plant applications, some of the examples in this document may be prohibited at your location, or could cause damage to equipment, or harm personnel.

About the Author:

This document is a collection of texts and graphics I've put together over the past few years, and has been distributed under the GFDL since 1999.

I hope you get much use out of it, and I would like your feedback as to how this document can be improved.

As a supplement to this document, I would like to invite you to my website at **<http://www.LearnAutomation.com>**. I'm in the process of uploading documentation and videos that will further help you with problems or questions you have with Allen Bradley processors.

"Human Knowledge Belongs to Everyone"

## Table of Contents

Questionnaire for Allen Bradley.....	9
Pre-Assessment.....	11
Glossary.....	15
Understanding Numbering Systems.....	19
Hardware .....	21
Discrete Input Modules.....	21
Discrete Output Modules.....	22
Analog Modules.....	23
The Chassis.....	24
The PLC-5 Chassis.....	24
The SLC-500 Chassis.....	24
The ControlLogix Chassis.....	25
The Flex Chassis.....	25
The Power Supply.....	26
The Processor.....	27
The Ethernet Module.....	30
Ethernet Addressing.....	32
Addressing.....	32
Subnet Mask.....	33
RSLinx.....	37
Utilizing the BootP/DHCP Server.....	37
Setting up the Ethernet Driver.....	39
Configuring the DF1 Driver.....	42
Backup Restore Utility.....	45
Flashing ControlLogix Modules.....	48
Creating a new Project.....	52
Setting up Local I/O Modules.....	53
Working With Tags.....	58
Controller Tags.....	58
Program Tags.....	60
Aliasing.....	65
Remote Chassis.....	67
Simple Subroutines.....	80
Basic Instructions.....	86
Examine If Closed.....	86
Examine If Open.....	87
Output To Energize.....	87
Output To Latch.....	88
Output To Unlatch .....	88
Timers.....	89
Timer On Delay (TON).....	90
Timer Off Delay (TOF).....	91
Retentative On Delay Timer (RTO).....	92

Counters.....	93
Analog Configuration.....	95
Compute Statement.....	97
Compare Statements.....	98
GSV Command.....	99
On line Editing.....	104
Start Rung Edits.....	105
Make Changes.....	106
Accept Edits.....	107
Test Edits.....	108
Assemble Edits.....	109
Forcing I/O.....	110
User Defined Data Types.....	112
Simple UDT's.....	113
Nesting UDT's.....	116
Trending.....	120
Troubleshooting.....	129
Search/Find.....	129
Cross Reference.....	129
Trending.....	129



Name \_\_\_\_\_ Date \_\_\_\_\_

## ***Questionnaire for Allen Bradley Automation Systems***

- 1)What is the primary purpose you are attending this class?
- 2)Are you interested in programming, troubleshooting, or both?
- 3)What do you find most difficult about Allen Bradley PLC's
- 4)How often do you access the Allen Bradley PLC? (once a day, once a week, once a month, etc?)
- 5)After taking this class, will you be putting your knowledge to use right away in the plant?
- 6)What type of equipment do you generally work with?
- 7)What types of networks are you using with your PLC system? Ie... Data Highway plus, controlnet, devicenet, Ethernet, etc?
- 8)What is your company's policy on forcing?
- 9)Do you generally have access to the Internet as you work?
- 10)Can you bring a copy of some plant programs into the classroom tomorrow?

11) Will you ever be installing new systems, or checking new systems once they have been installed?

12) Will you ever be modifying the I/O structure of existing systems?

13) Do you have any common system failures that are related to the Allen Bradley PLC? If so, what are these failures

14) Are you interested in learning features of RSLogix that are not currently in use by your plant, but, if used could reduce downtime?

Name \_\_\_\_\_ Date \_\_\_\_\_ Score \_\_\_\_\_

**ControlLogix Maintenance And Troubleshooting**  
**Pre-Test**

- 1) Name three devices that can be connected to a **discrete input** module:
  
- 2) Name three devices that you might find connected to a **discrete output** module:
  
- 3) Name one device that can be connected to an **analog input** module:
  
- 4) Name one field device that you would find connected to an **analog output** module:
  
- 5) On a discrete input module, a status light is on. What does this indicate?
  
- 6) On a discrete output module if a status light is on, what does this indicate?
  
- 7) Name one use of the serial port (communication channel 0) on the front of the processor.
  
- 8) What is the purpose of **RSLinux** software?
  
- 9) What is the purpose of **RSLogix 5000** software?

10)What is the purpose of **Tags** (Data Files)?

11)What is the purpose of **Tasks** (Program Files)?

12)Define a **BIT** of memory:

13)How many bits are available in a tag with the **DINT** data type?

14)What instruction must be placed in the Main Routine so the Sub-Routines will execute?

15)What are the five steps involved in performing an on line **edit** if the processor is in remote run mode?

16)If you have a motor that will not run when the operator pushes a start button, you may need to go on line with the processor and locate the output in the ControlLogix program that energizes the motor. This allows you to see what other conditions must be met before the processor will energize the output. What features of RSLogix might you use to locate the motor's output in the ControlLogix project?

17)In #16, you find that the reason for the failure is a bad thermal switch on the motor. Since you do not have any more thermal switches in stock, you decide to force the motor's output instruction on. What are some dangers with this, and what are some better options that you could have chosen.

- 18)The processor is in Remote Run mode. You perform an on line edit and inadvertently get the processor caught in an infinite loop. What will happen when your edits are tested?
- 19)Due to the way the ControlLogix processor scans the I/O, programmers will sometimes have a routine for Input Buffering, and another for Output Buffering. What would be the purpose of these two routines?
- 20)An operator informs you of variations in a product. You suspect the variations in product are due to a fluctuation in temperature. What feature of RSLogix software would allow you to graph a temperature over time?
- 21)If a neon lamp is wired directly to a triac output module, the neon lamp is always on even when processor is not calling for the output. What causes this, and how can it be corrected?
- 22)What is the purpose of the battery on the processor?
- 23)With a network of PLC's, it is possible to inadvertently download to the wrong processor. This can have disastrous results. What steps can you take to ensure you are not downloading to the wrong system?
- 24)Name some communication protocols that may be used for peer to peer communication between networked PLC's.
- 25)Name some communication protocols that may be used for communication between a processor, and I/O devices such as a remote chassis, drive, or robot.



# Glossary

## Addressing:

**Bit:** Smallest unit of information the PLC can process– ON or OFF

**Word:** 32 Bits for Double Integer

## Hardware:

**Input module:** Reads the STATUS of field devices

**Output module:** CONTROLS field devices

**Discrete module:** Reads or controls devices which only have 2 states:  
on or off

**Analog Module:** Reads or controls devices which have a range  
such as 0 to 10 volts or 4 to 20 milliamps

**Power Supply:** Provides control power to modules on the backplane

**Chassis:** The physical device that modules are plugged into.

**Local Chassis:** The chassis where the processor resides.

**Processor:** The 'Brain' of the PLC which contains the machine program.

## **Troubleshooting Tools:**

### **Cross Referencing:**

Allows the troubleshooter to quickly navigate through the program by listing all locations in ladder logic where a particular address is located.

Usually the troubleshooter will cross reference a false condition on a rung of logic to find the output that will turn the referenced bit on.

You can Access Cross Referencing by right clicking a particular address. (Note: You must be on the address, not the instruction)

### **Custom Data Monitor Utility:**

Allows the troubleshooter to gather data from various memory locations onto one screen for easy troubleshooting. For example: One can create a custom data monitor for the failure of a particular motor. Next time the motor fails, the troubleshooter can simply look down the list of conditions that must be met, and see in real time which condition is causing the failure.

The custom data monitor utility has to be installed as a separate tool. Since the CDM utility is not built into RSLogix, you must have RSLinx activated.

RSLinx Lite will not work with the CDM utility.

### **Force:**

Simulates real world jumpers. Use care while performing a force. You must understand fully how a force is going to affect your system. In most cases, only addresses starting with an I: or an O: can be forced.

To force an input or output, you can right click on the address in logic, then choose force on or force off. After the force is installed, forces can then be enabled from the on line tool bar.

### **Trending:**

Trending acts somewhat like a 'software chart recorder', and allows you to track an analog signal over time.



### **Communication terminology:**

**RSLinx:** This is the communication server. If RSLinx is not set up properly, RSLogix will not communicate to the processor.

**Driver:** Allows RSLinx to communicate with a particular hardware device. The most common drivers are the DF1 driver to communicate with Channel 0, and the PCMK driver for a laptop to communicate to Channel 1A for station 5. Configure drivers by clicking communication on the menu bar.

**RSWho:** A graphical screen which will display what devices RSLinx has established communication with. Access RSWho by clicking communication on the menu bar. Then click on the name of the driver you wish to use for communication. The right hand side of the screen will reveal devices the driver has communication with.

**RSLogix:** The software which allows you to troubleshoot or program a processor.

**Online:** Actively communicating with the processor (ladder spinning)

**Download:** If a program was changed offline, it must be downloaded (or sent to) the processor. When downloading the processor must be in program or remote program mode. A good way to download once RSLinx is properly set up is to click COMMS on the menu bar, and then go to Who Active. Click the driver name, highlight your processor, then click DOWNLOAD.

## **Memory Layout:**

**Tags:** A section of the processor's memory that stores information. You can also think of tag elements as variables. For example: The memory location “MainTorque” could store a drive torque value.

There are two scopes of tags: Program tags which are local to the program they Reside in, and Controller Scoped tags which are global. You can access the Program Tag database by double clicking program tags just above the MainRoutine of each program. Controller tags can be accessed at the top of the Controller organizer window.

**Tasks:** A section of the processor's memory which holds programs. Programs hold Routines. Each controller can have multiple tasks with multiple programs in each task. Each of these programs can then have multiple routines.

## **Atomic Data Types:**

<b>BOOL</b>	1-bit boolean 0 = cleared 1 = set
<b>SINT</b>	1-byte integer -128 to 127
<b>INT</b>	2-byte integer -32,768 to 32,767
<b>DINT</b>	4-byte integer -2,147,483,648 to 2,147,483,647
<b>REAL</b>	4-byte floating-point number -3.402823E38 to -1.1754944E-38

# Understanding Numbering Systems

Understanding numbering systems will help you to understand various ways in which data can be monitored in the ControlLogix processor. For example, if you are reading the value of a limit switch, you would want to change the numbering system (style) to binary. If you were viewing data from an analog module, you would want to set the style for decimal. The PLC-5 modules are numbered in an Octal addressing scheme, so if you read data from a PLC-5 module, you may want to set the style to Octal. Some devices such as LED displays are wired in Hex/BCD. The two most common styles for most plants will be Binary and Decimal.

- 1) **Binary** – Binary is a base 2 numbering system. You only have two numbers available in Binary, 0 and 1 for any position (ones, tens, hundreds, etc...). Binary is the most common style for discrete I/O such as limit switches, pushbuttons, solenoids, and motor starters.
- 2) **Decimal** -- Decimal is a base 10 numbering system. Only 10 numbers exist in the Decimal numbering scheme (0 to 9) for any given position. The Decimal style is used most often when displaying analog data, such as a pressure or temperature.
- 3) **Octal** – Octal is a base 8 numbering system. In Octal, 8 numbers are available (0-7) for use in any position. In older PLC's such as the PLC-2, and the PLC-5, The I/O modules were numbered in the Octal addressing scheme. The Octal style can be used when connecting to one of these older modules.
- 4) **Hexadecimal/Binary Coded Decimal** – Hex/BCD is a base 16 numbering system. 16 numbers are available for any digital position (0 to 9 then A to F). Devices such as LED displays and thumb wheels can be BCD Devices.

On the following chart, you will write down the decimal numbers 0 to 15, and the numeric conversions for each of these numbering systems. This chart will help you understand topics covered later in the course such as masking.

Decimal	Binary	Octal	Hex/BCD

## ***Hardware -- Discrete Input Modules***

The purpose of the discrete input module is to read the status of field devices. When a voltage is detected on the terminal of an input module with respect to common, the corresponding status light is energized, and during the processor scan, the value of 1 is placed into the input data table. Examples of input devices include switches, pushbuttons, or auxiliary contacts on a motor starter. The Removable Terminal Block (RTB) can be detached from the module if the locking tab is pushed up. The enclosure of the RTB will also slide off the terminal block for easy access to the terminals.

Please answer the following questions:

- 1) What is the catalog number of your DC Input module?
- 2) Name at least three field devices that can be connected to the DC Input module?
- 3) What do the status lights indicate on the front of the DC Input Module?
- 4) What is the slot number of the DC Input module at your station.



## ***Discrete Output Modules***

The purpose of the discrete output module is to control field devices. The discrete output module requires power from an external source. When a 1 is placed into the output tag of the ControlLogix (in run mode), a status light is energized on the module, and a connection is made between the source, and the corresponding output terminal. Examples of output devices include: lights, solenoids, motor starter coils, and contactors. If you have an inductive load as the output, be sure to use the proper surge suppression.

Please answer the following questions:

- 1) What is the catalog number of your DC Output module?
- 2) Name at least three field devices that can be connect to the DC Output module:
- 3) What do the status lights indicate on the DC Output module?
- 4) What is the slot number of the DC Output Module?
- 5) If the load on the DC output card is inductive, what should be done across the load to minimize the effects of inductive kick?



## *Analog Modules*

Analog modules are used to control and read the status of analog devices. Analog devices have a range of states instead of just on/off states like discrete devices.

Some analog modules have switches which determine whether the input channels are to be set up for voltage or current. Some analog modules are configured through software.

Examples of analog inputs include: Potentiometer, Pressure Transducers, Variable speed drives, and with a thermal couple module, temperature can be read into the processor's memory.

Examples of analog outputs include: Meters, Variable Speed Drives, Valve Positioners, and chart recorders.

An analog signal cannot be expressed with a single bit, and therefore analog values will consume a word of memory. For our class, we will use the Analog module on the Flex I/O chassis.

Please answer the following questions:

- 1) What is the catalog number of your analog module?
- 2) How many channels of Input, and how many channels of Output are available on this module?
- 3) How do you set up the input channels to accept either a current or a voltage input?
- 4) What range voltage or current will the Input channels accept on your module? What range of voltage will the output channels accept?
- 5) Name at least three devices that are analog inputs:
- 6) Name at least three field devices that are analog outputs:

## *The Chassis*

The chassis is the device which holds modules. Allen Bradley makes the ControlLogix chassis available in 4, 7, 10, 13, and 17 slots.

Here are some chassis:

The PLC-5 Chassis (With modules): (For dip switch settings on this chassis, refer to page 4-1 and 4-2 of the PLC-5 Quick Reference Guide.)



The SLC-500 Chassis (With modules):





The ControlLogix Chassis (With Modules):



The Flex Chassis:



## *The Power Supply*

The power supply supplies power to the modules on the backplane. Generally power from field devices DOES NOT come from the power supply. The power supply only provides control power to modules on the backplane. Power for field devices come from a separate source which is connected to the output module. The power supply merely provides the power needed to shut a contact, or fire a triac or transistor to pass power from this external source to the field device. On the back of the power supply, a jumper is used to set the voltage range.

Please answer the following questions:

- 1) Where does power come from to power field devices such as solenoids, lights, and motor starters?
- 2) What must be set up on a new power supply before it can be placed into service?
- 3) How many amps will your power supply provide to the backplane?
- 4) What is the catalog number of your power supply?



## *The Processor*

The processor is the main part of your ControlLogix system. The processor is where the program is stored that reads the status of your equipment, and based on certain status, makes a decision on what to control. For example: The processor is reading the status of a switch. When the operator energizes the switch, the processor might call for solenoid to energize that extends a cylinder. When the cylinder reaches the end of its travel, it might close a limit switch. The processor will see that a limit switch has been closed, and shut off the solenoid. Although traditionally the processor usually is placed in slot 0, it can be placed anywhere in the chassis, as long as the program is setup for the processor to be in that slot. You can also use as many processors as you like in a chassis (not to exceed the limitation of the power supply)

The processor consists of several components:

- 1) The battery: The battery retains the processor's program when the PLC is powered down. Certain AB documentation states that the shelf life of the battery is up to 2.5 years. When the battery is low, you will see a BATT light on the front of the processor. A minor fault bit is also set in the memory of the processor when the battery is low or missing.
- 2) On the front of the processor, you will find several status lights:
  1. RUN – Indicates when the processor is in RUN Mode
  2. OK – If flashing red, usually indicates a software problem. Go on line to get a description of the fault. You will find the description in the Controller Properties on the FAULT tab. If the fault light is solid red, this could indicate a hardware problem. You can try the following: re-seat the processor, clear memory and reload program, or replace processor.
  3. BATT-- Indicates the battery is low or missing
  4. IO – If flashing indicates the Processor lost its connection with at least one I/O device.
  5. FORCE – If flashing indicates forces are installed but not enabled... If solid indicates forces are installed, and enabled in the processor. This indicator is not available on all ControlLogix processors.
  6. RS232 – This light will flicker as data is transferred over the RS232 port (channel 0).
- 3) The Key Switch:
  1. Run Mode: In this position, certain tag values can be modified, but ladder logic cannot. The mode of the processor cannot be changed to program mode from the On line tool bar in RSLogix. When the switch is in run mode, a program cannot be downloaded to the processor.
  2. Program Mode: In this position, the ladder is not executing. Changes can be made to the ladder diagram or to data files. The processor cannot be changed to run mode from the on line tool bar in RSLogix while the switch is in this position.
  3. Remote Mode: When the key switch is in Remote Mode, the mode of the processor can be changed from RSLogix (Program or Run). On line editing is allowed.



## The ControlLogix Processor:



## ***The Ethernet Module***

Ethernet is a protocol that has been widely used for many years. Before Ethernet was used with PLC systems, it was used in Office environments for sharing files, printers, data from databases, etc...

Ethernet is the fastest communication protocol available for the ControlLogix system with speeds up to 100 Mbps (Million bits per second). Ethernet can be used to communicate with the ControlLogix system from a computer, for communication between controllers, to allow the controller to communicate with I/O, Or MMI (man-machine interface) devices to communicate with processors.

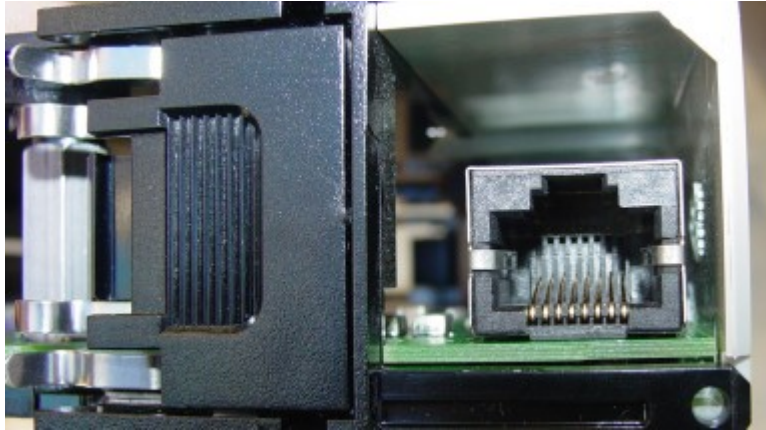
Every device on Ethernet has a Unique hardware address which can usually be found on the device itself, or on a configuration screen for the device. This hardware address can be used to issue the Ethernet module an address using a bootP utility. The hardware address will also scroll across the alphanumeric display if no IP address has been assigned.



Once the IP address is assigned, the IP will scroll across the alphanumeric display. This IP address can then be used in the Ethernet Driver for RSLinx, or if type the IP address into the address bar of a web browser, such as Mozilla, the modules on board web server will show the module's status, and the status of every other module in the chassis.



On the bottom of the 1756-ENBT module, you will find an RJ-45 port. Using a standard patch cable, you can connect the module to an Ethernet switch or a hub. To connect to your computer directly, you will need a crossover cable (switches transmit and receive).



## *Ethernet Addressing*

Every device on the same Ethernet network must have a unique address. Allen Bradley PLC's currently use the IP (Internet protocol) addressing scheme (version 4). This is the addressing scheme discussed in this document.

Examples of Ethernet devices might be a Personal Computer (PC), a 1756-ENBT module, a 1794-AENT flex adapter, a printer with built-in print server capabilities, and plant servers for data storage and processing.

To see what the IP address is of your Windows NT, 2000, or XP machine type:

**ipconfig /all**

at the command prompt. Here is the result:

```
Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . : 
    Description . . . . . : SiS 900 PCI Fast Ethernet
    Physical Address. . . . . : 00-0D-87-03-F4-71
    Dhcp Enabled. . . . . : No
    IP Address. . . . . : 192.168.0.101
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1
    DNS Servers . . . . . : 12.127.17.72
                           12.127.16.68

C:\Documents and Settings\rbryce>
```

For Unix/Linux systems, the command **/sbin/ifconfig** will product similar results.

In this example, my PC has the IP address of **192.168.0.101**. No other machine on the same network will have this same IP address, nor should you attempt to assign the same address to another device on the network. If this happens, one of the devices will not be seen.

Each segment of the IP address is called an **OCTET**. All IP addresses in the IPv4 addressing scheme are made up of 4 octets. Each octet is an 8 bit unsigned integer.



## Subnet Mask

Notice the Subnet Mask: **255.255.255.0**. The purpose of the subnet mask is to identify which part of the IP address is the network address, and which part of the IP address is the host or terminal on the network. To understand how the subnet mask works, it has to be broken down into a binary format... Look at the IP address in Binary:

11000000 . 10101000 . 00000000 . 01100101 = 192.168.0.101

Now look at the subnet mask:

11111111 . 11111111 . 11111111 . 00000000 = 255.255.255.0

Anywhere a 1 exists in the subnet mask, that bit of the IP address is viewed as the NETWORK part of the address.... Lets see what bits are passed:

11000000	.	10101000	.	00000000	.	01100101	=	192.168.0.101
11111111	.	11111111	.	11111111	.	00000000	=	255.255.255.0
<hr/>								
11000000	.	10101000	.	00000000	.	don't pass	=	192.168.0.X

Wherever there was a 1 in the subnet, we passed that bit of the IP address as part of the network address. Therefore, we would say the network address for this machine is 192.168.0.X. Every device on this network must have an IP address that starts with 192.168.0. and X is the terminal address on the network.

For example: Devices with these two IP addresses can communicate with each other directly without going through a router:

**192.168.0.3 and 192.168.0.200 with subnet 255.255.255.0**

These two devices cannot communicate with each other directly:

**192.168.1.3 and 192.168.3.200 with subnet 255.255.255.0**

However if the subnet mask was changed:

**192.168.1.3 and 192.168.3.200 with subnet 255.255.0.0**

These two devices will communicate with each other because the network address is only made up of the first two octets, 192.168.X.X... since the network address is the same for the two devices, they will communicate directly.

## Subnet Mask Exercise:

Indicate whether or not the following devices can communicate with each other directly:

- |     |             |   |     |    |
|-----|-------------|---|-----|----|
| 1.  | 192.168.0.5 | and 192.168.0.6 with subnet mask of 255.255.255.0 | YES | NO |
| 2.  | 192.168.1.5 | and 192.168.0.6 with subnet mask of 255.255.255.0 | YES | NO |
| 3.  | 192.168.1.5 | and 192.168.0.6 with subnet mask of 255.255.0.0   | YES | NO |
| 4.  | 10.1.1.5    | and 10.1.1.6 with subnet mask of 255.255.255.0    | YES | NO |
| 5.  | 10.1.1.5    | and 10.1.1.6 with subnet mask of 255.0.0.0        | YES | NO |
| 6.  | 10.2.5.5    | and 10.1.1.6 with subnet mask of 255.255.255.0    | YES | NO |
| 7.  | 10.2.5.5    | and 10.1.1.6 with subnet mask of 255.0.0.0        | YES | NO |
| 8.  | 10.2.5.5    | and 10.1.1.6 with subnet mask of 255.255.0.0      | YES | NO |
| 9.  | 192.168.0.1 | and 10.1.1.6 with subnet mask of 255.0.0.0        | YES | NO |
| 10. | 10.2.1.5    | and 10.1.1.6 with subnet mask of 255.0.0.0        | YES | NO |

## **Other Terms:**

### **GATEWAY**

The gateway address is the IP address of a server or hardware router that connects you to other networks such as the Internet.

### **DNS (Primary and Secondary)**

The DNS server (Domain Name Server) resolves host names into IP addresses. When you enter an address such as Yahoo.com into your web browser, your PC does not understand where to go. It must ask the DNS server to look up the IP address of a given host names. Host names are for humans to understand. Computers understand IP addresses.

### **DHCP**

Dynamic Host Configuration Protocol – When a device such as a computer is configured to use DHCP, A DHCP server should be available on the network. As soon as the device connects to the network, it will ask the DHCP server to automatically assign an IP address, subnet mask, DNS servers, Gateway address, etc. This address is dynamic, so the device could get a different IP address each time it's connected.

### **BootP**

Bootstrap Protocol – Similar to DHCP, except a BootP device will get the same IP address every time it connects. The BootP server has a list of hardware addresses, and IP addresses that belong to each device. When a device (such as a PLC) connects to the network, it will give the BootP server it's hardware address. The server will then look up the hardware address in a list, and see which IP address belongs to the PLC. The BootP server will then return the IP address (and other information such as the subnet mask) to be used by the device.



## ***RSLinux -- Utilizing the BootP/DHCP Server***

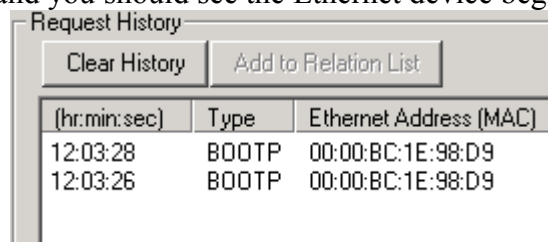
- 1) Write down the Ethernet Hardware Address of the device you wish to configure. This is also called the MAC address. Here is an example of a hardware address:

**00:00:BC:1E:98:D9**

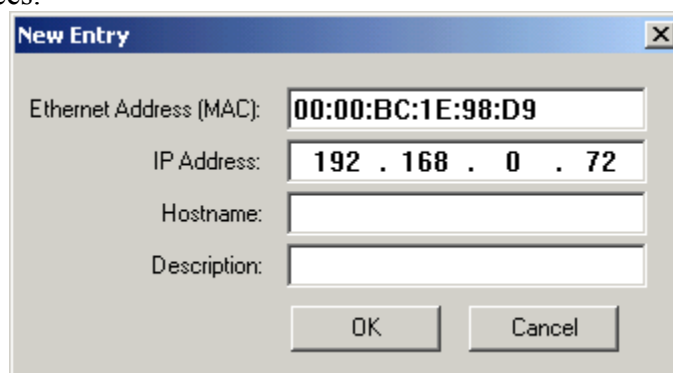
- 2) Run the BootP/DHCP utility. You can access this utility if it is installed by clicking Start | Programs (or All Programs in Windows XP) | Rockwell Software | BootP/DHCP Server | BootP DHCP Server.



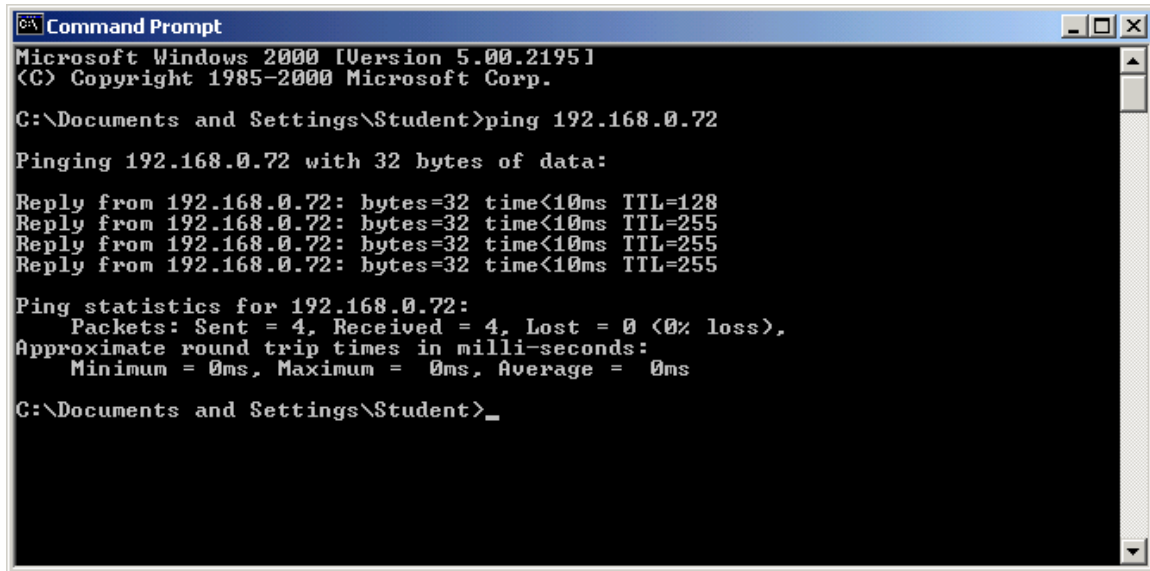
- 3) Once the server is open, it may ask for some specific network information if this is the first time the BootP server has been run. You can obtain this information from your network administrator. For this example, we just set the subnet mask to 255.255.255.0 then click OK.
- 4) Power up your processor, and you should see the Ethernet device begin to request an address.



- 5) Double click on the device. You will then be prompted to assign an IP address to the device. Be sure you double click the right device. Entering an IP address into the wrong equipment could have disastrous consequences.



- 6) You could also enter a host name and description at this time if you wish.
- 7) If you wish to verify communication, you can ping the device from the command prompt. By default, the command prompt can be accessed from Start | Programs | Accessories | Command Prompt.



```
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Student>ping 192.168.0.72

Pinging 192.168.0.72 with 32 bytes of data:

Reply from 192.168.0.72: bytes=32 time<10ms TTL=128
Reply from 192.168.0.72: bytes=32 time<10ms TTL=255
Reply from 192.168.0.72: bytes=32 time<10ms TTL=255
Reply from 192.168.0.72: bytes=32 time<10ms TTL=255

Ping statistics for 192.168.0.72:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\Student>
```

You should get replies. To ping continuously, use the -t flag after the ping command. A control-C will stop the ping command.

## ***Setting up the Ethernet Driver in RSLinx***

The Ethernet driver is used to make a connection to Ethernet Devices, such as an Ethernet PLC-5, or a ControlLogix system. The following steps will walk you through a sample configuration of the Ethernet driver in RSLinx.

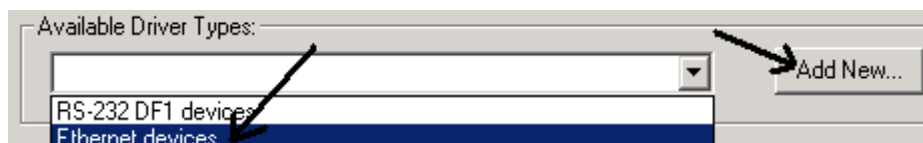
- 1) Open RSLinx communication server



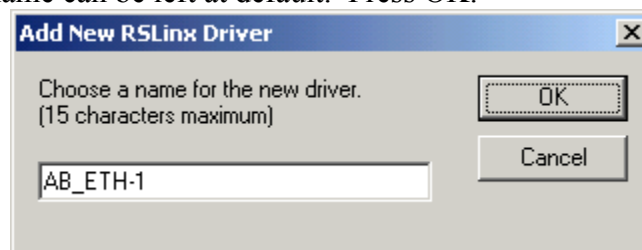
- 2) Click 'Communication' on the menu bar, and then choose 'Configure Drivers'.



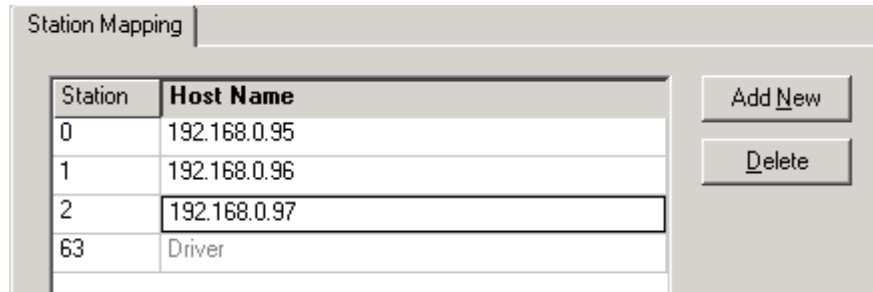
- 3) From the Available driver types pull down menu, choose 'Ethernet Drives', then press the 'Add New' button.



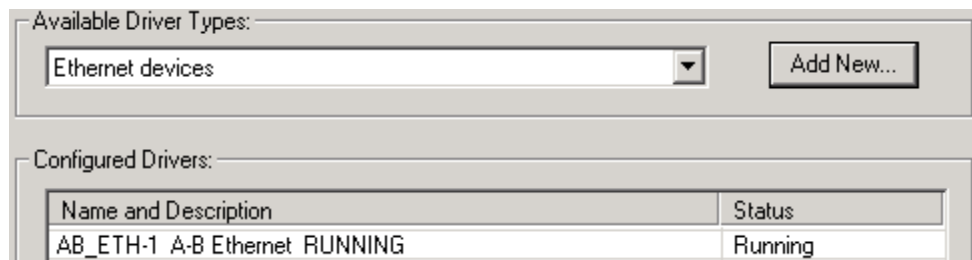
- 4) For this example, the name can be left at default. Press OK.



- 5) Populate the list of hostnames. If you do not have a way to resolve hostnames, you can enter the IP address of the devices you wish to connect to (as shown below in the example). The IP address for each device can usually be obtained from the network administrator, drawings, the offline project, or in some cases, the IP address is displayed on the front of the module.



- 6) Press Apply, then OK. You will see the driver is now running. Close the 'Configure Drivers' screen.

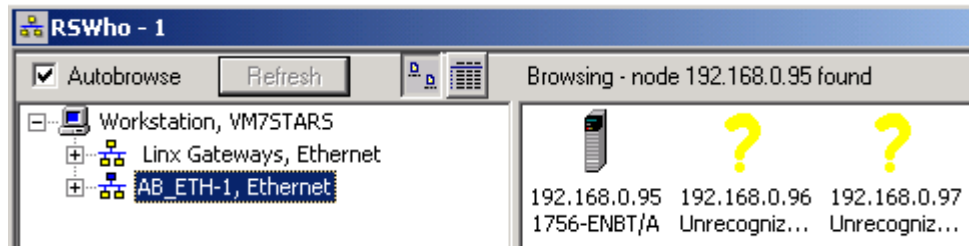


- 7) To test your drivers, click the RSWho icon in the tool bar of RSLinx.\





- 8) Click the Ethernet Driver (the one you just configured) on the left side of the screen. The devices you are communicating with will appear on the right. In this example, 192.168.0.95 is a ControlLogix module, and the other devices are not present, or not a recognized PLC module.



- 9) To go on line with a PLC, you must go to RSLogix at this point.

## Configuring the DF1 Driver in RSLinx

The DF1 Driver is used for point to point communication over RS232 between a COM port on a PC, and the serial port (Channel 0) of a processor. The following steps will take you through a sample configuration of the DF1 RS232 driver.

- 1) Open RSLinx Communication Server. If there is no short-cut on the desktop, you can access RSLinx by clicking Start | Programs | Rockwell Software | RSLinx | RSLinx



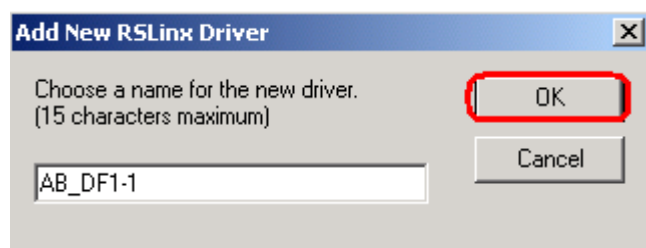
- 2) Click 'Communication' on the menu bar, then choose 'Configure Drivers'.



- 3) From the Available Driver Types pull down menu, choose 'RS232 DF1 Devices', then press the ADD NEW button.



- 4) For this example, the name can be left at default. Press OK.



- 5) Although the communication parameters can be entered manually, if you are currently connected to the processor, just hit the 'AutoConfigure' button. RSLinx will hit the processor with different baud rates, and different settings, until it finds a setting it gets a response on. When this happens, you will get a message that the autoconfiguration was successful. Press OK when finished.

**Configure RS-232 DF1 Devices**

Device Name: AB\_DF1-1

Comm Port: COM1 Device: Logix 5550 / CompactLogix

Baud Rate: 19200 Station Number: 00 (Decimal)

Parity: None Error Checking: BCC

Stop Bits: 1 Protocol: Full Duplex

**Auto-Configure** Auto Configuration Successfull

☐ Use Modem Dialer **Configure Dialer**

**OK** Cancel Delete Help

- 6) You will see the driver is now running. Close the "Configure Drivers" screen.

Available Driver Types:

RS-232 DF1 devices Add New...

**Close** Help

Configured Drivers:

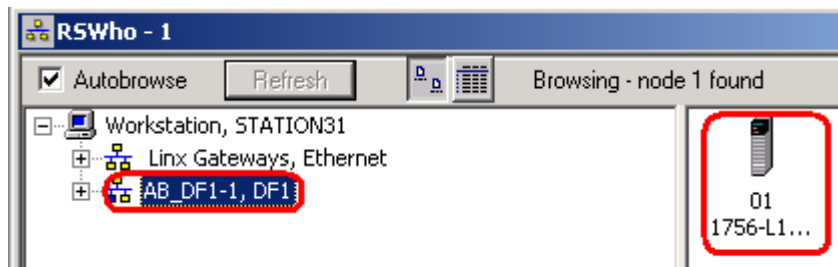
Name and Description	Status
AB_DF1-1 DF1 Sta: 0 COM1: RUNNING	<u>Running</u>

Configure...

7) To test your drivers, click the RSWho icon in the tool bar of RSLinx.



8) Click the DF1 driver (the one you just configured) on the left side of your screen. The devices you are communicating with will appear on the right.



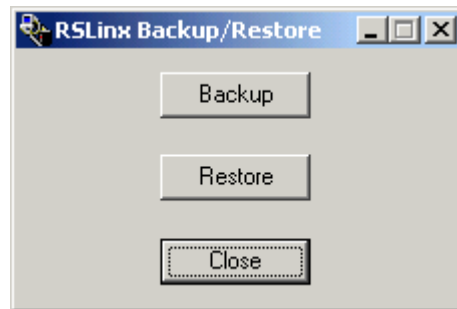
To go on line, you must go to RSLogix at this point.

## ***RSLinx Backup Restore Utility***

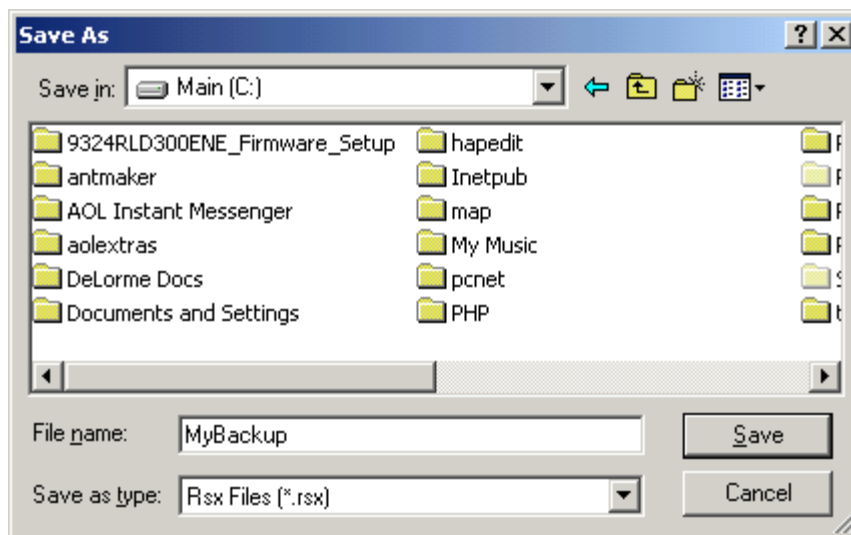
The RSLinx Backup Restore Utility can be used to backup the current driver configuration of RSLinx, or restore the configuration from a previous backup at an earlier time.

### **Backing up the current Configuration:**

To access the Backup Restore Utility, click Start | Programs | Rockwell Software | RSLinx | Backup Restore Utility.



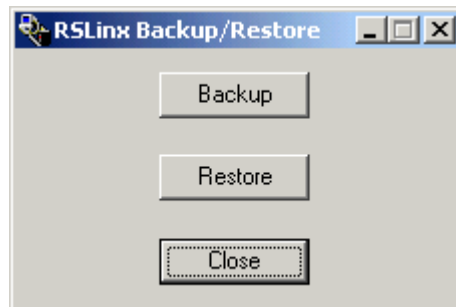
- 1) Click the 'Backup' button.
- 2) A dialog screen will appear asking where you want to save the backup file. Choose a location from the pull down menu. If you are saving this to a floppy disk, choose the A drive. For this example, the driver configuration will be saved to the C: Drive. You must also enter a file name, and then press SAVE.



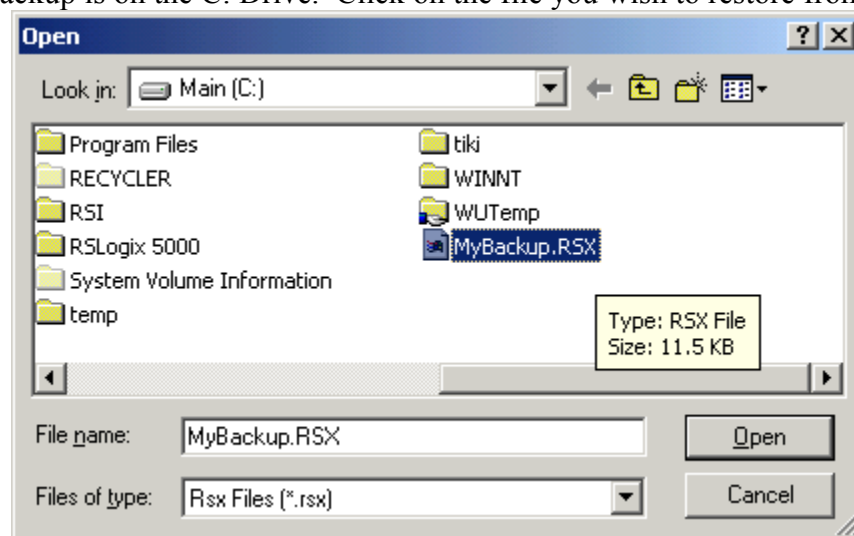
- 3) You will then get a message that the operation completed successfully. Press OK, then you can close the RSLinx Backup Restore Utility.

### Restoring a previous Configuration:

To access the Backup Restore Utility, click Start | Programs | Rockwell Software | RSLinx | Backup Restore Utility. Be aware that RSLinx must be shut down to perform this operation. If RSLinx is not shut down, you will be prompted accordingly.



- 1) Click the 'Restore' button.
- 2) A dialog screen will appear asking where the backup file is stored at. Choose a location from the pull down menu. If the backup file was on a floppy disk, you would choose the A: drive. For this example, the backup is on the C: Drive. Click on the file you wish to restore from, then press 'Open'.



- 3) A dialog box should appear indicating that the operation was completed successfully. If you got an error, try the restore procedure again. In some versions of RSLinx, BRU must be ran twice if RSLinx was open.



## ***Flashing ControlLogix Modules***

In order to flash ControlLogix firmware, you need to have the Control Flash utility installed. If it is not installed on your PC, the installation program can probably be found on the same disk as RSLogix 5000.

In this example, we are going to flash a 1756- L1 processor.

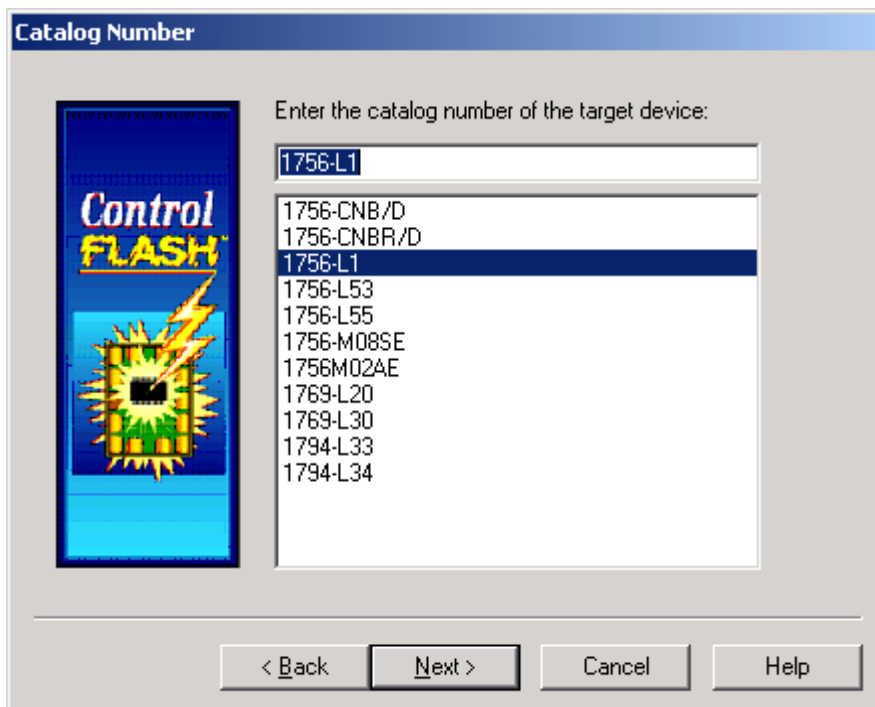
- 1) Before starting, make sure necessary drivers are configured in RSLinx. If your drivers are not configured in RSLinx, then restore from the appropriate backup file, or consult your documentation on how to configure specific drivers to communicate with your equipment. To open the Control Flash utility click START|PROGRAMS|Flash Programming Tools|Control Flash.



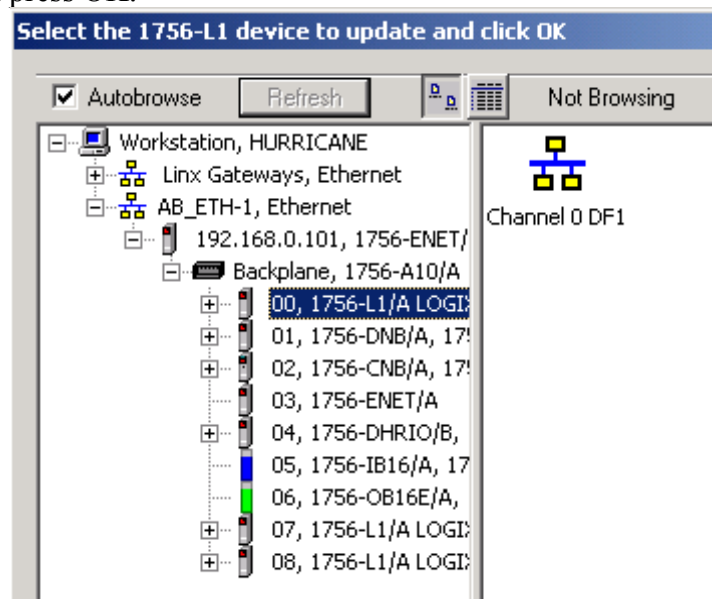
- 2) Be sure to read all instructions that are presented to you, and take all necessary precautions while flashing a module. If your flash procedure is interrupted, you could damage the module.



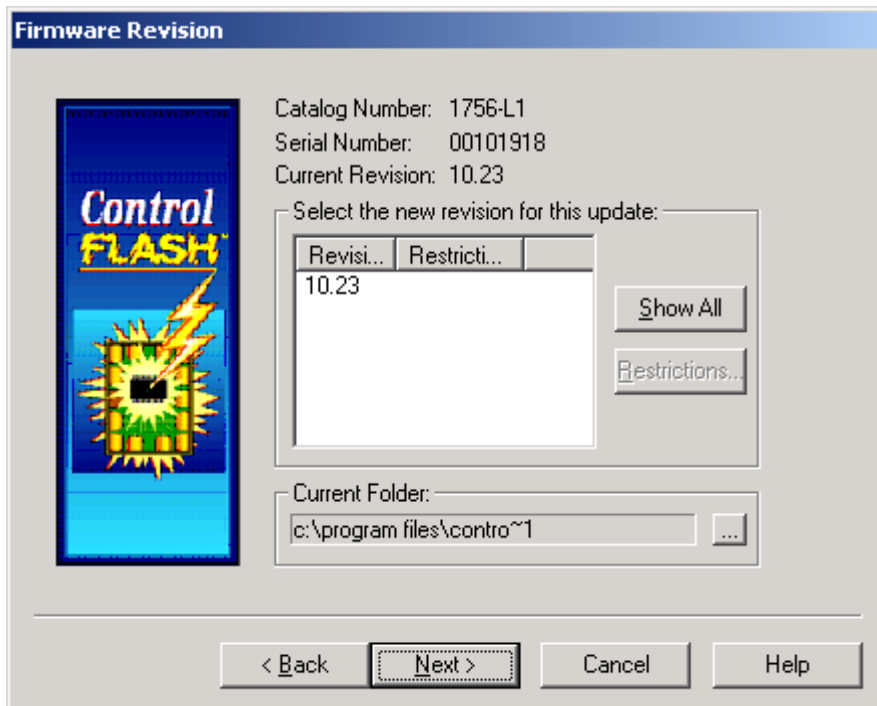
- 3) On the first screen, we are going to press the NEXT button.



- 4) Choose the device that you wish to flash. For this example, we will use the 1756-L1 option. Then press NEXT. Now the ControlFlash utility will ask for the path of the device you wish to flash. Drill down through RSLogix until you find the device you are wanting to flash. Highlight the device, then press OK.



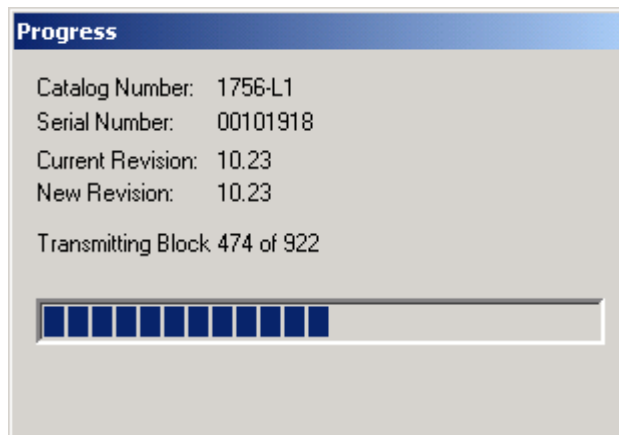
- 5) Next you are asked what firmware version you want to flash the module to. In this case, the module will be flashed to firmware version 10.23. If the processor's firmware was not the same firmware that we needed, we would highlight the revision of firmware we want for this update, and then press NEXT.



- 6) Next you will receive a warning. Read the warning carefully. If you still wish to flash the module, press FINISH.

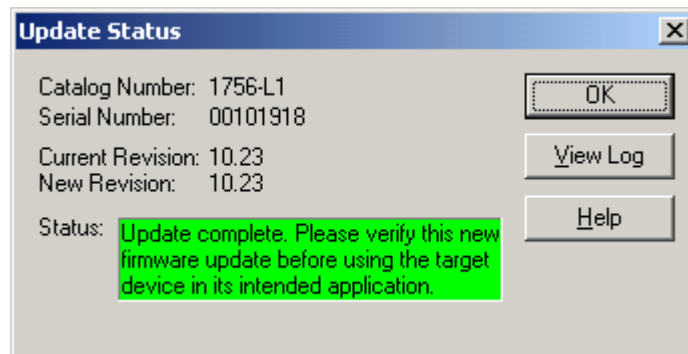
**DANGER:** The target module is about to be update with new firmware. During the update the module will be unable to perform its normal control function. Please make sure that all processes affected by this equipment have been suspended and that all safety critical functions are not affected. To abort this firmware update, press Cancel now. To begin the update now, press Finish.

The firmware update will now take place.



There may be more than one stage in updating a module, so be sure not to interrupt the update process until it is completely finished.

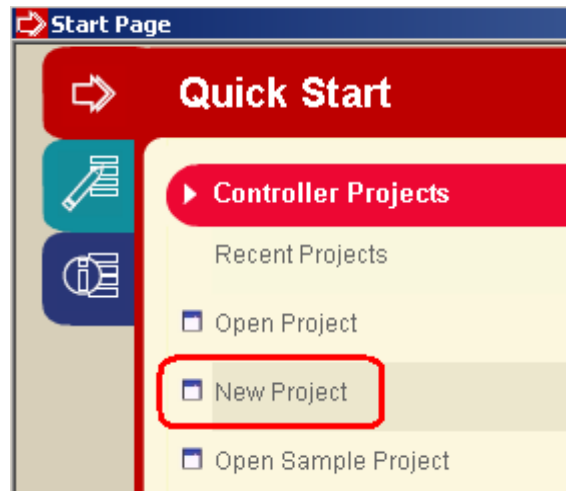
When the update is finished, you will see the following message:



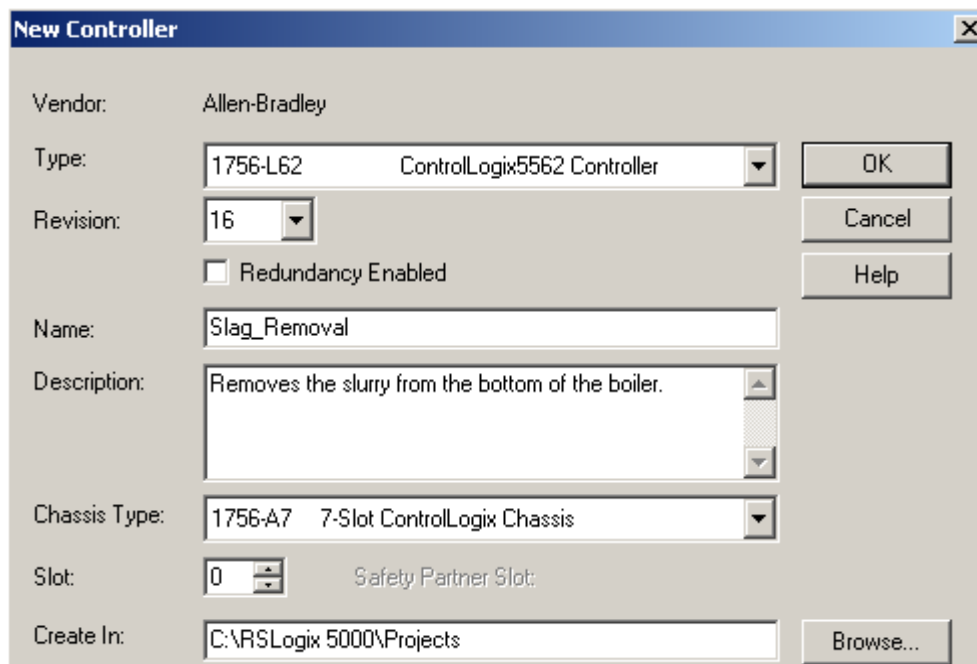
## Creating a new RSLogix 5000 Project

1) Open RSLogix 5000 – You may have a short cut on the desktop, or under Start | Programs | Rockwell Software | RSLogix 5000 Enterprise Series | RSLogix 5000

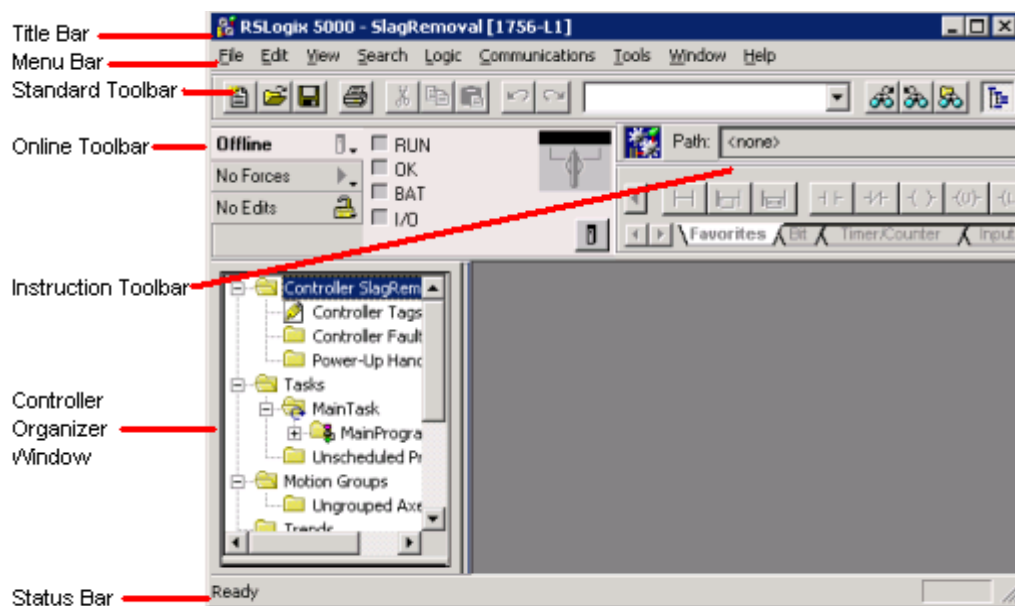
2) Select “New Project”



3) Next, we are going to set up a new processor. The image below is simply for example. You will want to populate the fields according to the specifications of your own local system.

The image shows the 'New Controller' dialog box. The 'Vendor' is set to 'Allen-Bradley'. The 'Type' is '1756-L62 ControlLogix5562 Controller'. The 'Revision' is '16'. The 'Name' is 'Slag Removal'. The 'Description' is 'Removes the slurry from the bottom of the boiler.' The 'Chassis Type' is '1756-A7 7-Slot ControlLogix Chassis'. The 'Slot' is '0'. The 'Create In' field is 'C:\RSLogix 5000\Projects'. The 'Redundancy Enabled' checkbox is unchecked. The 'OK', 'Cancel', and 'Help' buttons are on the right. A 'Browse...' button is next to the 'Create In' field.

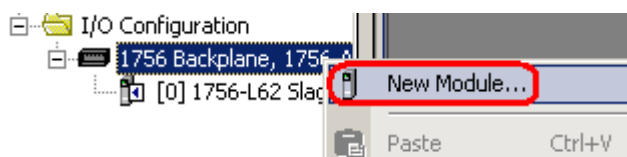
- 4) You have a valid project. It will not do much though because we have not configured any I/O yet. Let's take a moment to discuss some components of the RSLogix 5000 user interface.



- 5) Next we will add the following modules to the I/O Configuration at the bottom of the controller organizer window:

1. 1756-IB16      Version 2.5      Slot 5
2. 1745-OB16E    Version 2.4      Slot 6

- 6) To do this, right click the backplane in the I/O Configuration folder of the project tree, and select 'New Module'.



7) Expand the digital modules, and locate your 1756-IB16 module.

Module	Description	Vendor
+ Analog		
+ Communications		
+ Controllers		
- Digital		
1756-IA16	16 Point 79V-132V AC Input	Allen-Bradley
1756-IA16I	16 Point 79V-132V AC Isolated Input	Allen-Bradley
1756-IA32/A	32 Point 74V-132V AC Input	Allen-Bradley
1756-IA8D	8 Point 79V-132V AC Diagnostic Input	Allen-Bradley
1756-IB16	16 Point 10V-31.2V DC Input	Allen-Bradley
1756-IB16D	16 Point 10V-30V DC Diagnostic Input	Allen-Bradley

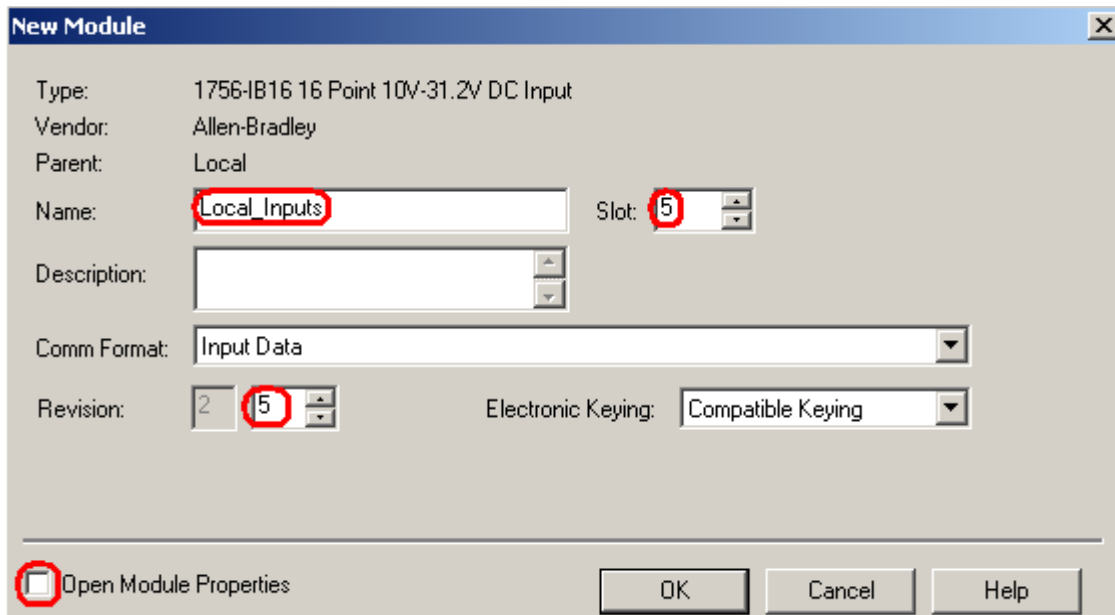
8) The module is version 2.5, so the Major revision is 2, and the minor revision is 5. Click OK on the Major revision dialog box if your revision is correct.

**Select Major Revision**

Select Major Rev for 1756-IB16 Module Profile being Created:

Major Revision:

9) Complete the module properties dialog box as follows, then click '**Finish**'.

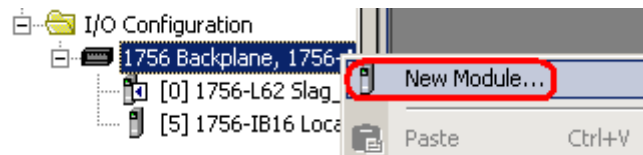


The 'New Module' dialog box is shown with the following settings:

- Type: 1756-IB16 16 Point 10V-31.2V DC Input
- Vendor: Allen-Bradley
- Parent: Local
- Name: Local Inputs (circled in red)
- Slot: 5 (circled in red)
- Description: (empty)
- Comm Format: Input Data
- Revision: 2 (circled in red)
- Electronic Keying: Compatible Keying

At the bottom left, there is a button labeled 'Open Module Properties' with a folder icon, which is circled in red. To the right are 'OK', 'Cancel', and 'Help' buttons.

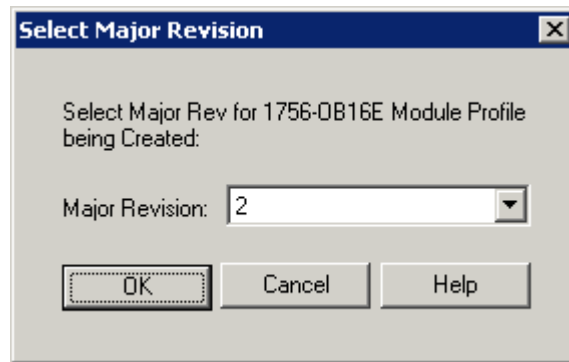
10) You will notice the 1756-IB16 module appears in the I/O Configuration folder. Next, we will add the 1756-OB16E. Right click the backplane in the I/O Configuration folder, and select '**New Module**'.



11) Choose the 1756-OB16E card from the digital modules list, then press **OK**.

Module	Description	Vendor
1756-OA8	8 Point 74V-265V AC Output	Allen-Bradley
1756-OA8D	8 Point 74V-132V AC Diagnostic Output	Allen-Bradley
1756-OA8E	8 Point 74V-132V AC Electronically Fused Output	Allen-Bradley
1756-OB16D	16 Point 19.2V-30V DC Diagnostic Output	Allen-Bradley
1756-OB16E	16 Point 10V-31.2V DC Electronically Fused Output	Allen-Bradley
1756-OB16I	16 Point 10V-30V DC Isolated Output, Sink/Source	Allen-Bradley

12) Since the module is version 2.4, choose 2 as the major revision.

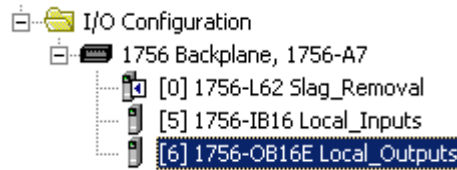


13) Complete the module properties dialog as shown, then press '**Finish**'.

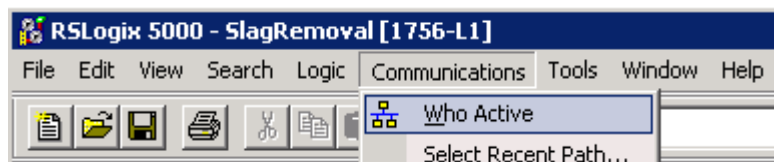
Type:	1756-OB16E 16 Point 10V-31.2V DC Electronically Fused Output		
Vendor:	Allen-Bradley		
Parent:	Local		
Name:	<input type="text" value="Local_Outputs"/>	Slot:	<input type="text" value="6"/>
Description:	<input type="text"/>		
Comm Format:	<input type="text" value="CST Timestamped Fuse Data - Output Data"/>		
Revision:	<input type="text" value="2"/> <input type="text" value="4"/>	Electronic Keying:	<input type="text" value="Compatible Module"/>



14) You will notice both modules are now in the I/O Configuration tree. Be sure to adjust this procedure to suit the modules you are actually using with your own system.



15) You are ready to download. Since the communication path has not been selected yet, let's click Communications | WhoActive From the menu bar



5) Choose the path to your processor, and download.

*Note: When no path has been developed, a 4 step procedure can be used to download to the processor. Make note of this page. If your download fails because a path is needed, follow these 4 steps.*

- A) Click Communications | Who Active
- B) Browse to your **PROCESSOR**
- C) Highlight your **PROCESSOR**
- D) Click **Download**, Upload, or Go Online. In this particular case, we want to download.

You CANNOT download to any module in this case, except for the processor. The processor is the module which stores the PLC program. If your function buttons are gray (in such a way that you cannot click the button), chances are that you do not have the processor highlighted.

## Working With Tags

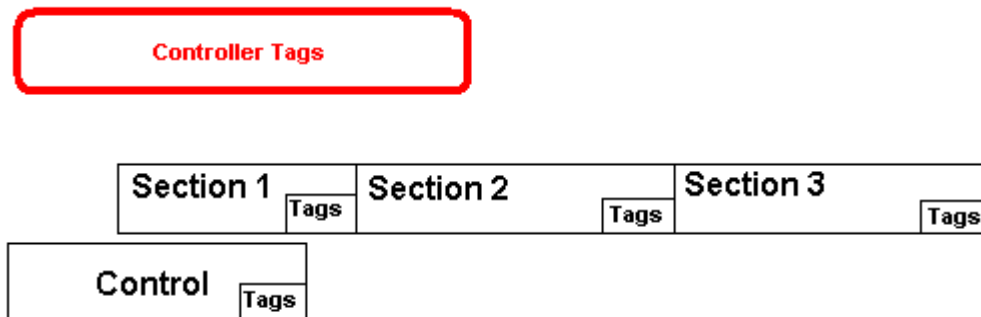
Tags are **variables** that your program reads and manipulates. There are two types of Tags: **Controller tags**, and **Program tags**.

Controller tags are **GLOBAL**, which means that any program or any controller can read or write to the tag. Program tags are **LOCAL** to the program they reside in.

### Controller Tags:

If one program needs to communicate with another program, you would use a Controller Tag that each program can read from or write to. For this example, let's say a conveyor has three segments. Each segment has its own program. When an E-Stop button is pressed, we would want all sections to shut down. Therefore the E-STOP would be a GLOBAL tag.

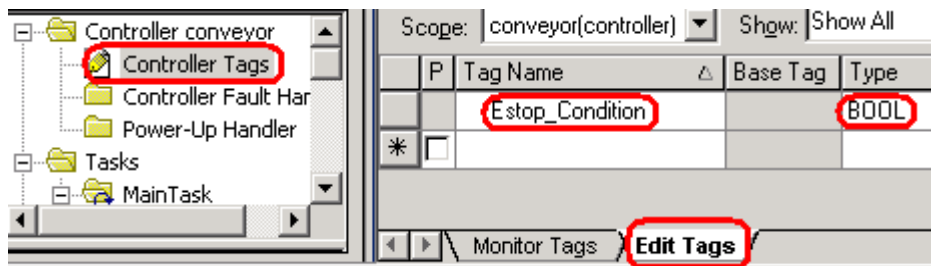
Look at the following example:



Each program has its own tag database, that no other program can access, but if a program writes a value to a Controller Tag, any program can access the tag.

For this example, I've created a new project called conveyor with an input module in slot 5 and an output module in slot 6.

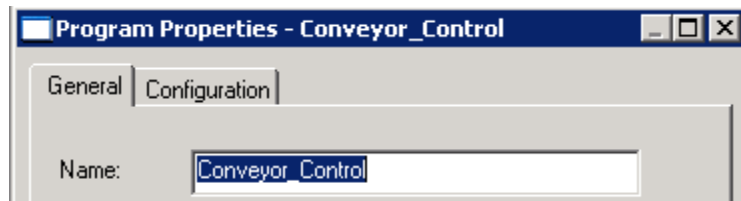
Open the Controller Tag Database, and be sure 'Edit Tags' is selected. Declare the tag called Estop\_Condition, with a BOOL data type. The BOOL data type means one bit of information is to be stored. This is similar to a B3 bit in a PLC or SLC. Press enter.



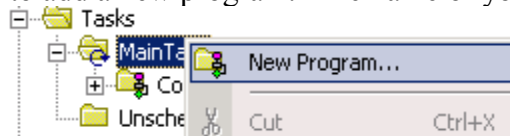
## Program Tags

Program tags are local to the program they reside in. One main advantage of program tags is that we can create one program and copy it multiple times. The exact same program tags can be used in each instance of the program. This makes the process of building logic for similar pieces of equipment very simple.

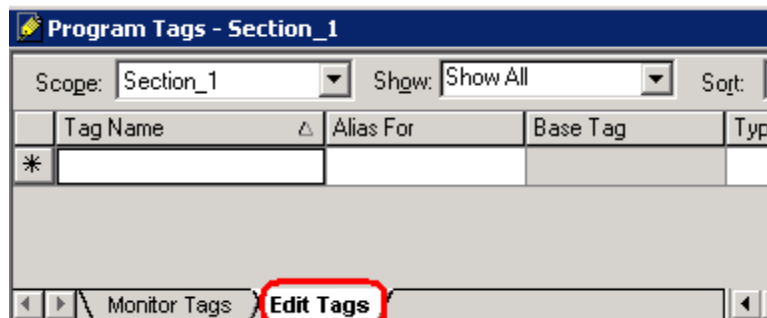
- 1) Right click on the main program and choose 'Properties' Rename MainProgram to Conveyor\_Control.



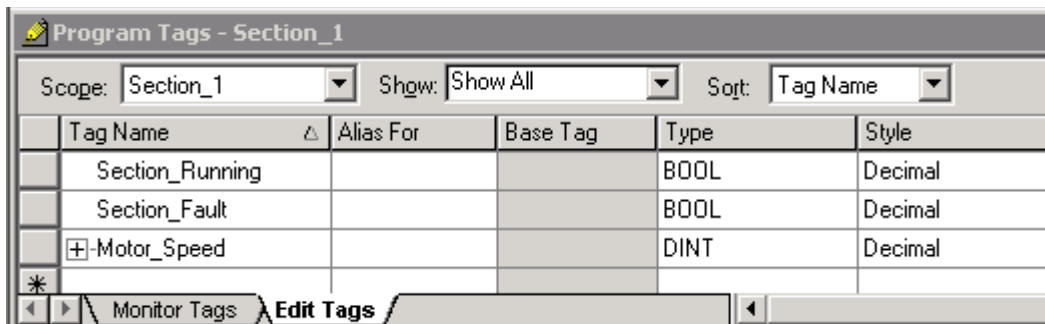
- 2) Right click on the MainTask to add a new program. The name of your program is Section\_1.



- 3) Expand Section\_1 and open the Program Tags. Notice there are no tags in the program tag database. At the bottom of the tag editor screen, be sure 'Edit Tags' is selected.



- 4) Set up three program tags as shown. Pay attention to the Data Type. The data type of the tag specifies the way the data is structured within the tag. A BOOL data type will store 1 bit of information, and a DINT (Double Integer) data type will store 32 bits of information (This is usually used for numbers)

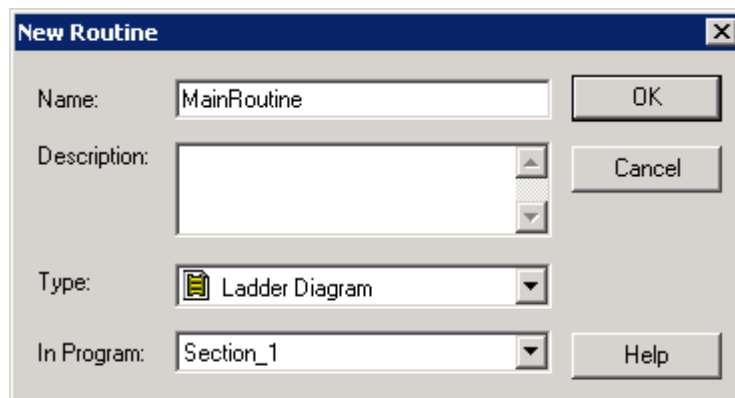


Tag Name	Alias For	Base Tag	Type	Style
Section_Running			BOOL	Decimal
Section_Fault			BOOL	Decimal
+Motor_Speed			DINT	Decimal

- 5) Now, let's add some logic: Right click the Section\_1 program and add a new routine.



- 7) The name will be MainRoutine, then press OK.



**New Routine**

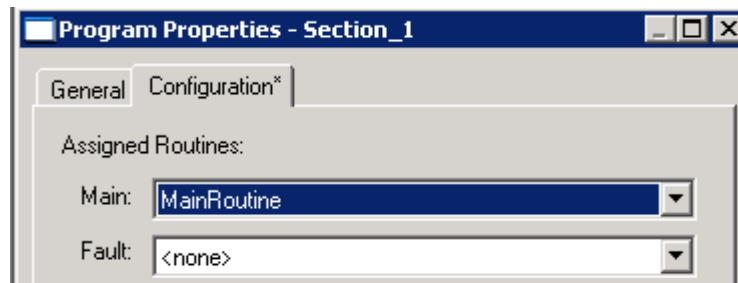
Name:

Description:

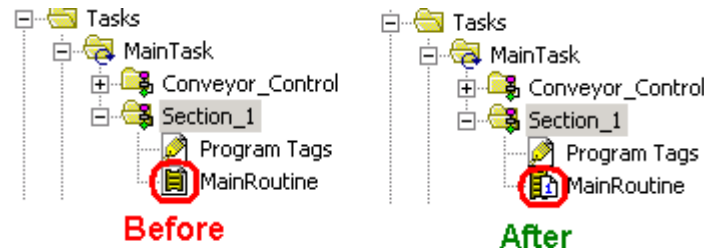
Type:

In Program:

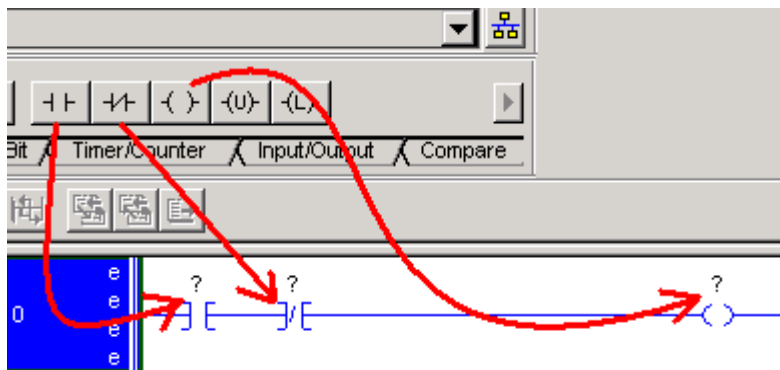
- 8) We named this the MainRoutine, but did not configure it as the MainRoutine yet. Every program needs to have a main routine. That is the routine the processor scans within the program. You will notice in the controller organizer window, there is no 1 on the ladder icon yet. Right click the Section\_1 program and go to properties. Click the Configuration tab. In the Pull down menu next to Main, choose the MainRoutine. Apply your changes, then press OK.



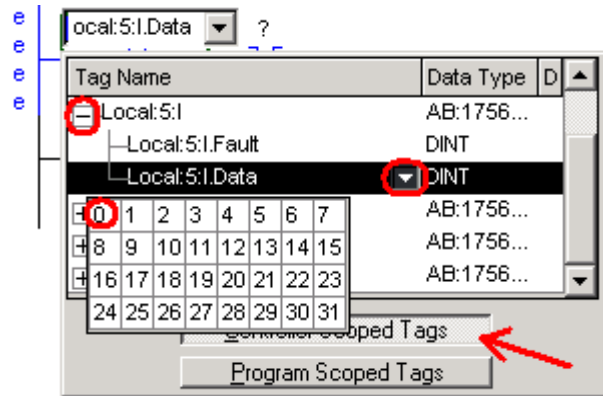
- 9) Notice the MainRoutine is now has a 1 on it...



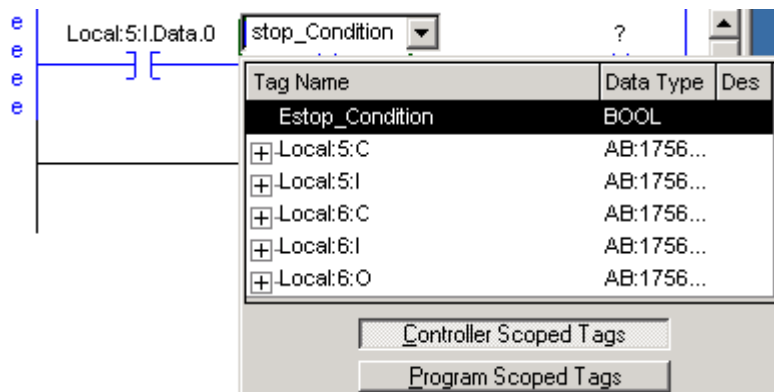
- 10) Open the MainRoutine, and add the following logic using the drag and drop method:



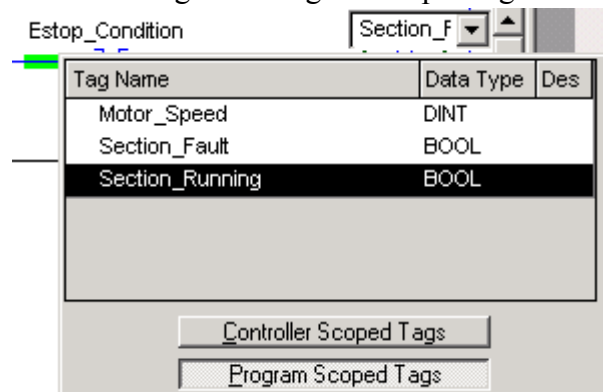
- 11) The question marks indicate that we need an address on each instruction. Double click the question mark on the XIC, and click the pull down tab.
- 12) Be sure Controller Tags is selected. Expand Local:5:I, and click the pull down tab next to data to reveal all 32 bits. Choose bit 0.



- 13) For the XIO, choose the ESTOP Tag.

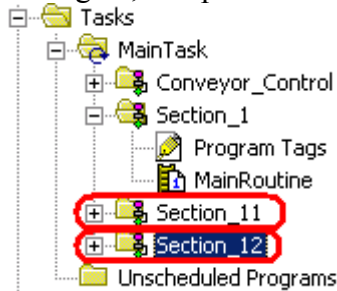


- 14) For the OTE, Choose 'Section Running' as a Program scoped tag.

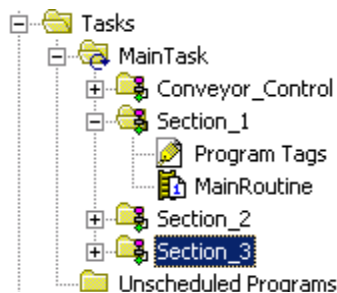


15. Now that we have one complete program (Or as complete as we need it for example... Let's copy the program and use it for other sections of the conveyor.

16) Right click the Section\_1 program and choose 'Copy'. Right click on the 'MainTask' and choose paste. Then right click the 'MainTask' again, and paste a second time.



17) This gave us a total of 3 copies of the same program. Right click on Section\_11, go to the properties, and rename it to Section\_2. Right click Section\_2, go to properties, and rename to Section\_3. When finished, your project will look like this:



18) Now all we have to do is go back and change the I/O in each program to reflect the actual switch that turns on the conveyor for that section, and if we had it set up... The output the logic writes to. All the internal bits are already written for us, and the program structure is in place.



## Aliasing

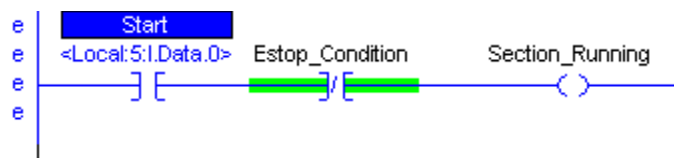
This process can be taken one step further. The program can be set up so nothing has to be changed in logic have to be changed after it has been copied. Aliasing allows this to be done. An Alias is a tag that is a shortcut to another tag. We can create program tags in the program tag database that point to the real world I/O. After a program has been copied, you just need to go into the program tag database, and change the address the aliases point to. Look at this example.

- 1) Open the program tags of Section\_1. Be sure 'Edit Tags' is selected and create a tag called 'Start'. In the Alias for column, make this tag alias for Local:5:I.Data.0.

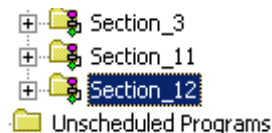
Scope:	Section_1	Show:	Show All	Sort:	Tag
	Tag Name	Alias For	Base Tag	Type	
	Section_Running			BOOL	
	Section_Fault			BOOL	
	+ Motor_Speed			DINT	
	Start	Local:5:I.Data.0(C)	Local:5:I.Data...	BOOL	
*					

Notice the C next to the Input module's address. This means that you are pointing to a controller tag.

- 2) Now go back to the MainRoutine, and change the address on the first instruction to the alias name. Notice by default the actual memory location the alias is pointing to appears below the alias name.



- 3) Now copy section on and paste it twice again into the main task. That will give us sections 11 and 12.



4) Go into section 11, and change the Start alias to point to bit 11 of the input module.

Tag Name	△	Alias For	Base Tag
Motor_Speed			
Section_Fault			
Section_Running			
Start		Local:5:I.Data.11(C)	Local:5:I.Data.11(C)

5) Do the same for section 12. Change the start tag to look at bit 12 of the input module.

6) When bit 11 goes high on the input module, Conveyor 11's start bit will be energized. When Bit 12 goes high on the input module, Conveyor 12's start bit will energize! You can see how aliasing would allow you to quickly develop programs that are very similar.

A Controller tag can be an alias for another controller tag. A program tag can alias another program tag. A program tag can alias a controller tag, but a controller tag CANNOT alias a program tag.

You can alias to several tag levels. The start tag you just created pointed to a bit level tag. But if it would have pointed to the data word instead, we would have to specify the bit number manually in logic. Look at the chart below:

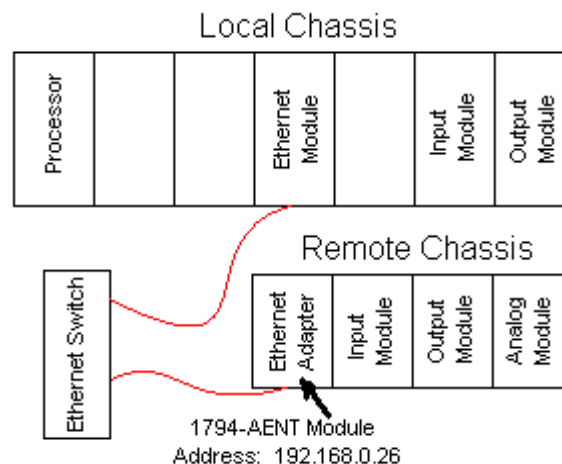
If the name is alias for the displayed part of the address	Then Switch 0's Address in logic would look like this:
<div>Switch</div> <div>Local:5:I.Data.0</div>	Switch
<div>Switch</div> <div>Local:5:I.Data.0</div>	Switch.0
<div>Switch</div> <div>Local:5:I.Data.0</div>	Switch.Data.0

## Remote Chassis

In many systems, many points of I/O are located far away from the **local chassis**. The **local chassis** is the chassis where the processor in focus resides. In many cases, it is easier to mount a chassis at the remote location. A communication cable will allow the processor to control the chassis at the remote location. For example, if 256 points of I/O were 300 feet from the local chassis, it would be easier to mount a chassis at the remote location. You would then run the 256 points of I/O just a few feet to the remote chassis, then run one communication cable back to the local chassis.

The example below shows the Ethernet/IP communication protocol, however, many other protocols follow the same model such as ControlNet or Remote I/O with slightly different wiring and configuration changes.

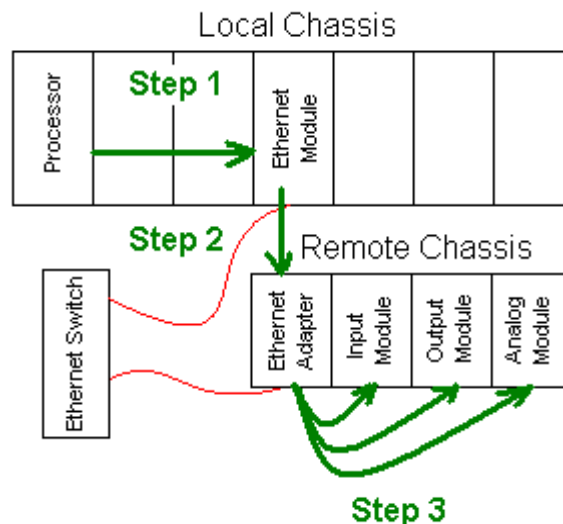
Look at the diagram below. In this example, we have an input and output module in the local chassis. This would be for I/O in the local vicinity. For I/O in another location, we can use a remote chassis. The processor will establish a connection to this remote I/O, and will read inputs, and control outputs on this chassis. In this example, the local chassis is ControlLogix, and the remote chassis is FLEX I/O (This could also be many other types of chassis such as another ControlLogix chassis, PLC-5 Chassis, SLC Chassis, Block I/O, etc....). This procedure will assume that you have an existing program, and that IP addresses have already been assigned to the Ethernet module, and Ethernet Adapter. If you were to use ControlNet, you would assign node numbers instead of IP addresses. You can assign node numbers to these modules by physically dialing in a node number on the modules themselves. For Remote I/O, you would set up the DIP switches according to the user manuals for each module.



## Communication path

Look at the diagram below showing the communication path. The processor is where the program resides, so the path we choose will be relative to the processor itself.

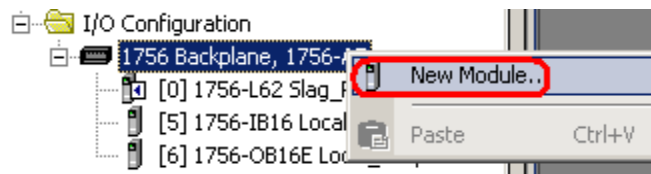
- 1) The processor must first connect to the 1756-ENBT module in the local chassis. If you are using ControlNet, this would be a 1756-CNB(R) module. For the older remote I/O Protocol, this would be a 1756-DHRIO module.
- 2) Next, we must tell the local 1756-ENBT module to connect to the adapter at the remote chassis. Remember we are using Flex I/O for this particular example, so we would connect to the 1794-AENT module (Ethernet) 1794-ACN(R)(ControlNet), or 1794-ASB (Remote I/O).
- 3) The next step is to have the adapter connect to the individual modules within it's chassis. We are using the following modules:
  1. Slot 0 – 1794-IB16 (DC input module)
  2. Slot 1 – 1794-OB16 (DC output module)
  3. Slot 2 – 1794-IE8 (Analog input module)



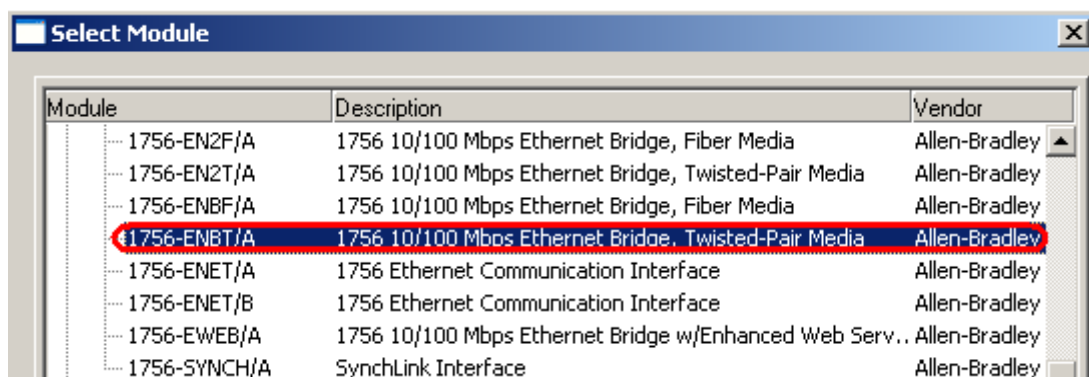
## I/O Configuration

Now that we have decided the layout for our system, we need to go to the ControlLogix project, and set up the remote chassis under I/O Configuration. Recall that 3 steps will accomplish this connection: First, we connect to the Local ENBT module, then the ENBT will connect to the AENT. Next, the AENT will connect to the modules that are in it's own chassis.

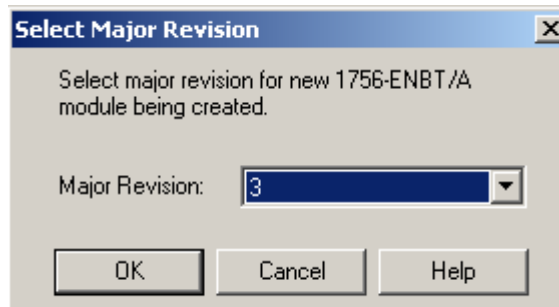
- 1) If you do not already have the Ethernet module set up in your I/O Configuration, right click the backplane of the I/O Configuration folder, and select 'New Module'.



- 2) In the Communication category, select the 1756-ENBT module.



- 3) Next, you will choose the major revision of the 1756-ENBT module. This revision is usually on the label on the side of the module, however, this label may not be up to date. You can use the module information from RSLinx, or type the IP address into your web browser, and click 'Browse Chassis' to determine the revision level. At the time this manual was written, the modules we use for class are 3.6. This means the Major revision is 3, and the minor revision is 6. Therefore, we must select 3 as the major revision. Press OK when finished.



- 4) Next, complete the dialog box for the Ethernet module. Your location may have specific naming standards, but we will name this module '**Local\_Ethernet\_Module**'. The **IP address** scrolls across the front of the ENBT module (assuming an IP address has been assigned). In our classroom, the ENBT module is in slot **3**, and the minor revision is 4 (Recall the module's revision was 2.4, 2 being the major, and 4 being the minor). We will leave the keying as **compatible module**. Press FINISH.

**New Module**

Type: 1756-ENBT/A 1756 10/100 Mbps Ethernet Bridge, Twisted-Pair Media Change Type...

Vendor: Allen-Bradley

Parent: Local

Name: Local\_ENBT

Description:

Slot: 3

Revision: 3 6

Address / Host Name

☒ IP Address: 192 . 168 . 0 . 96

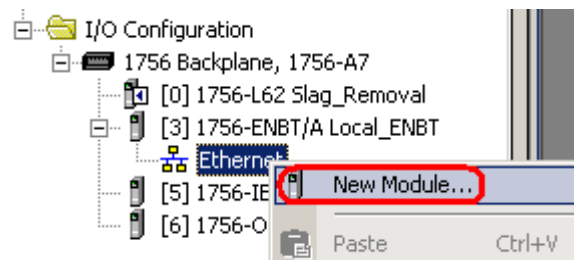
☐ Host Name:

Electronic Keying: Compatible Keying

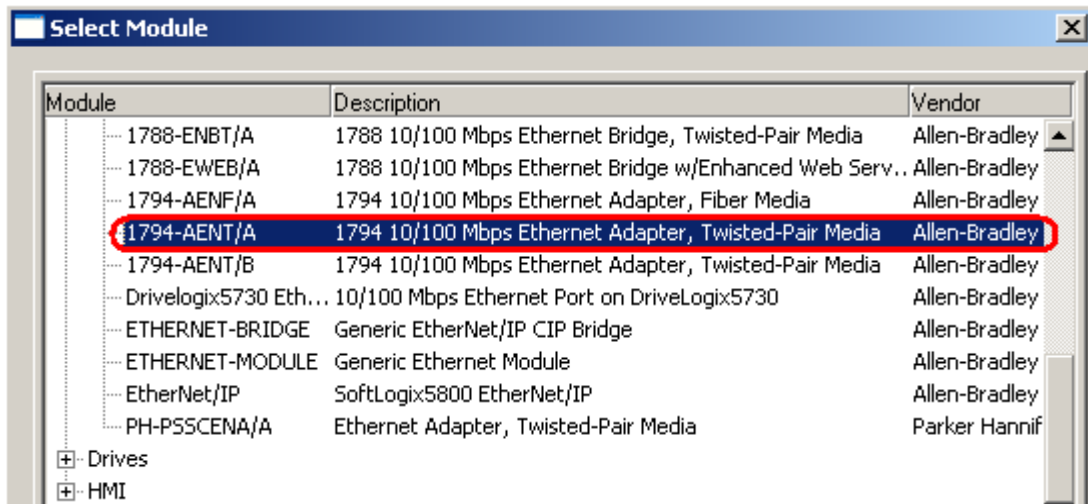
☐ Open Module Properties

OK Cancel Help

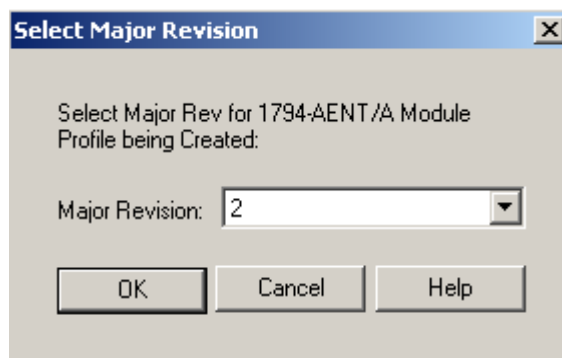
- 5) Our next step is to have the 1756-ENBT connect to the 1794-AENT module over Ethernet. In I/O Configuration, right click the Ethernet network coming from the ENBT module. Remember we are connecting to the AENT module from the Ethernet (Not directly from the backplane)



- 6) Choose the 1794-AENT/A (Series A) from the list of available devices in the Communications category. Press OK. (Be sure you select the **1794-AENT/A**)



- 7) Next, select the major revision of the AENT module. We can get this information from the web browser, by typing the IP address into the browser's address bar, and click 'Module Configuration'. At the time this manual was written, the AENT module had firmware version 2.12. Therefore, we must enter 2 as the major revision.





- 8) You may have naming conventions for remote chassis at your own location. For our classroom use, we will name the module '**Remote\_Chassis**'. You can usually get the **IP address** of this module from your network administrator, schematics, or other documentation if the address was not written on the front of the module. Use the IP address the instructor assigns to you. We will leave the comm format as '**Rack Optimization**', and this will treat the three modules in the chassis as a single connection instead of having a separate connection for each module. The 1756-ENBT module only supports 64 connections in many cases. This chassis consists of **3** slots (not counting the adapter). The minor revision is **12** (because our version was 2.12), and leave the keying as '**Compatible Module**'. Press FINISH when you are finished configuring the module.

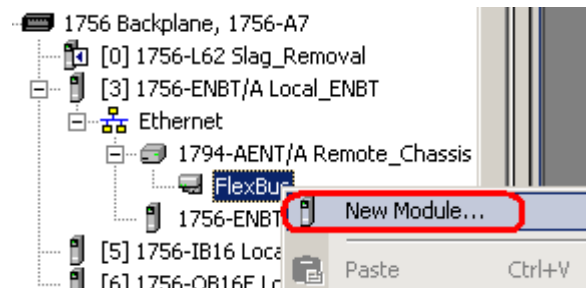
Type:	1794-AENT/A 1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media		
Vendor:	Allen-Bradley		
Parent:	Local_Ethernet_Module		
Name:	<input type="text" value="Remote_Chassis"/>		
Description:	<input type="text"/>		
Comm Format:	<input type="text" value="Rack Optimization"/>		
Chassis Size:	<input type="text" value="3"/>		
Revision:	<input type="text" value="2"/>	<input type="text" value="12"/>	Electronic Keying: <input type="text" value="Compatible Module"/>

Address / Host Name

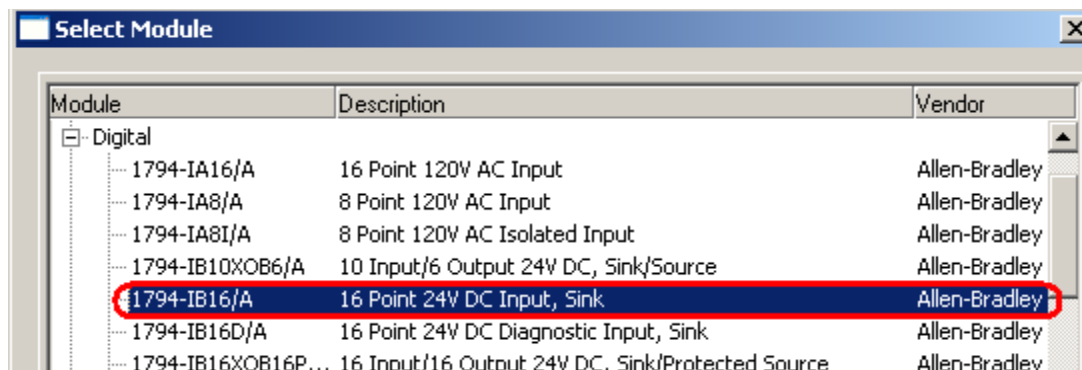
☒ IP Address:

☐ Host Name:

- 9) Now that a connection has been made between the ENBT module and the AENT module, we need to establish a connection between the AENT module and the three modules that are in it's chassis. To add the first module (The 1794-IB16), right click the flexbus of the remote flex chassis in the I/O Configuration, and select 'New Module'.



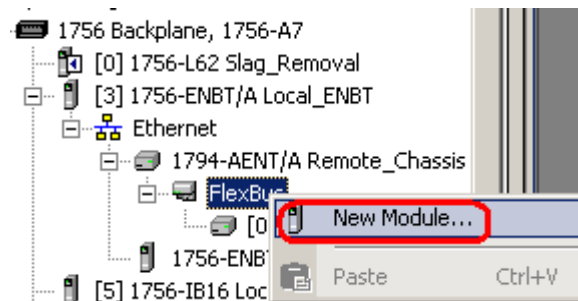
- 10) Choose the 1794-IB16 module from the list of digital modules, then press OK.



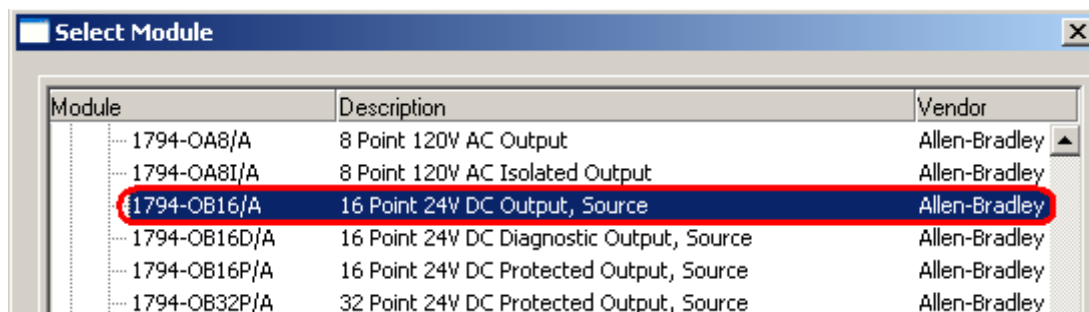
- 11) Complete the Module configuration dialog as follows, then press FINISH.

Type: 1794-IB16/A 16 Point 24V DC Input, Sink  
Vendor: Allen-Bradley  
Parent: Remote\_Chassis  
Name: Remote\_Inputs Slot: 0  
Description:   
Comm Format: Rack Optimization  
Revision: 1 1 Electronic Keying: Compatible Module

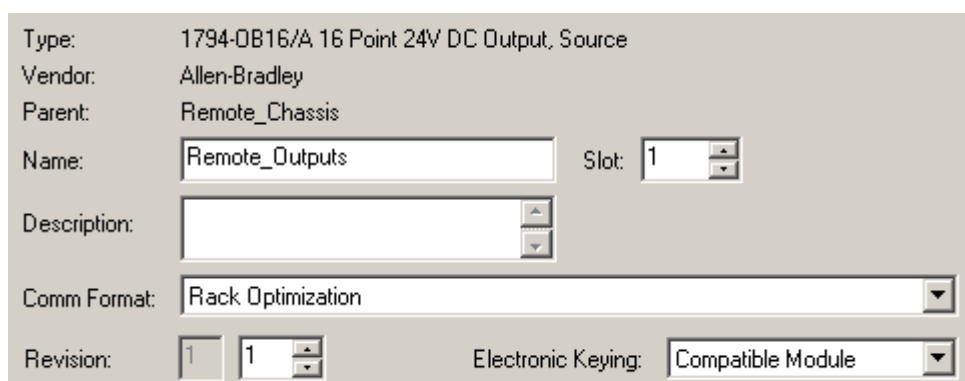
12) Next, we will add the 1794-OB16 module to the I/O Configuration tree. Again, this module resides on the flexbus, so right click the flexbus, and select 'New Module'



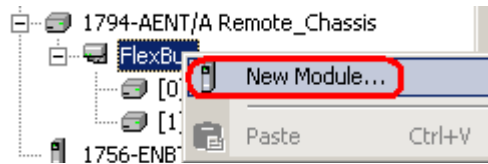
13) From the list of digital modules, select the 1794-OB16 DC output module. Press OK.



14) Complete the module configuration dialog as shown, then press FINISH.



- 15) Next we need to add the last module to the remote chassis. This will be the 1794-IE8. This module resides on the flexbus of the flex chassis. Right click the flexbus, and select 'New Module'.



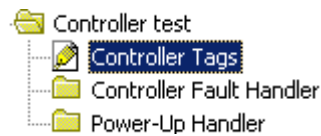
- 16) From the list of analog modules, choose the 1794-IE8 module, then press OK.

Module	Description	Vendor
[-] Analog		
1794-IE12/A	12 Channel 24V DC Non-Isolated Voltage/Current Analog ..	Allen-Bradley
1794-IE4XOE2/B	4 Input/2 Output 24V DC Non-Isolated Analog	Allen-Bradley
1794-IE8/B	8 Channel 24V DC Non-Isolated Voltage/Current Analog I...	Allen-Bradley
1794-IE8H/A	8 Channel Analog Input /HART	Allen-Bradley
1794-IE8XOE4/A	8 Input/4 Output Channel 24V DC Non-Isolated Voltage/...	Allen-Bradley
1794-IF2XOF2I/A	2 Input/2 Output 24V DC Isolated Analog	Allen-Bradley
1794-IF4I/A	4 Channel 24V DC Isolated Analog Input	Allen-Bradley

- 17) complete the module configuration dialog as shown, then press FINISH.

Type:	1794-IE8/B 8 Channel 24V DC Non-Isolated Voltage/Current Analog Input		
Vendor:	Allen-Bradley		
Parent:	Remote_Chassis		
Name:	Remote_Analog	Slot:	2
Description:			
Comm Format:	Input Data		
Revision:	2	1	Electronic Keying: Compatible Module

- 18) Now that all of our modules have been added to the I/O Configuration, RSLogix has created tags for us in the controller tag database. Let's take a look at the controller tags to see where data from these three modules will appear.



- 19) Take a look at the tag names that RSLogix generated for us.

Tag Name
+Remote_Chassis:0:C
+Remote_Chassis:0:I
+Remote_Chassis:1:C
+Remote_Chassis:1:O
+Remote_Chassis:2:C
+Remote_Chassis:2:I
+Remote_Chassis:I
+Remote_Chassis:O

- 20) Notice the tag names assume the name of the adapter in the remote chassis. You will also notice for this example, that we have two tags for each slot. The slot number immediately follows the tag name.

Tag Name
+Remote_Chassis 0 C
+Remote_Chassis 0 I
+Remote_Chassis 1 C
+Remote_Chassis 1 O
+Remote_Chassis 2 C
+Remote_Chassis 2 I
+Remote_Chassis:I
+Remote_Chassis:O

- 21) Since we established a connection to each module in the chassis individually, we have two sets of tags for this remote I/O. We have the base tags, which contain most all of the information we need to know about the chassis, and we have the derived tags which follow the same naming convention as the local I/O. The derived tags alias corresponding memory locations in the base tag that reflect the data for it's own slot when possible.

Tag Name	Value	Force Mask
+ Remote_Chassis:0:C	{...}	
+ Remote_Chassis:0:I	2#0...	
+ Remote_Chassis:1:C	{...}	
+ Remote_Chassis:1:O		
+ Remote_Chassis:2:C	{...}	
+ Remote_Chassis:2:I	{...}	
+ Remote_Chassis:I		
+ Remote_Chassis:O		










Derived Tags

Base Tags

- 22) If you go to edit tags, you will see what based tags the derived tags are aliasing.

Scope:	<input type="text" value="test(controller)"/>	Show:	<input type="text" value="Show All"/>
P	Tag Name	Alias For	
	<div><div></div><div>+ Remote_Chassis:0:C</div></div>		
	<div><div></div><div>+ Remote_Chassis:0:I</div></div>	Remote_Chassis:I.Data[0]	
	<div><div></div><div>+ Remote_Chassis:1:C</div></div>		
	<div><div></div><div>+ Remote_Chassis:1:O</div></div>	Remote_Chassis:O.Data[1]	
	<div><div></div><div>+ Remote_Chassis:2:C</div></div>		
	<div><div></div><div>+ Remote_Chassis:2:I</div></div>		
	<div><div></div><div>+ Remote_Chassis:I</div></div>		
	<div><div></div><div>+ Remote_Chassis:O</div></div>		

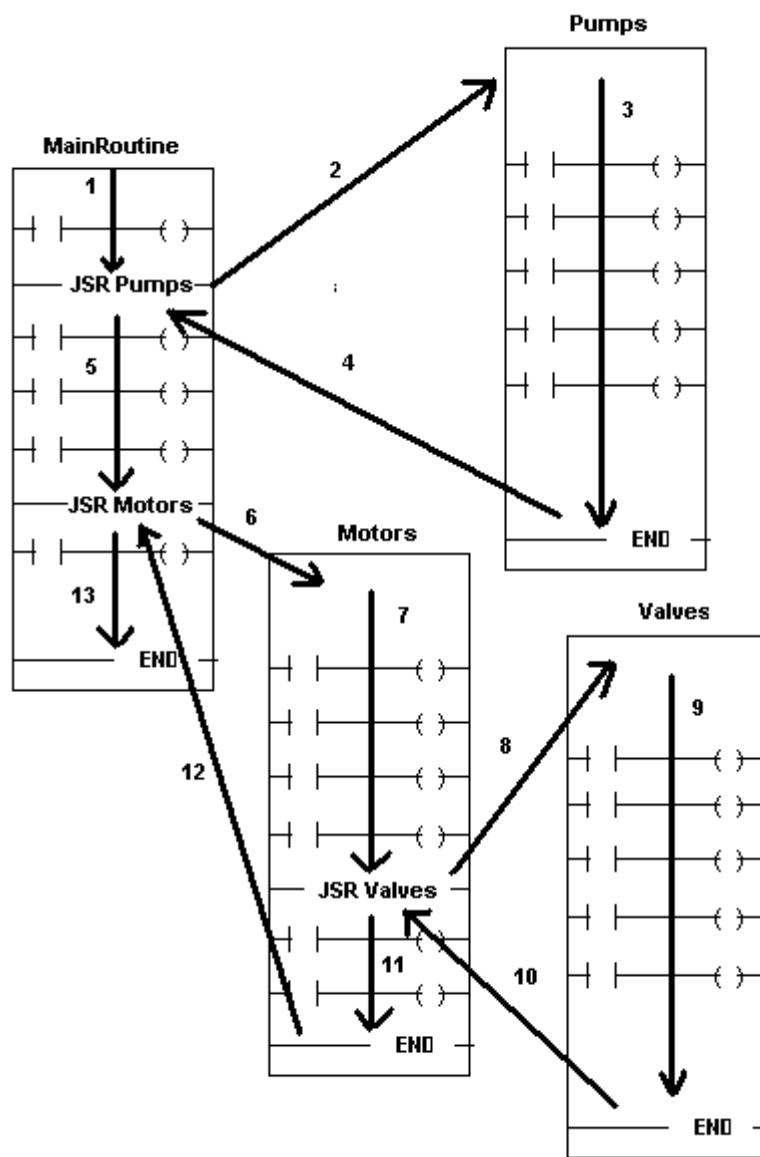
- 23) For this course we will be using the derived tags. To see the data from the DC Input Module, you must be on line, on the monitor tags tab, then expand Remote\_Chassis:0:I. You will see each input from this module. These tags can be used directly in ladder logic, or you can create another alias to use for your project.

	Tag Name 	Value 
	Remote_Chassis:0:C	{ ... }
	Remote_Chassis:0:I	2#0...
	Remote_Chassis:0:I.0	0
	Remote_Chassis:0:I.1	0
	Remote_Chassis:0:I.2	0
	Remote_Chassis:0:I.3	0
	Remote_Chassis:0:I.4	0

## Simple Subroutines

Subroutines can be used to organize a program into smaller sections for organization and ease of troubleshooting. They can also be used to improve processor scan time. For example: If a group calculations only need to be performed at midnight each day for reporting purposes, the logic which performs these calculations can be placed in a subroutine which is only executed at midnight each day. Subroutines can also be used for logic which might need to execute several times throughout a program scan, such as conversions from Celsius to Fahrenheit.

The processor will ONLY scan the Main Routine of a program by default. If we wish for other routines to be executed, we must instruct the processor to do so. The **JSR** instruction is used for this purpose. For this course, we will simply use the JSR instruction to scan other subroutines, however the JSR instruction can be used to pass data to these routines as well.



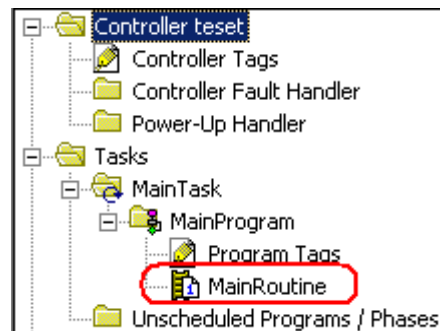


Let's take a close look at this diagram

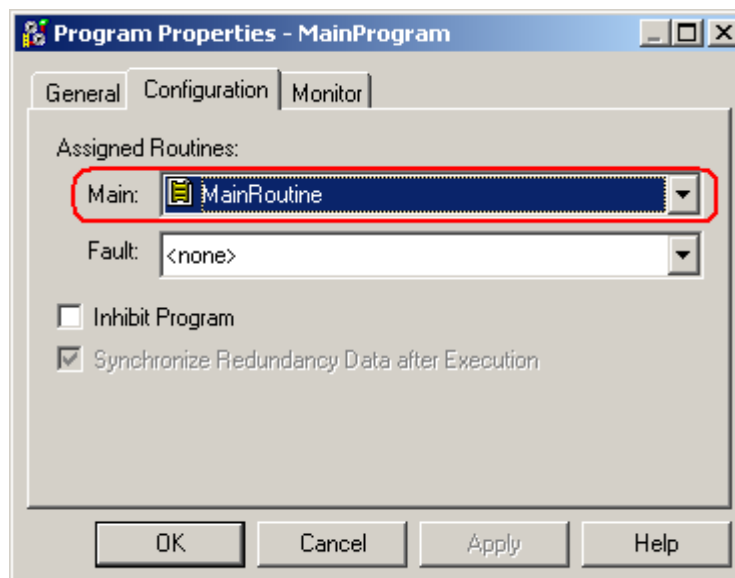
- 1) The processor begins its scan in the “MainRoutine” of a program.
- 2) The processor will then jump to the top of the “Pumps” Routine.
- 3) The processor executes the entire “Pumps” Routine
- 4) The processor returns to the very next instruction in the “MainRoutine”
- 5) The “MainRoutine” is executed until another **JSR** statement is found.
- 6) The processor then jumps to the top of the “Motors” routine.
- 7) The “Motors” routine is executed until a **JSR** statement is found
- 8) Next, the processor goes to the top of the “Valves” routine. (Note: This is called “nesting” subroutines, and there are limitations on how many routines can be nested)
- 9) “Valves” is executed until the end of the routine is reached.
- 10) Now, the processor returns to the JSR statement that called the “Valves” routine.
- 11) The remainder of “Motors” is executed.
- 12) The processor now returns back to the JSR statement from which the “Motors” routine was called.
- 13) The remainder of the “MainRoutine” is executed.

*Note: When passing values to and from subroutines, the SBR statement can be used to receive a value from a JSR statement, and the RET can return a value to a JSR statement. **For the purposes of this class, we will simply be using the JSR statement without RET and SBR statements.***

Now we're going to create some subroutines in RSLogix 5000. Be sure you are looking at the “MainRoutine” of the “MainProgram” of the “MainTask” as shown:



Notice the #1 on the ladder next to the MainRoutine. The name of routines can be changed, so can identify which routine is “main” by this indication. Any routine can be configured as “main” in the configuration tab of the program properties.

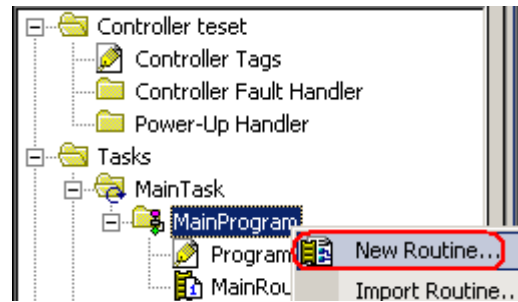


*Note #1: Up to 32 programs can exist in a given task, and each of these programs will have their own main routine.*

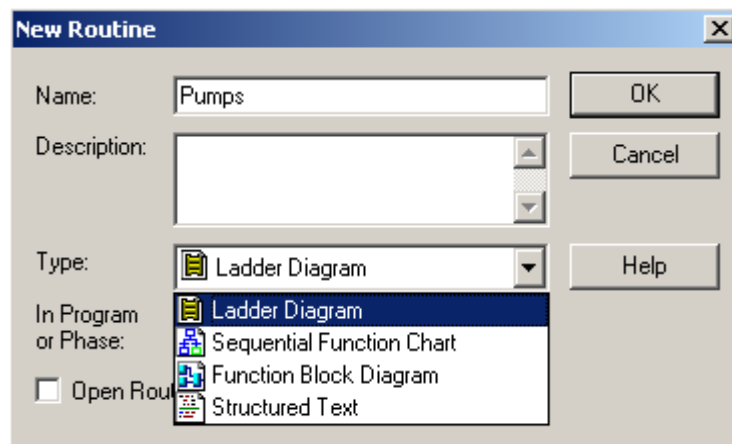
*Note #2: Up to 32 tasks can exist in a controller. Only one of these tasks, however can be continuous (executing every scan). The other tasks are either periodic or event driven.*

Now, let's add our subroutines, and the JSR statements which allow them to execute.

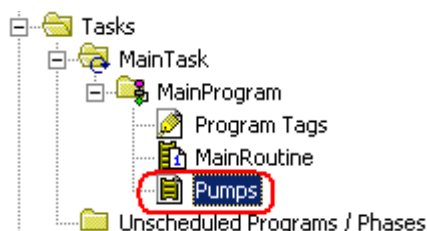
- 1) Right-click the "MainProgram" folder, and choose "New Routine"



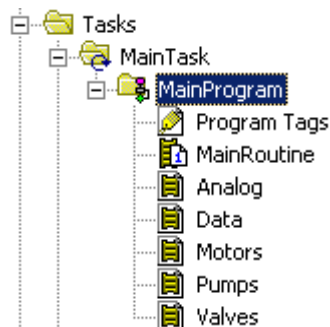
- 2) Name your new routine "Pumps". You will notice there are several programming languages the ControlLogix processor supports. For this course, we are going to be using "Ladder Diagrams"



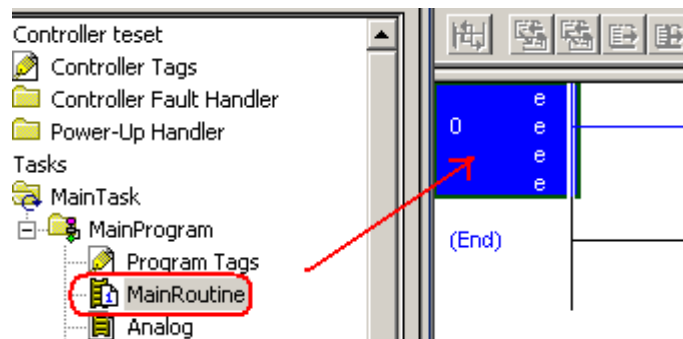
- 3) Press OK, and you will notice your new routine has been added.



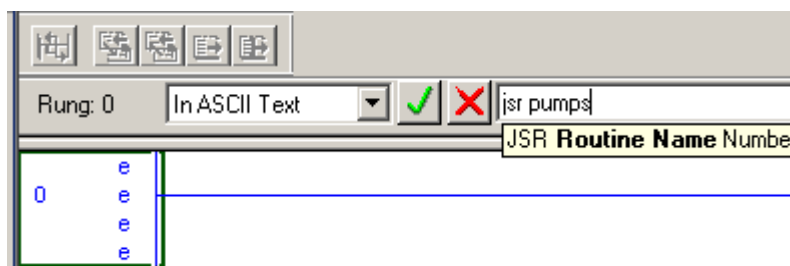
- 4) Repeat this procedure to add the following routines: Motors, Valves, Analog, and Data. When you are finished, your program tree should appear as shown:



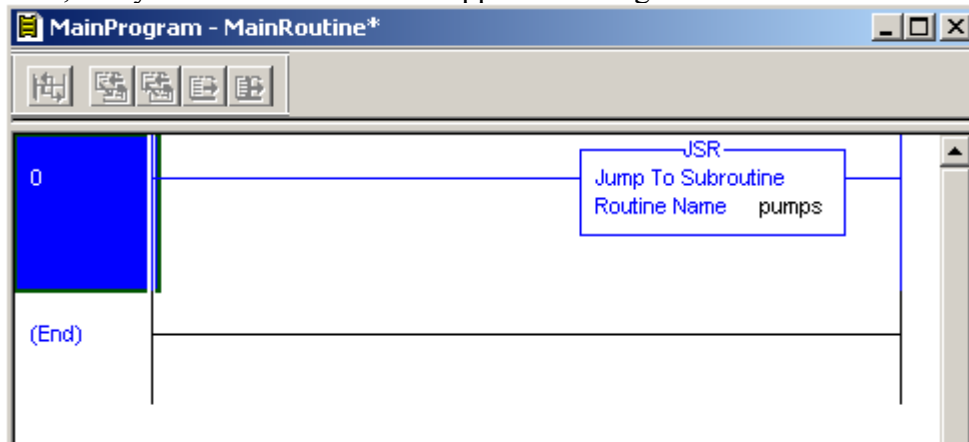
- 5) You will notice your routines appear in alphabetical order. At this point, the only routine the processor would execute is the MainRoutine, because we did not place JSR instructions in the MainRoutine to allow these subroutines to execute. Double click the MainRoutine to view the logic.



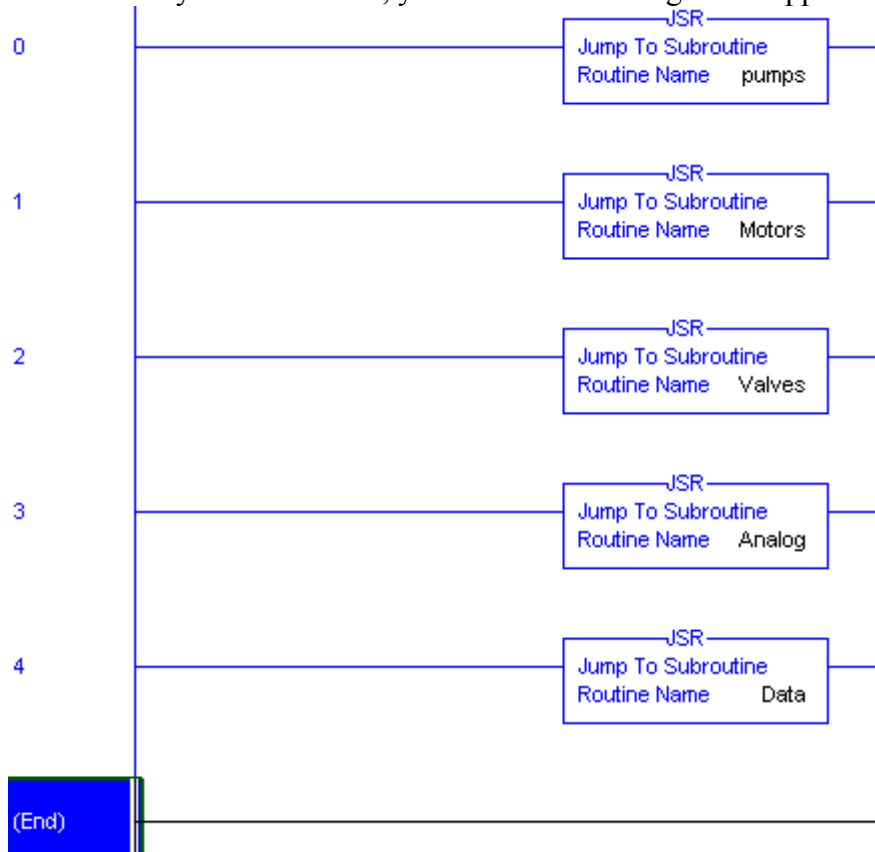
- 6) Now we will instruct the processor to jump to the first routine we created, which is “Pumps”. The processor will execute the routines in the order in which we jump to them. With Rung #0 highlighted, type the following text: jsr Pumps (jsr, then space bar, then Pumps)



- 7) Press enter, and your JSR statement will appear on Rung #0.



- 8) Now, highlight the “End” rung, and repeat this procedure for Motors, Valves, Analog, and Data. Be sure to highlight the “End” rung before you begin to type, or you will overwrite previous work. When you are finished, your MainRoutine logic will appear as shown.



- 9) You are now ready to write logic in these subroutines, and the logic will be executed.

## ***Basic Instructions***

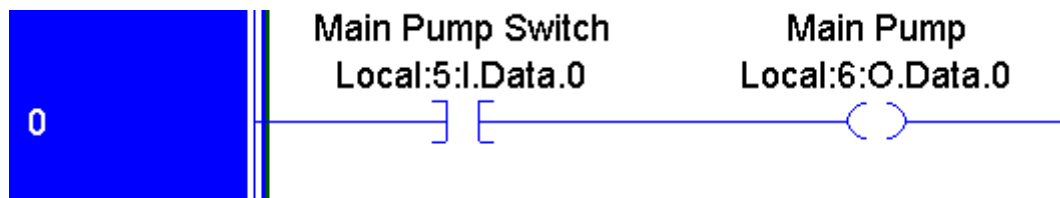
### **A Little History**

A common programming language used in PLC's is called Ladder Logic. Ladder Logic was developed years ago to help electricians adapt to PLC's. Ladder logic is still widely in use today although this language appears to be weakening. Ladder logic is similar to Assembly Language in many ways which was widely used to program computers years ago. Since then, higher level languages such as PASCAL have come along. In the last few years we have seen a more Object Oriented approach to programming in languages such as Java. The ControlLogix processor seems to be following the Object Oriented approach with it's User Defined data types (UDTs), and event driven tasks.

Here are some of the instructions available in Ladder Logic:

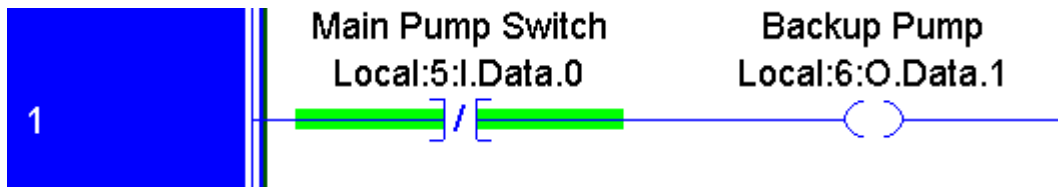
### **Examine If Closed (XIC)**

You will find that most instructions in the SLC, PLC-5, and ControlLogix consist of three character pneumonics. The XIC looks at a given bit of memory in the processor. If this bit is on, then the XIC will intensify indicating logical continuity through the instruction. Here is what the XIC looks like in logic.



## Examine If Open (XIO)

The XIO is just the opposite of the XIC instruction. The XIO looks at a bit in memory. If the bit is a 0, then the XIO is true. It will intensify indicating logical continuity through the instruction, and the next instruction in the rung will be examined. This is usually referred to as a NOT instruction because the address the instruction points to must NOT be on for the instruction to be true. Here is an example of how the XIO will appear in ladder logic:

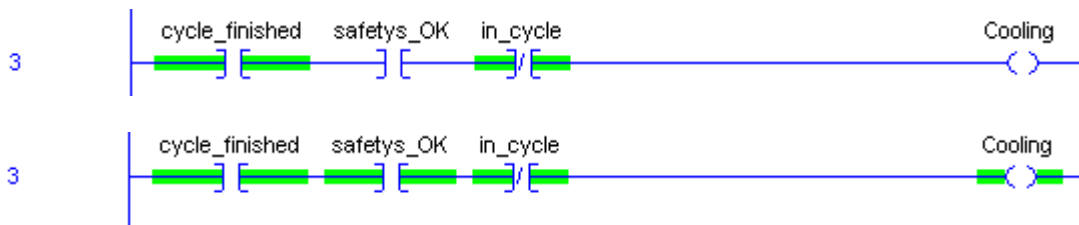


In the above example, you can see that as soon as the Main Pump Switch is shut off, a bit is set to run the backup pump.

## Output To Energize (OTE)

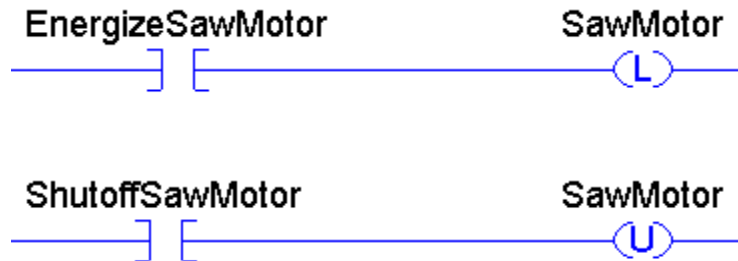
The output to energize simply turns a bit on when it is evaluated as true, and shuts a bit off when the instruction is evaluated as false. Using the same address on an OTE in two different places in the program is considered bad programming practice. The two OTE's can interfere with each other, and makes troubleshooting difficult.

Below you will find two different states of the same rung. The first state shows the rung as false, so a zero is written to B3:1/0. The second state is true, and a 1 will be written to B3:1/0.

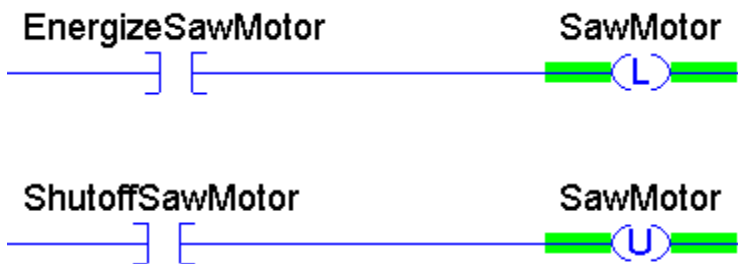


## Output To Latch (OTL) and Output To Unlatch (OTU)

The Output To Latch instruction will write a 1 to it's address when true. When the OTL goes false again, the output address will remain a 1 until another instruction such as the Output to Unlatch shuts it back off. ***This is true even if the processor powers down, and is brought back up!!*** You must use caution when using the Latch/Unlatch when controlling real world devices. Here is what the Latch/Unlatch will look like in logic:



If the output address is off, both the latch and unlatch instructions are not intensified, but once the bit is turned on, you will see both the latch and unlatch intensified even though both inputs are shut off.



Due to the processor scan cycle, since the unlatch is placed after the latch, if both inputs were to go true, the Unlatch instruction would win, and the output address will be shut off. If the latch was after the unlatch, then the latch would be the last instruction scanned, and therefore the bit would be left in the energized state.



## Timers

Timers are generally used for delaying an event from taking place, or to delay a device from shutting off either on an on transition or an off transition. There are three types of timers: The Timer ON delay (TON), Timer Off delay (TOF), and the Retentative Timer On delay (RTO).

Timers can be created as Controller Tags or Program Tags. A tag of the TIMER data type consists of the following components: Preset word (PRE), Accumulate word (ACC), Done bit (DN), Timer Timing bit (TT), and Enable bit (EN). For Timers, the Enable bit follows the rung condition.

<input type="checkbox"/>	<input type="checkbox"/> MotorDelay			TIMER	
	<input type="checkbox"/> MotorDelay.PRE			DINT	Decimal
	<input type="checkbox"/> MotorDelay.ACC			DINT	Decimal
	<input type="checkbox"/> MotorDelay.EN			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.TT			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.DN			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.FS			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.LS			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.OV			BOOL	Decimal
	<input type="checkbox"/> MotorDelay.ER			BOOL	Decimal

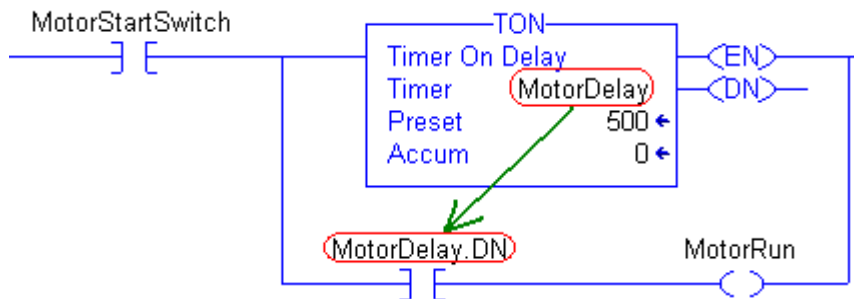
The entire timer is addressed by it's element (example: T4:0) Pieces of the timer can be used in logic however such as the DN bit on an XIC (T4:0/DN), or the Accumulated value in a MOV statement (T4:0.ACC)

## Timer On Delay (TON)

The Timer On delay delays an event from taking place. Once the timer becomes true, the enable bit becomes true instantly. The timer will also start timing instantly, so the TT bit becomes high. Since the timer is timing, the accumulated value will increment.

Once the Accumulated value reaches the preset, the done bit (DN) will go high, and the timer will stop timing. The accumulated value remains at (or near) the preset until the rung goes false again. Here is what a typical timer might look like in logic:

When the switch is energized, the timer will begin timing. When the ACC value reaches the PRE value, the DN bit goes high, and the main motor will start. Since the Time Base is .001, therefore  $5000 \text{ (preset)} \times .001 \text{ (timebase)} = 5 \text{ second delay}$ .



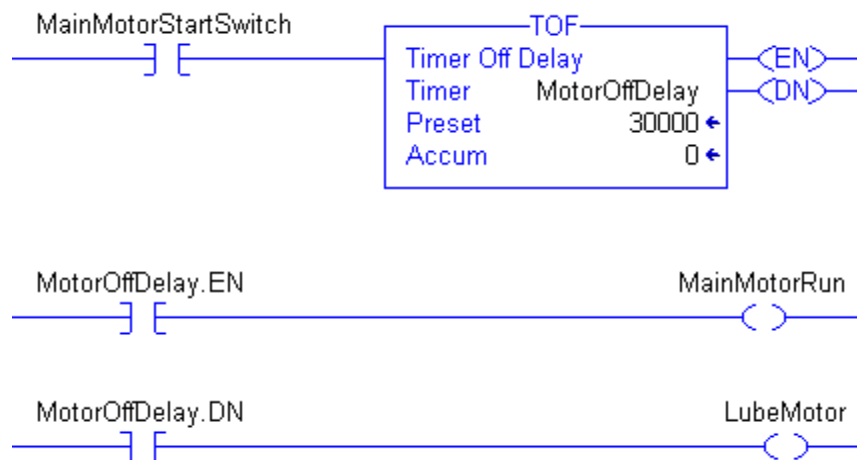
## Timer Off Delay (TOF)

The Off Delay Timer is generally used to delay an event from shutting off. Image a lube system on a large motor. As long as the main motor is turning, the lube pump should be running. When the main motor shuts off, you wouldn't want to shut off the lube pump immediately because the main motor needs time to coast down to zero RPM's. The Main motor could run off the EN bit, and the Lube motor could run off the DN bit.

On the Off delay timer, as soon as the rung goes true, The EN bit goes true as it does for all timers. Since the Off delay timer does not delay the DN bit from shutting off, the DN bit goes high immediately. Remember, the TOF instruction delays the DN bit from shutting off, not turning on. (Plus if we are delaying the DN bit from shutting off, it needs to be high to begin with). While the rung is true, the timer is not timing, and the ACC value is at zero.

When the rung is shut off, the EN bit shuts off immediately. The ACC value will start timing until it reaches PRE then the DN bit will shut off.

Here is what the TOF instruction might look like in logic:

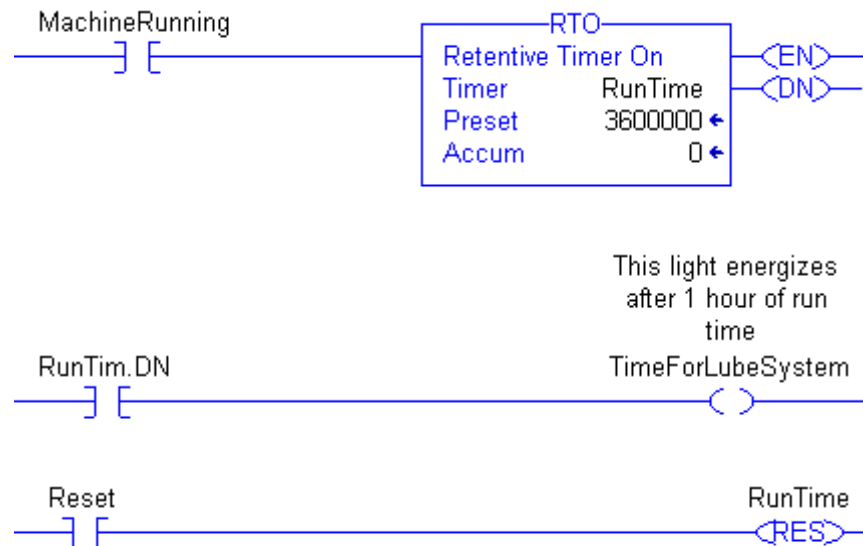


When the motor switch is energized, both the main motor and the lube motor will energize immediately. When the main motor switch is shut off, the main motor shuts off immediately, but since the TOF delays the DN bit from shutting off, the Lube motor will shut off 30 seconds later. Warning: Using the RES instruction on a TOF instruction could cause unpredictable operation.

## Retentative On Delay Timer (RTO)

The RTO instruction works a lot like the TON instruction with one main exception: When the rung goes false on the RTO instruction, it will retain the ACC value. When the rung becomes true again, the ACC value will pick up from where it left off. One good application for the RTO would be an hour meter to indicate total runtime for machinery.

Since the RTO does not reset itself when the rung goes false, the RES instruction must be used to reset a timer. Here is a practical application:



In this example, once the machine accumulates 1 hour of run time, a light might come on indicating that a lubrication needs to be engaged. Once the operator lubricates the machine, he can reset the hour meter.

## Counters

Counters count rung transitions. The CTU runs the accumulated value of the counter up on the false to true rung transition, and the CTD instruction runs the accumulated value down. The CTU and CTD can be used in conjunction with each other.

### Counters consist of the following components:

ACC	Accumulated Value	PRE	Preset Value
CD	Count Down Bit	CU	Count Up bit
OV	Overflow Bit	UN	Underflow bit

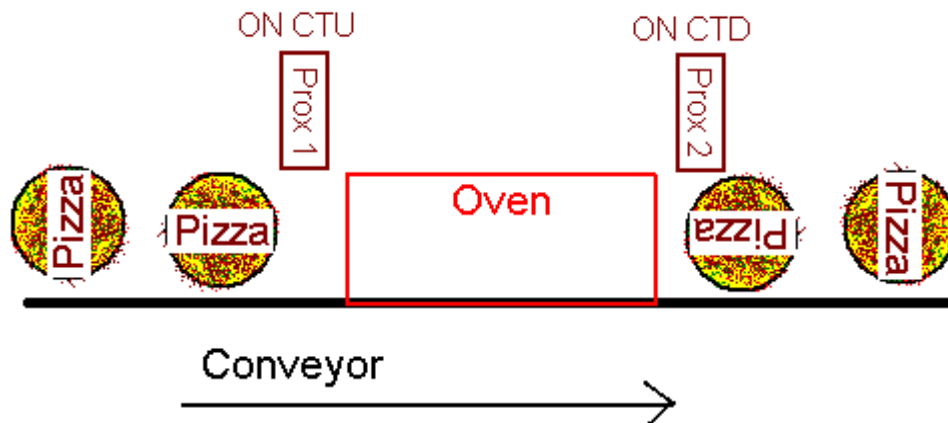
Tags used for counters are declared with the COUNTER data type. Here is an example:

<input type="checkbox"/>	<input type="checkbox"/> PartsCounter			COUNTER	
	<input checked="" type="checkbox"/> PartsCounter.PRE			DINT	Decimal
	<input checked="" type="checkbox"/> PartsCounter.ACC			DINT	Decimal
	<input type="checkbox"/> PartsCounter.CU			BOOL	Decimal
	<input type="checkbox"/> PartsCounter.CD			BOOL	Decimal
	<input type="checkbox"/> PartsCounter.DN			BOOL	Decimal
	<input type="checkbox"/> PartsCounter.OV			BOOL	Decimal
	<input type="checkbox"/> PartsCounter.UN			BOOL	Decimal

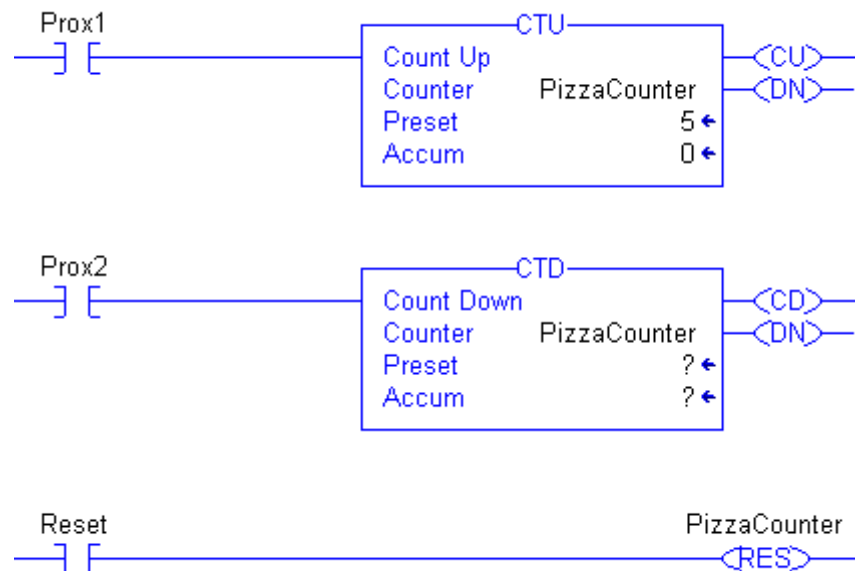
**For the CTU instruction:** The CU bit is high when the CTU instruction is true. The ACC value increments by the value of 1 each time the CU bit goes high. When the ACC reaches the PRE, the DN bit will be set. The CTU will continue to increment the accumulated value until it reaches the maximum possible value for a 32 bit signed integer (2147483647). If the CU bit goes high one more time, the OV bit will be set, and the ACC value will go to -2147483648. Each time the CU bit goes high, the ACC value will still continue to increment (become less negative).

**For the CTD instruction:** The CD bit is high when the CTD instruction is true. The ACC value decrements by the value of 1 each time the CD bit goes high. Any time the ACC is above or equal to the PRE, the DN bit will remain set. The DN bit is reset if the ACC falls below the PRE at any time. The CTD will continue to decrement the accumulated value until it reaches the minimum possible value for a 32 bit signed integer (-2147483648). If the CD bit goes high one more time, the UN bit will be set, and the ACC value will go to 2147483647. Each time the CD bit goes high, the ACC value will still continue to decrement (become less positive).

Here is a practical example of a CTU/CTD implementation:

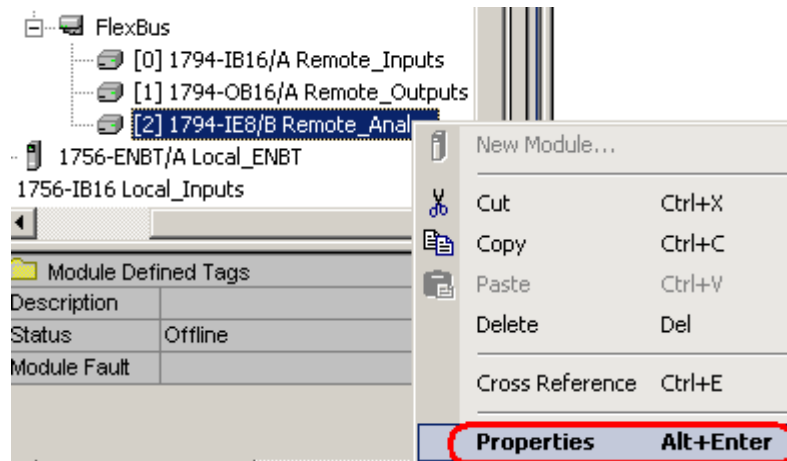


Each time a pizza goes into the oven, the ACC value is incremented by one. Each time a pizza comes out of the oven, the ACC value is decremented by one. Therefore, the ACC value represents how many pizzas are in the oven at any given time. The DN bit could be used to shut the conveyor down if pizzas are going into the oven and not coming out!

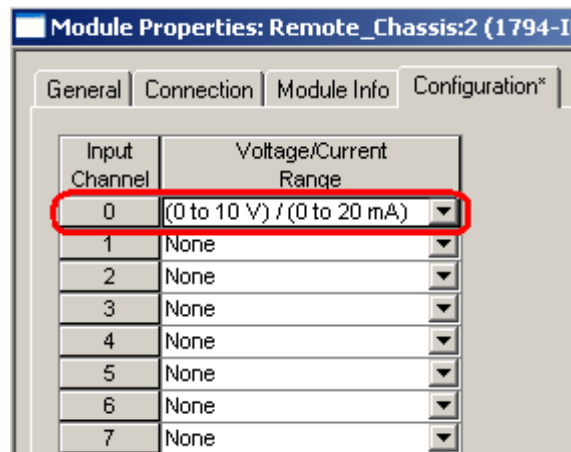


## Analog Configuration

- 1) First, we must configure the analog module for the proper voltage. To do this, right click the 1794-IE8 module in the I/O configuration tree, and select “Properties”



- 2) Now, on the “Configuration” tab, set up channel 0 for 0 to 10 volts as shown:



- 3) Go to the Controller Tag Database, and find the tag for this module. Recall that we named the AENT module “Remote\_Chassis”, so all the tags for the modules on flex are going to begin with this name. The IE8 module is in slot 2 of the Remote\_Chassis, and a potentiometer is an input, so find the tag “Remote\_Chassis:2:I.Ch0Data”.

Scope:
 

Slag\_Removal

Show...

Show All

Name	Value
+ Remote_Chassis:1:C	{...}
+ Remote_Chassis:1:O	2#0000_000...
+ Remote_Chassis:2:C	{...}
- Remote_Chassis:2:I	{...}
+ Remote_Chassis:2:I.Fault	2#0000_000...
+ Remote_Chassis:2:I.Ch0Data	0
+ Remote_Chassis:2:I.Ch1Data	0

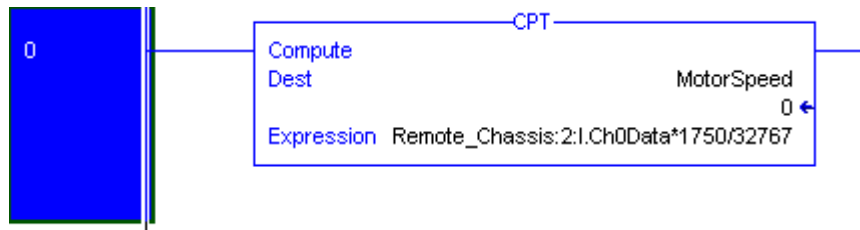
- 4) Spin your pot, and you will notice the data is now changing. With your pot fully counter-clockwise, what is the value in your tag? \_\_\_\_\_ With your pot fully clockwise, what is the value in your tag? \_\_\_\_\_
- 5) Next, we need to scale this data into engineering units. For this example, we will assume that your low value (raw data) was 0, and your high value (raw data) was 32767. We are going to scale this from 0 to 1750 RPMs to represent the speed of a motor. After we perform the scale, we are going to need a place to put the data. Create a tag called “Motor Speed”. It's OK to leave this as a DINT since the high value will only be around 1750. To create this tag, go to Edit Tags, and type “MotorSpeed” as your tag name. Recall that tag names cannot contain spaces.

+	Remote_Chas...		AB:1794_AEN_3...	
+	Remote_Chas...		AB:1794_AEN_3...	
*	MotorSpeed		DINT	Decimal
Monitor Tags		Edit Tags		



## Compute Statement

- 6) Now we are ready to perform a math function on the raw data from the pot to scale it to engineering units. Once we perform this math function, we will store the result to the tag we just created called “MotorSpeed”. The CPT (compute) statement is generally used to scale data into engineering units. For this example, please set up the Compute Statement as shown:

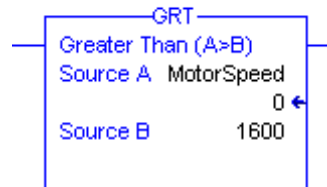


- 7) To scale the data, we multiply the raw data by the maximum number we are scaling to, and divide by the maximum number we are scaling from. In many scaling equations, an offset must also be applied if the raw minimum and scaled minimum do not both start at zero.
- 8) If you are offline, download your work. If you are online, finalize your edits, and when you spin your potentiometer, you should see that the MotorSpeed tag now varies from 0 to 1750 RPM.

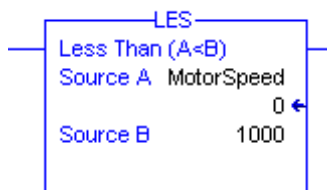
## Compare Statements

There are many compare statements which are commonly used with analog data. Here, we'll list just a few.

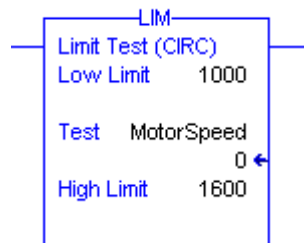
GRT – Greater than – When Source A is Greater than Source B, the statement is true.



LES – Less than – When Source A is Less than Source B, the statement is true.



LIM – Limit – When the test variable is between (or at) the high and the low value, the statement is true. This statement is used to test whether a value lies between two set points. Reversing the low and the high values effectively reverses the instruction in such a way that the instruction is FALSE when the test is between the limits.



## **Using the GSV Command (Accessing the system time)**

The PLC-5 and SLC-500 had a STATUS FILE which could be used by logic at any time simply by referring to the memory location where the time and date were stored.

The system time in ControlLogix works much differently. When a new project is created, no variables exist in the tag databases. If you are going to use a variable in logic, it must first be created first.

The system time in ControlLogix is called the WALLCLOCKTIME object. First an array must be allocated in the tag database, then we must use a GSV (Get System Value) command to continuously load the time from the system into the new array.

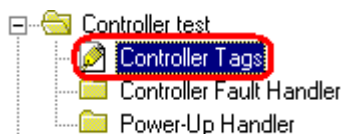
You should access the help file in RSLogix 5000 for a complete description of the WALLCLOCKTIME object. To access the help file, click Help on the menu bar, then click Contents. Click the FIND tab. If this is the first time you are using the Help|Find feature, you may be prompted to select next, then finish to build the help database. Type wallclocktime in step 1, then in step 3, double click 'accessing the wallclocktime object'.

For this example, we are simply going to create an array of seven elements in the controller tag database, then use the GSV command to populate the array with the system time. The purpose of each of the seven elements are as follows:

- Element 0 – Year
- Element 1 – Month
- Element 2 – Day
- Element 3 – Hour
- Element 4 – Minute
- Element 5 – Second
- Element 6 – Microsecond

Let's get started....

- 1) First we need to open the controller tag database at the top of your controller organizer window.



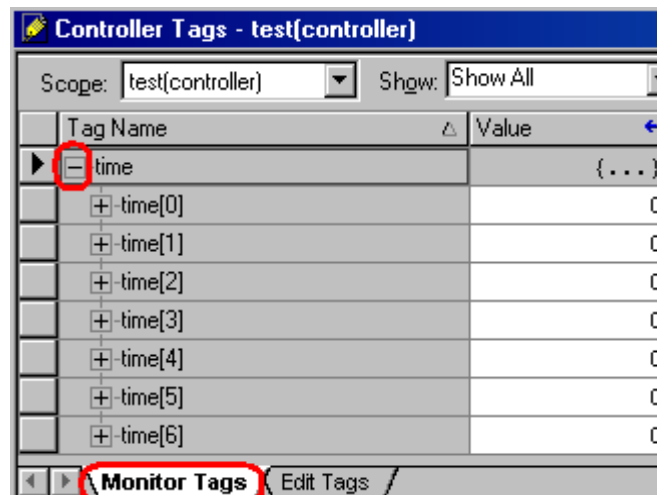
- 2) Next, let's be sure the 'edit tags' tab is selected so we can add a new tag.



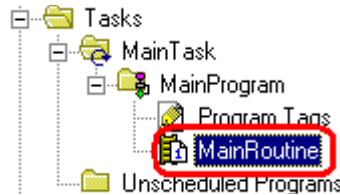
- 3) The new tag we create will be called 'time', and the type will be 'DINT[7]'. Recall that the [7] will create an array of 7 elements. Press enter to accept the tag.



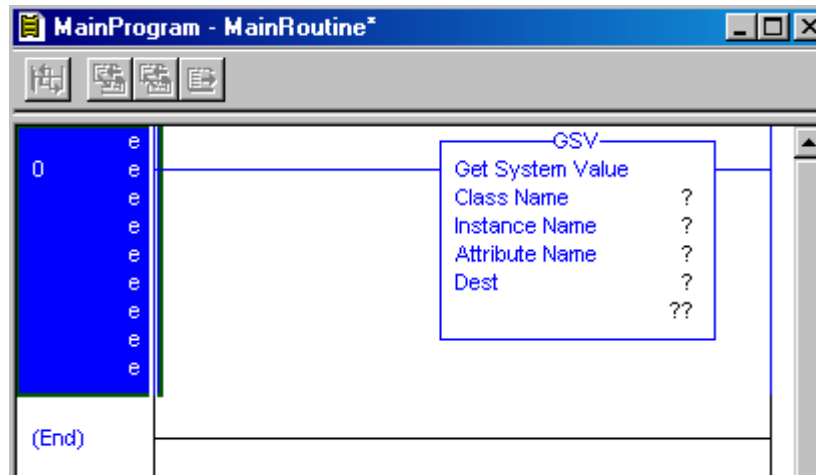
- 4) Let's look at the Array. Click "Monitor Tags". Click the "+" next to the tag name, and you will see that seven elements have been created.



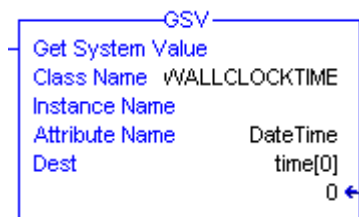
- 5) Next we need to choose which routine we will be adding the GSV to. Recall that the purpose of this GSV is to extract the wall clock time from the system, and load it into the tag we just created. For this example, we will go to the MainRoutine of the MainProgram. You can access the routines from the controller organizer window.



- 6) Be sure the end rung is highlighted, then type GSV (Then press “Enter”) You will notice the GSV command has been added to you logic.

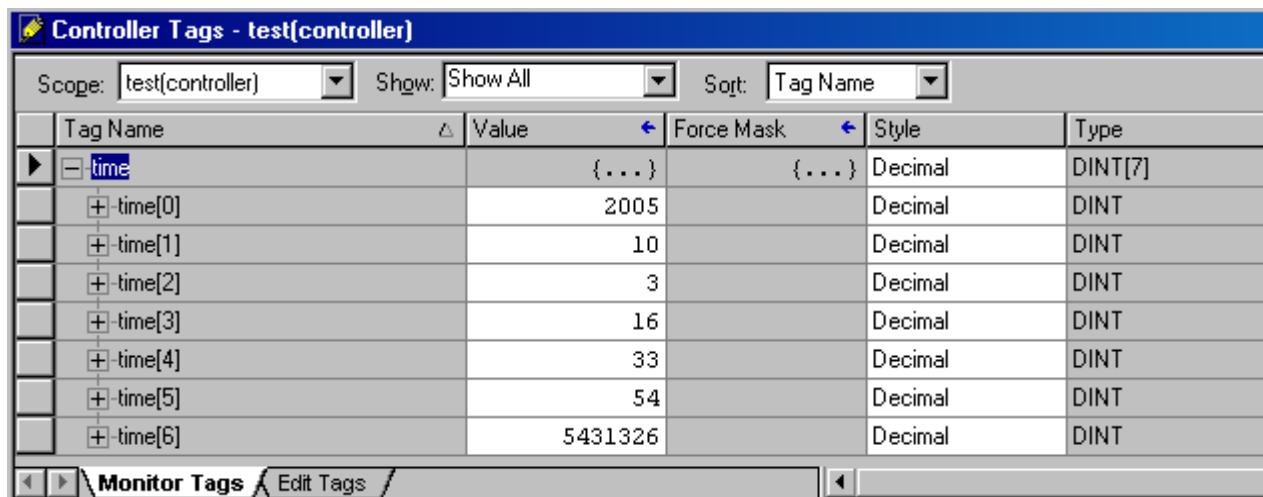


- 7) Double click the “?” next to Class Name. The class will be WALLCLOCKTIME. There is only one instance available of this class, so we don't need to select one. The Attribute will be DateTime. The destination should be time[0].



- 8) Now you can download your work, and go on line in Run mode. If your GSV command is working properly, the GSV command will be extracting the time from the system, and loading the wall clock time (of 7 elements) into our time tag starting with element 0. To test this, go back to your controller tags, Be sure the time tag is still expanded, and you will see data in the following format:

time[0] = Year  
time[1] = Month  
time[2] = Day  
time[3] = Hour  
time[4] = Minute  
time[5] = Second  
time[6] = Microsecond



The screenshot shows a software window titled "Controller Tags - test(controller)". It features a table with columns: Tag Name, Value, Force Mask, Style, and Type. The "time" tag is expanded, showing its array elements from time[0] to time[6]. The values are: 2005, 10, 3, 16, 33, 54, and 5431326 respectively. The "Style" column shows "Decimal" for all elements, and the "Type" column shows "DINT[7]" for the parent tag and "DINT" for the elements. At the bottom, there are tabs for "Monitor Tags" and "Edit Tags", with "Monitor Tags" being the active tab.

Tag Name	Value	Force Mask	Style	Type
time	{...}	{...}	Decimal	DINT[7]
+ time[0]	2005		Decimal	DINT
+ time[1]	10		Decimal	DINT
+ time[2]	3		Decimal	DINT
+ time[3]	16		Decimal	DINT
+ time[4]	33		Decimal	DINT
+ time[5]	54		Decimal	DINT
+ time[6]	5431326		Decimal	DINT



## ***On line Editing for ControlLogix***

There are five basic steps in performing an edit on line.

- 1) Start Edits,
- 2) Make Changes,
- 3) Accept edits,
- 4) Test Edits, and
- 5) Assemble edits.

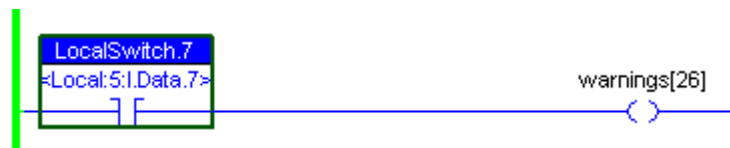
**Note:** Beginning with Version 13, '**Finalize Edits**' will accept, test, and assemble all in one step. Unless you are very experienced, it is recommended that you follow the full 5 step procedure.

Although these steps seem very simple there are a few rules to watch out for.

- You cannot change the data type of existing tags. If you create a new tag with the wrong data type, you must delete the tag, and declare it again.
- You cannot make an on line edit if the key switch is in Run Mode.
- You do not need to perform an on line edit to directly change a value in the data table such as the preset of a timer or counter.
- If the processor is in program mode, you do not need to test and assemble after accepting.
- If the processor is in program mode, and a rung is deleted, there is no warning.

Let's walk through the 5 step procedure:

Look at the rung below. Our objective is to transfer control of the output to LocalSwitch.6. If you click on bit LocalSwitch.7 and attempt to make a change, nothing happens.





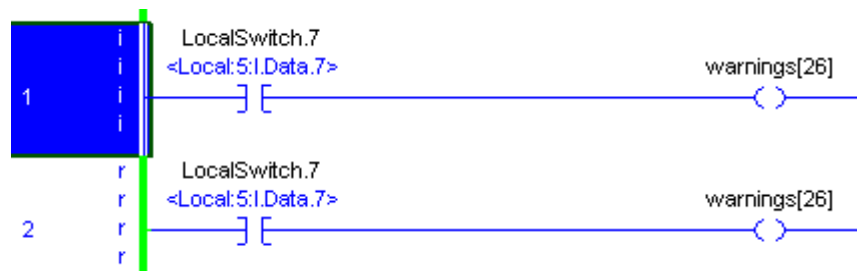
## Step 1) Start Rung Edits

The first step is to put the rung into edit mode. There are several ways this can be done:

- Double click the rung number
- Right click the rung number and start rung edits
- From 'Logic' on the menu bar, click On line Edits, then start pending rung edits
- Click the start rung edit icon in the on line editing tool bar just above the ladder view



Notice that RSLogix made a copy of the rung for us to work with. By looking at the power rails, you can see the bottom rung is being executed by the processor, and the top rung is the one you need to make edits to. You will also notice the e (edit) or i (insert) and r (replace) in the margin are lower case. This means the edits are not in the processor yet. If you are adding new logic instead of modifying existing logic, this is the step where you add a new rung.



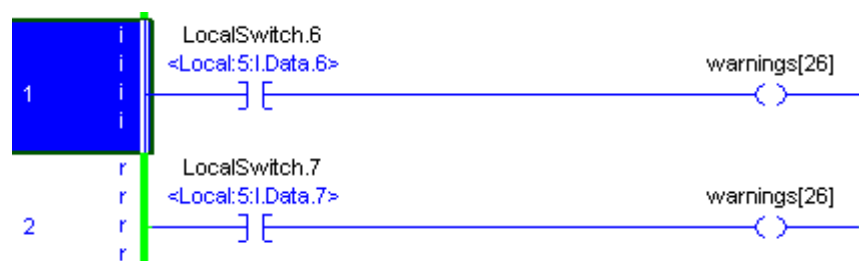
## Step 2) Make Changes

Now that the rung is in edit mode, changes can be made.

If you added a new rung in step #1, this is where you need to add your logic to the new rung.

Be careful not to add any logic that will fault the processor or cause damage to personnel or equipment. Notice the i (insert) and r (replace) zones are in lower case. This means the changes are in RAM only, and have not been sent to the processor.

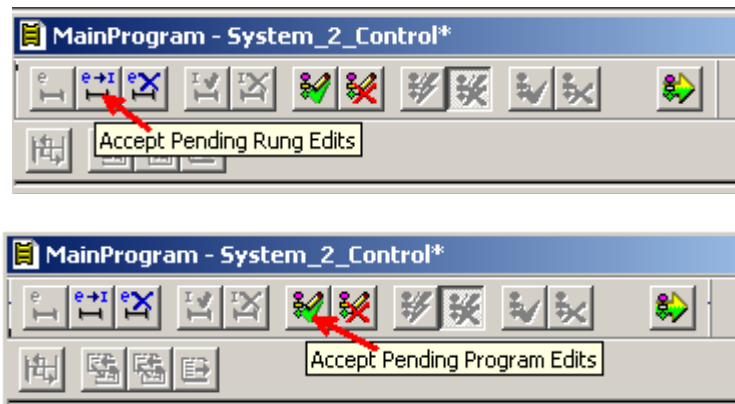
In this example, bit 7 is being changed to bit 6 on the input.



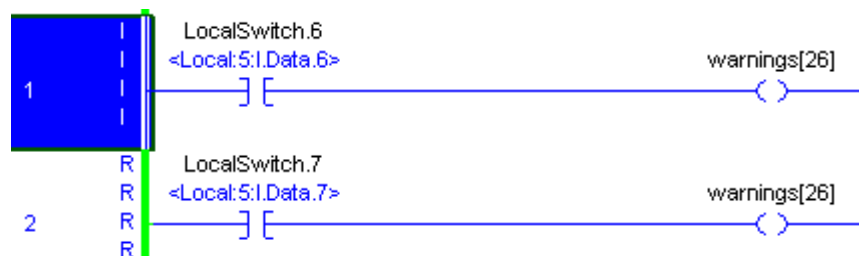
### Step 3) Accept Edits

Now that your rung is set up as you need it, it's time to send the edits to the processor. You can accept pending **rung** edits (This would just accept the **rung** you have selected), or you can accept pending **program** edits (This would accept all the edits in the current **program**) There are several ways to perform the next three steps.

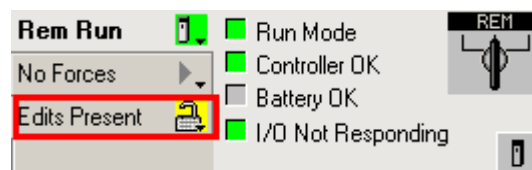
- Right click the rung number, and accept edits
- Click Logic | On line Edits | Accept (rung or program edits) from the menu bar
- Click one of the Accept Edits icons in the on line editing tool bar as shown below



Notice in the margin rung 1 is marked for insertion, and rung 2 is marked for removal. The I's and R's are capital because the edits are now in the processor. Look at the power rails. You can see the old rung is still being executed by the processor.



You will also see that pending edits exist by looking at the on line tool bar.



## Step 4) Test Edits

When you test edits, the new or modified rungs will become active. The old rungs will be left in the processor until we are sure our new rungs are working properly. Be aware that if you change an output address, there might no longer be logic writing to that address. This means that you could abandon a bit in the ON state.

You can test your edits by doing one of the following actions:

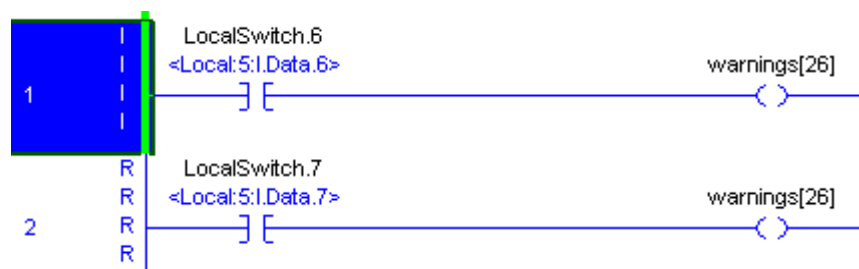
- 1) Right click the rung number
- 2) Choose Logic | On line Edits | Test accepted program edits from the menu bar
- 3) Click the Test icon in the on line edit tool bar above your logic window.



If you are modifying an input type address you should also be careful. If the rung was previously true, you may want to make sure your new logic is also going to be true at the moment you accept, or the the output may shut off.

Let's test the edits, and you will notice the new rung(s) are active. If the edits do not work the way you anticipated, you can un-test to revert to the old rung while you make other changes to the new rung.

Notice the power rails:



## Step 5) Assemble Edits

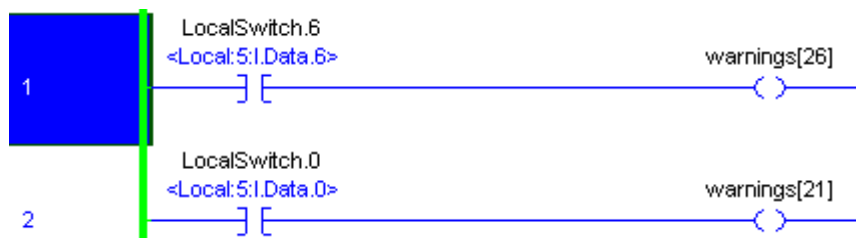
If your logic is working properly, go ahead and assemble the edits. Assembling removes the old rung, and the edit zone markers. After Assembling, you may want to save your work to the hard drive.

You can assemble by using one of the following methods:

- 1) Right click the rung number, and choose accept edits (if available in your version)
- 2) Click Logic | On line Edits | Assemble accepted program edits from the menu bar.
- 3) Click the Assemble Edits icon in the on line edits tool bar.



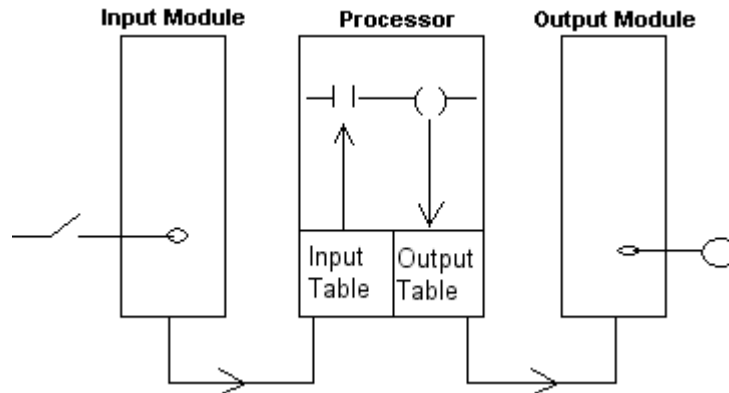
Notice the Logic now appears to be normal:



## ***Forcing I/O***

Forcing can be used for troubleshooting, and to some extent simulates real world jumpers. Leaving forces in the processor, or depending on forced I/O to make your equipment run is considered bad practice.

Look at the diagram below:



Under normal circumstances, the following events take place:

1. The switch is shut
2. A 1 appears in the input tag
3. The XIC instruction goes true
4. The OTE is enabled
5. A 1 is written to the output tag
6. The light will energize on the output module

### **Forcing the input:**

If you place a jumper across the switch, you would have the same effect as the switch always being shut. A 1 would always be in the data table, the logic would be true, and the light would energize. The same effect applies to forcing. Forcing the input on would result in a 1 in the input data table for the switch, and all logic would be executed as if the switch was shut. The opposite applies to an OFF force. An Off force would be similar to cutting a lead on the switch. A zero would result in the input data table.

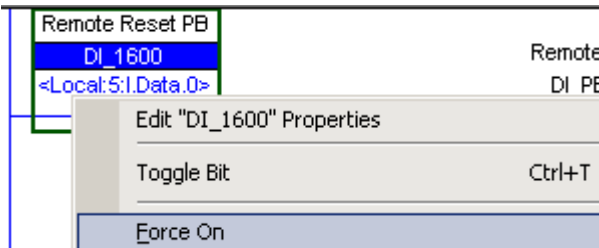
### **Forcing the output:**

If you place a jumper to the output, the output table would still be a zero if the logic is false. Information does not flow from the output device to the output data table. Therefore, any XIC instruction that is looking at the output bit would also be false. The same applies to forcing. If you force an output device, the output data table will still be controlled by the ladder logic.

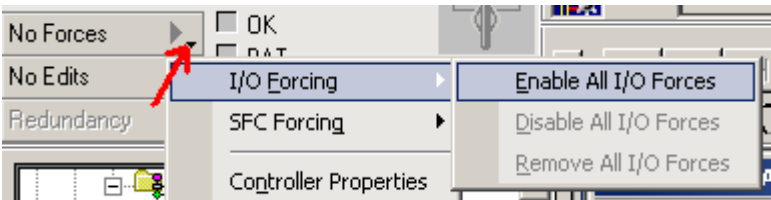
*Note: Even though forcing an output does not directly effect the data table, The field device itself could feed an input back into the processor causing other things to happen in logic. Know your system before using the force feature.*

There are several ways to force I/O. Forcing can be applied from ladder logic, or from the Controller Tag database. Internal memory locations cannot be forced. You can only force real I/O, Aliases to real I/O, or producer/consumer tags.

In this example, we will force an input directly from ladder logic. Right click on the input address, and choose 'Force On'.



Notice the force light on the processor begins to flash (if your process has a force light) indicating that forces are installed, but not enabled. The force can be enabled from the on line tool bar as shown below:



The force light on your processor will now be solid amber indicating that installed forces have been enabled. If we go to the data table, you will see that the input bit is on, and it is red indicating that a force has been enabled on the input. You will also see that the force mask reflects which bits have been forced in the data word. Forcing can be preformed directly from the force mask as well.

[-] Local:5:I	{...}	Forced
[+] Local:5:I.Fault	2#0...	
[-] Local:5:I.Data	2#... 1	2#.....1
[-] Local:5:I.Data.0	1	1

In the force mask, the value of 1 indicates a bit has been forced on. The value of 0 indicates an off force, and a period indicates no force is installed on a particular bit.

## ***Working with User Defined Data Types***

A User Defined Data type allows the user to create his own data structure offline. To understand user defined Data Types, you must first understand pre-defined data types. In the PLC-5, T4:0 was a variable with the timer data type. T4:0 contained several members: T4:0.ACC, T4:0.PRE, T4:0/DN, T4:0/EN, and T4:0/TT.

In ControlLogix, a variable such as lubedelay can be created, and given the data type of “Timer”. Once this assignment is made, the variable “lubedelay” can be expanded to reveal all of it's components: lubedelay.ACC, lubedelay.PRE, lubedelay.DN, lubedelay.TT, and lubedelay.EN. All of these variables are updated by a timer instruction in logic.

The timer object is always on the 'menu' as a pre defined data type, but you don't have any timers by default. You have to set up a variable in the tag database, and assign that variable the timer data type. You would now have a timer.

Instead of using the data structure of a timer, you may want to create your own data structures. For example, you may want to create a data type called 'TankAlarm', and have the alarm data type consist of some members such as HighLevel, and LowLevel.

Now any tag you assign a TankAlarm data type will inherit both the HighLevel and LowLevel members.

Data structures and data types are not directly usable by the logic. Setting up a User Defined Data type merely puts an item on a menu. You don't actually have an instance of the item until you declare a tag with your data type.

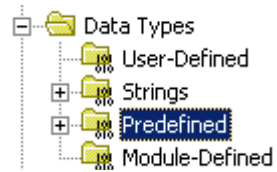
For example: Steak and Shake sells hamburgers (or steak burgers). Seeing the hamburger on a menu does not allow you to utilize the product in any way until you order at least one instance of the menu item. The same is true for the timer example we used earlier. The timer data structure is always available, but you don't have any timers until you create a tag with the timer data type.

Let's take a look at the timer data structure to see what members it consists of, and then we will create our own data structure for TankAlarms.

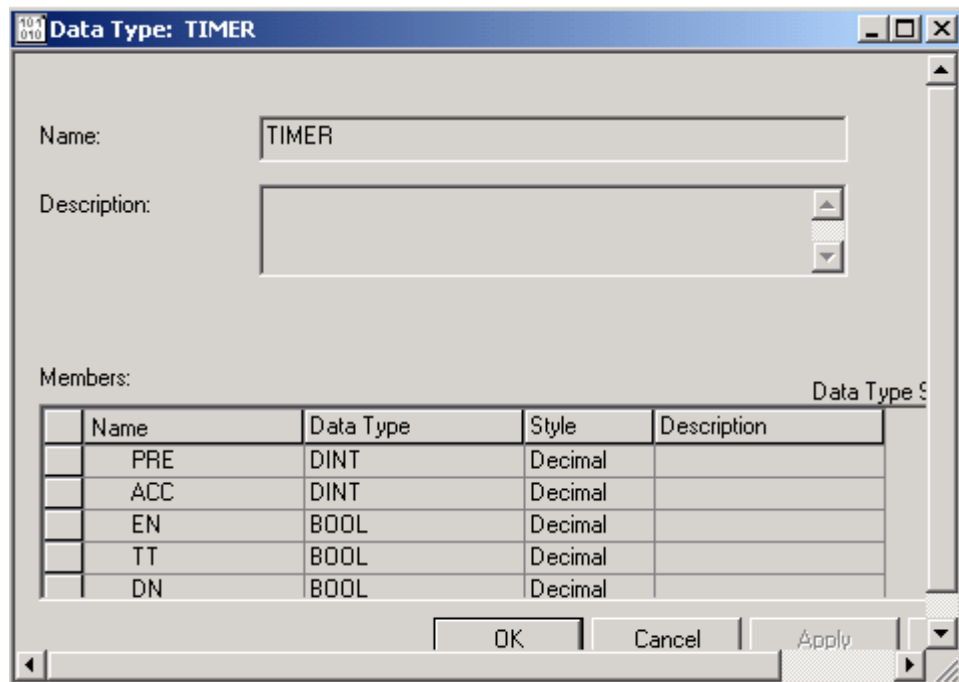


## Simple UDT's

In an open project, locate the Data Types folder in the controller organizer window. Under the Data Types folder, you will find the Predefined data types. Expand the Predefined data types folder, and double click TIMER.



Notice the members of the Timer Data Structure. Any tag we assign in the tag database as a timer will inherit all of these members.

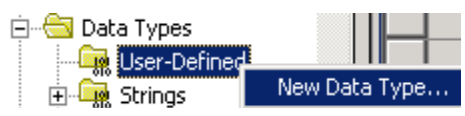


When you create MyTimer with the timer type, the tag can be expanded to reveal all the components.:

Tag Name ▾	Alias For	Base Tag	Type
[-] MyTimer			TIMER
[+] MyTimer.PRE			DINT
[+] MyTimer.ACC			DINT
[-] MyTimer.EN			BOOL
[-] MyTimer.TT			BOOL
[-] MyTimer.DN			BOOL

Next, we are going to create a User Defined Data structure called “TankAlarm”. This data structure will have two members: HighLevel, and LowLevel.

Right Click on the User-Defined folder under Data Types, and select “New Data Type”.



The name of this data type will be TankAlarm. The two members, HighLevel, and LowLevel, will each have a bool data type.

Name: TankAlarm

Description: Alarms for Tanks

Members: 

Data Type Size:

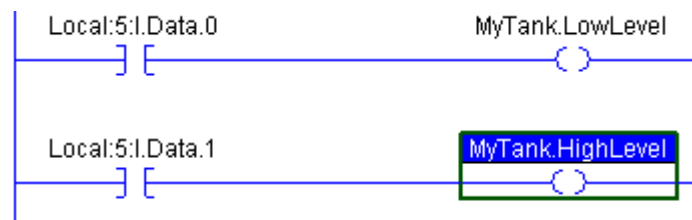
	Name	Data Type	Style	Description
	HighLevel	bool	Decimal	
	LowLevel	bool	Decimal	
*				

Now that “TankAlarm” is an available data type, we can now create an instance of this data type in the tag database.

In this example, I created a tag called MyTank, and gave it the TankAlarm data type. MyTank inherited the members of the TankAlarm data type. We can now have XIC's and OTE's in logic that use MyTank.LowLevel, and MyTank.Highlevel variables.

[-] MyTank			TankAlarm
[-] MyTank.HighLevel			BOOL
[-] MyTank.LowLevel			BOOL

Here is an example of what logic might look like to populate these bits:



## Nesting UDT's

Additional information may be needed for MyTank. Status information may also be available. The TankStatus data type will consist of three members: Level, Draining, and Filling. Let's go ahead and set up this data type, then we will nest TankAlarm, and TankStatus members into a tank data type. This will provide us with all the information we need about MyTank all in one area. Lets do this one step at a time.

First, Create a UDT for TankStatus as shown (Right click the User-Defined folder and select 'new data type'):

The screenshot shows the 'New Data Type' dialog box. The 'Name' field is 'TankStatus' and the 'Description' field is 'Status information for tanks'. Below these fields is a table for defining the members of the UDT.

Members:				Data Type Size:
	Name	Data Type	Style	Description
	Level	DINT	Decimal	
	Filling	BOOL	Decimal	
	Draining	BOOL	Decimal	
*				

Notice that Level is an actual value, so it will have to be DINT. Filling and Draining are either true or false, so their type is BOOL. Apply your changes.

We can use the TankStatus Data type in the tag database as it is, but we would have to create a separate tagname, which is not good. Remember our goal is to organize data.

To have all the data under one tagname, we are going to have to create another data structure that has the members TankStatus and TankAlarm. Each of those data structures have their own members.

Create the data structure as shown:

Name:

Description:

Members:

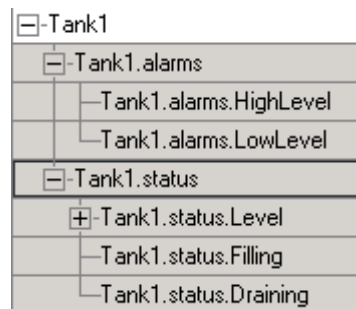
	Name	Data Type	Style	Description
	alarms	TankAlarm		
	status	TankStatus		

Now go to the Controller Tag Database, and create 3 tags, Tank1, Tank2, and Tank3 each having the data type of 'Tank'. Look what happens!

- Tank1			Tank
+ Tank1.alarms			TankAlarm
+ Tank1.status			TankStatus
- Tank2			Tank
+ Tank2.alarms			TankAlarm
+ Tank2.status			TankStatus
- Tank3			Tank
+ Tank3.alarms			TankAlarm
+ Tank3.status			TankStatus

All three Tanks had the 'Tank' data type. (Recall that the tank data type consisted of two members, alarm and status.

Remember also that alarm and status had their own structures as well. Expand the alarm and status tags.



All the data for each Tank is now well organized. These tags should then be incorporated into the input and output data mapping routines.



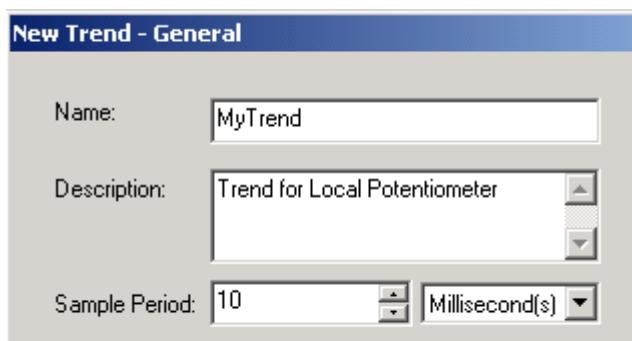
## Trending

Trending is used to graph data over time. This can be either an analog signal or a discrete signal. You are allowed 8 pens per trend chart with a maximum of 255 trending charts per project. This is a simple procedure that will guide you through the creation of a trend chart in your project.

- 1) Right click the **trends** folder in the Controller Organizer Window, and select '*new trend*'.

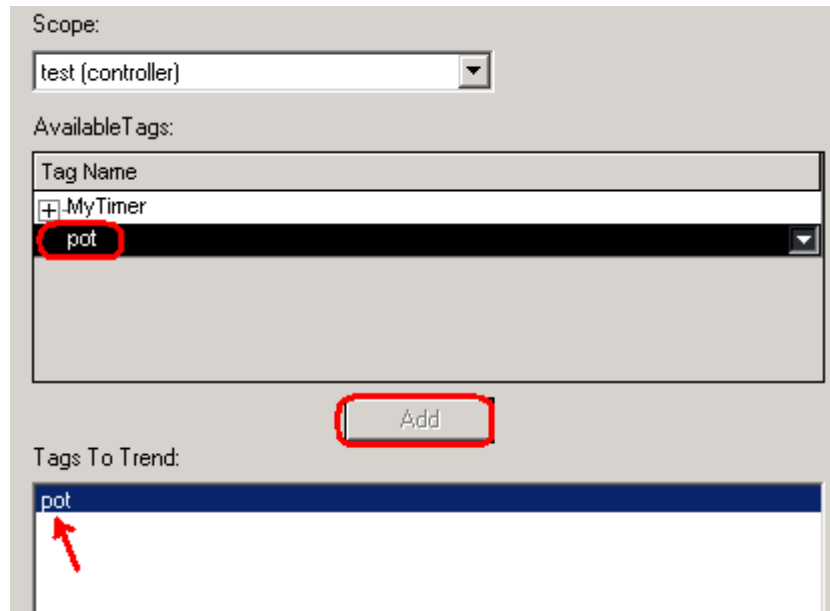


- 2) In the new trend dialog window, name your trend, and add a description. For this example, we will leave the sample period at default. Click '*Next*'.

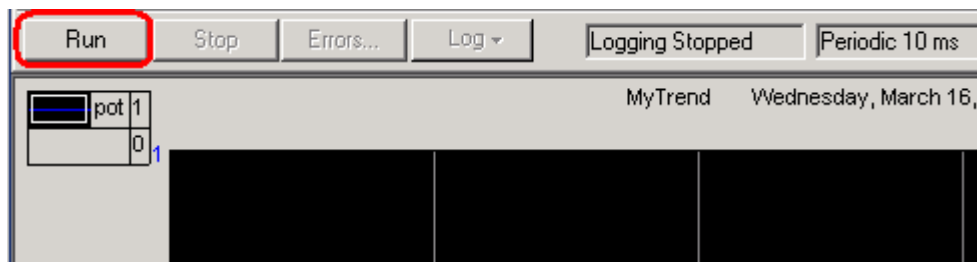




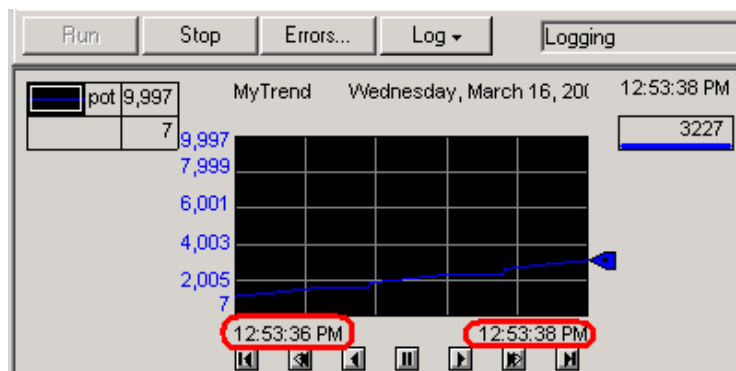
- 3) Choose the tag you wish to trend, then press the '*Add*' button. You will notice your tag is now in the list of tags to trend. Click *Finish*.



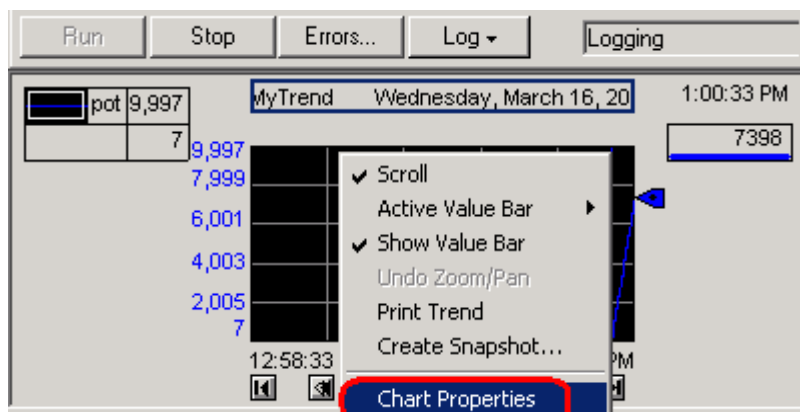
- 4) Next, Click the 'Run' button in the upper left corner of the trending chart.



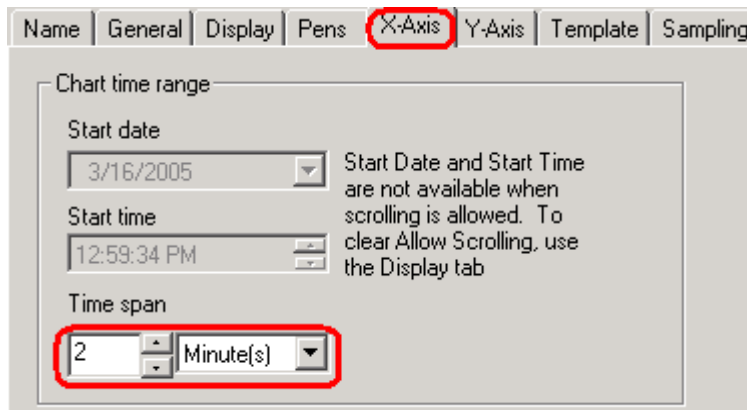
- 5) You will see your chart start tracking. You will notice the chart runs very fast. You only have a time period of 2 seconds currently being displayed from the left to the right side of the screen.



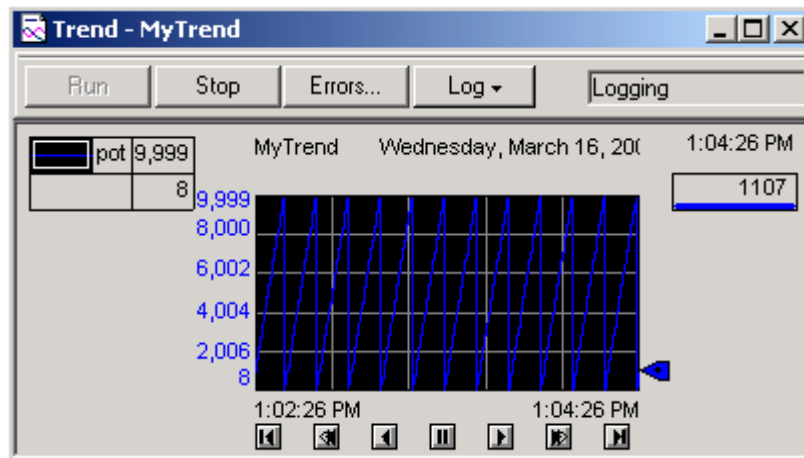
- 6) We are going to change this to 2 minutes so the chart will run much slower. This will give us a better indication of what the signal is doing over time. The X axis runs left to right (This is Time), and the Scale runs from bottom to top. To reconfigure the X axis, right click on the chart, and choose '**Chart Properties**'.



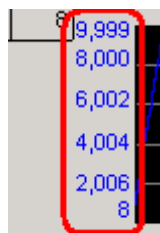
7) On the X tab, change the time span to **2 minutes**. Then press *Apply*, and *OK*.



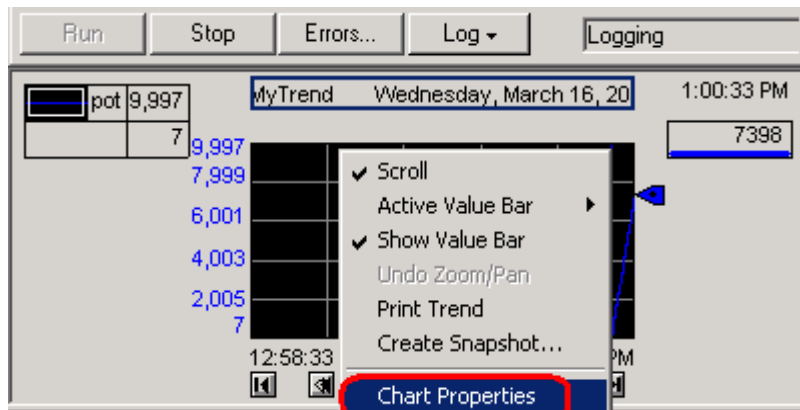
8) You will see your chart is now tracking over a 2 minute time period.



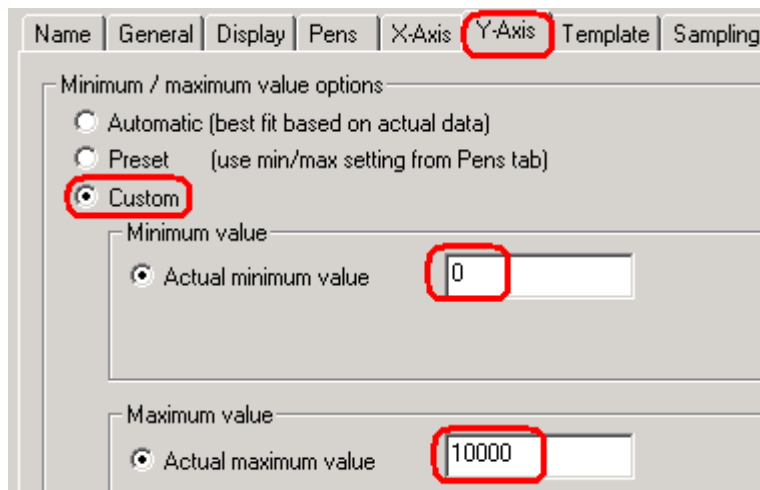
9) You will see the scale is not reflecting the full range possible for the analog signal. This is because the chart defaults to automatic mode. It will take the minimum and maximum value and adjust the scale accordingly. We will account for this in the next step.



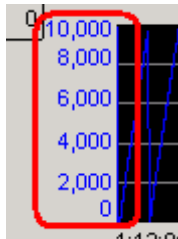
10) Right click on the chart, and go back to '*Chart Properties*'



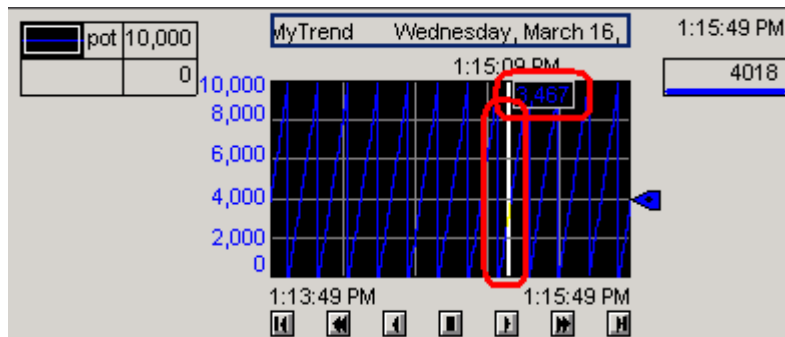
11) On the 'Y Axis' tab, you will notice 3 options. Automatic Mode is what we were running by default. This adjust the scale based on actual data. You can also choose preset if you like, and enter a different min/max value for each pen on the pens tab. The last option is to choose 'Custom'. This will let us lock in our own values for this particular trend chart. We will use custom for this example, and enter 0 as the minimum, and 10000 as the maximum. These values can be adjusted based on the data you are getting from your analog source. Press **Apply**, then **OK** when finished.



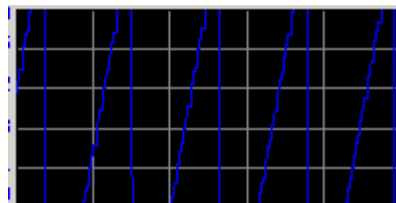
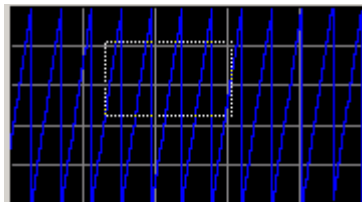
12) You will see the scale on the chart is now locked in with the custom values you entered from the 'Y Axis' tab.



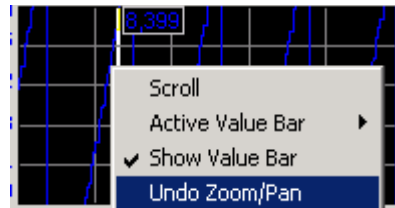
13) By clicking anywhere on the chart, a value bar will appear. This value bar will indicate the exact value of the tag at that moment in time.



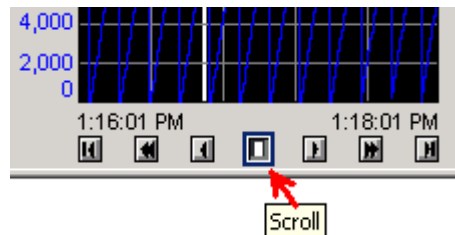
14) If you click on an area of the chart, and drag your mouse, you will draw a box around a certain area of the graph. When you release your mouse, you will be zoomed in on that particular area of the chart. You will know you are in zoom mode by a magnifying glass on your mouse cursor.



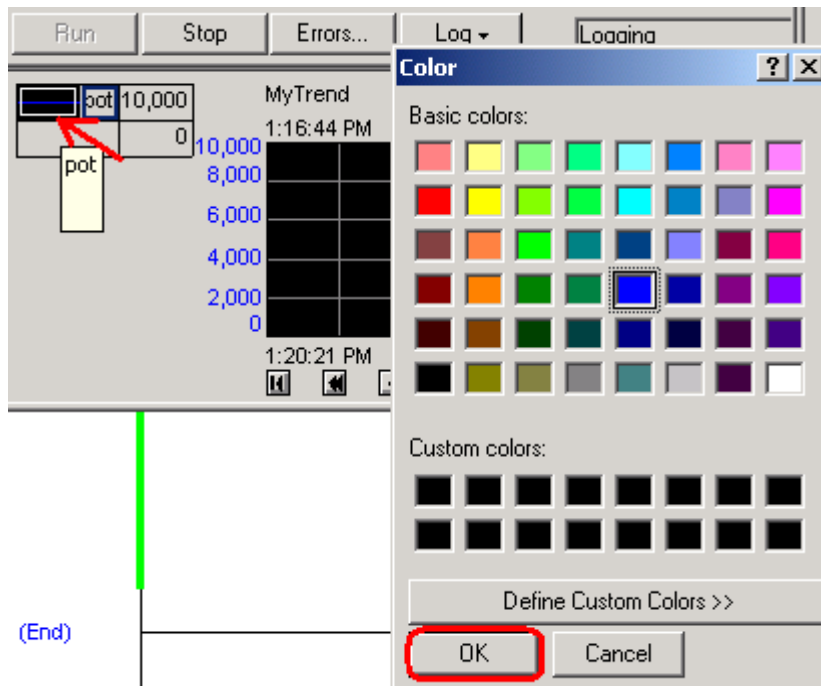
15) To get out of Zoom mode, right click on the chart, and Undo Zoom/Pan.



16) Next, you must restart your chart by un-pausing the graph.



17) Your chart should now be tracking. If you wish to change the color of your pen, Double-Click the pen in the upper left hand corner of your chart. A pallet will appear. Choose your new color from the pallet, then press **OK**.



- 18) You know the basic method for setting up a trending chart. At this time, take a few minutes to explore the other options you have for the trending chart. You can add more pens if you like under the 'pens' tab. This will allow you to track more addresses. You can also change the width of each pen, make them visible, or invisible, etc.... Try to change the color of the background.... If you have questions, ask the instructor.

The screenshot shows a software window with several tabs: Name, General, Display, Pens, X-Axis, Y-Axis, Template, Sampling, Start Trigger, and Stop Trigger. The 'Pens' tab is active.

**Pen Attributes**

	Tag\Expr.	Color	Visible	Width	Type	Style	Marker	Min
1	pot	Green	On	1	Analog	-----	None	0.000000

Below the table is a large grey rectangular area, likely a preview or workspace, with a horizontal scrollbar at the bottom.

Buttons: Add/Configure Tags, Delete Pen(s)

**Multiple Pen Edits**

Visible	Width	Type	Style	Marker	Min	Max	Eng. Units

Buttons: Clear Selections, Apply to Selected Pen(s)

Bottom buttons: OK, Cancel, Apply, Help





## **Hands On Troubleshooting**

Your instructor will guide you through the following troubleshooting utilities:

Search/Find

Cross Reference

Trending

Extensive time will be provided for you to practice tracing down outputs, cross referencing through a plant program, and learning which inputs are required to energize various outputs.

