Allen Bradley PLC-5

Level 2 – Advanced Concepts

EthernetSupport.com

"Customized Automation Training"

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

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____________

PLC/SLC Maintenance and Troubleshooting
PreTest 20 Minutes

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 RSLinx software?

9) What is the purpose of RSLogix software?

10)What is the purpose of Data Tables?

11)What is the purpose of Program Files?

12)Define a BIT of memory:

13)Define a WORD of memory:

14)If Ladder 2 is the main routine, and the only main routine, What must be placed in ladder 2 to instruct the processor to execute other routines (subroutines)?

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 PLC 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 PLC 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 jump to a routine that does not exist. What will happen the instant your edits are tested?

19)You suspect an intermittent problem with a limit switch on a safety gate. The intermittent problem is periodically shutting a system down. What feature of RSLogix software might you use to confirm or rule out what you suspect?

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.

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.

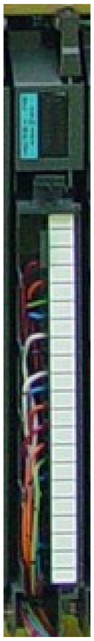
Please answer the following questions:

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) In what numbering system are your inputs labeled in?



Discrete Output Modules

The purpose of the discrete output module is to control field devices. The discrete input module requires power from an external source. When a 1 is placed into the output table of the PLC (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 numbering system are the outputs labeled in?

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 the PLC-5, you must utilize block transfers to gather information from Analog modules. For our class, the SLC chassis will run on remote I/O.

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?

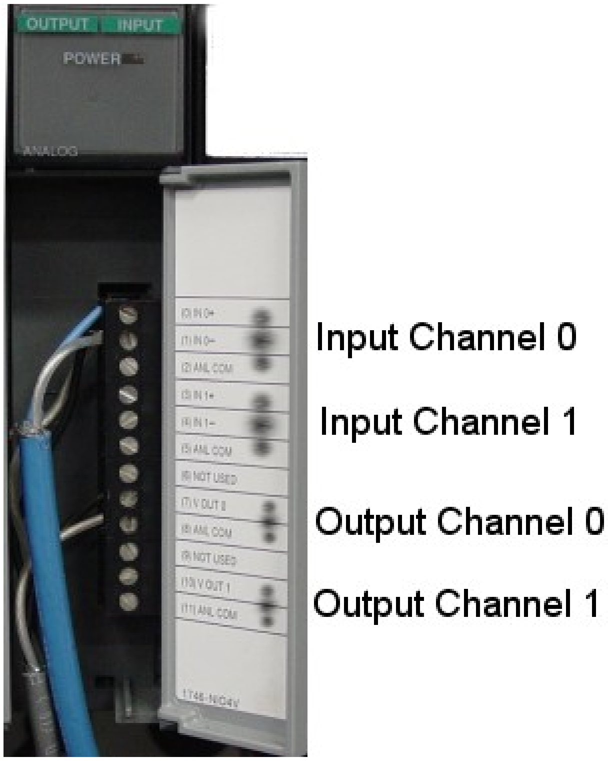
3) How do you set up the input channels to accept either a current or a voltage input?

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:

An Analog Module – 1747-NIO4V



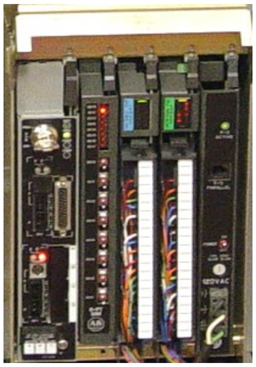
The Chassis

The chassis is the device which holds modules. Allen Bradley makes the SLC chassis available in 4, 7, 10, or 13 slots, and the PLC-5 chassis in 4, 8, 12, or 16 slots. For the SLC, the processor is included in the slot count. For the PLC-5 chassis, the processor is not included in the slot count.

Both the SLC and the PLC support extended local I/O. This is more common in the SLC, however. For the SLC system, a maximum of 3 chassis can be connected together using extended local I/O, but not to exceed the valid slot count of 0 to 31. The 1746-C7 cable can be used to connect two chassis together which are mounted horizontally (side by side), and the 1746-C9 cable is longer for chassis that are mounted vertically (one above the other).

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.

The power supply has a fuse which protects the AC side of the power supply. If this fuse blows, the power supply is probably defective and in need of repair.

Please answer the following questions:

1) Where does power come from to power field devices such as solenoids, lights, and motor starters?

2) What does a blown fuse indicate on a power supply?

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 SLC PLC-5 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 it's travel, it might close a limit switch. The processor will see that a limit switch has been closed, and shut off the solenoid.

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 status bit is also set in the memory of the processor (S:10/0 for the PLC). Once you have an indication of a low battery, change the battery within one to two days. Change the battery while the processor is powered up.

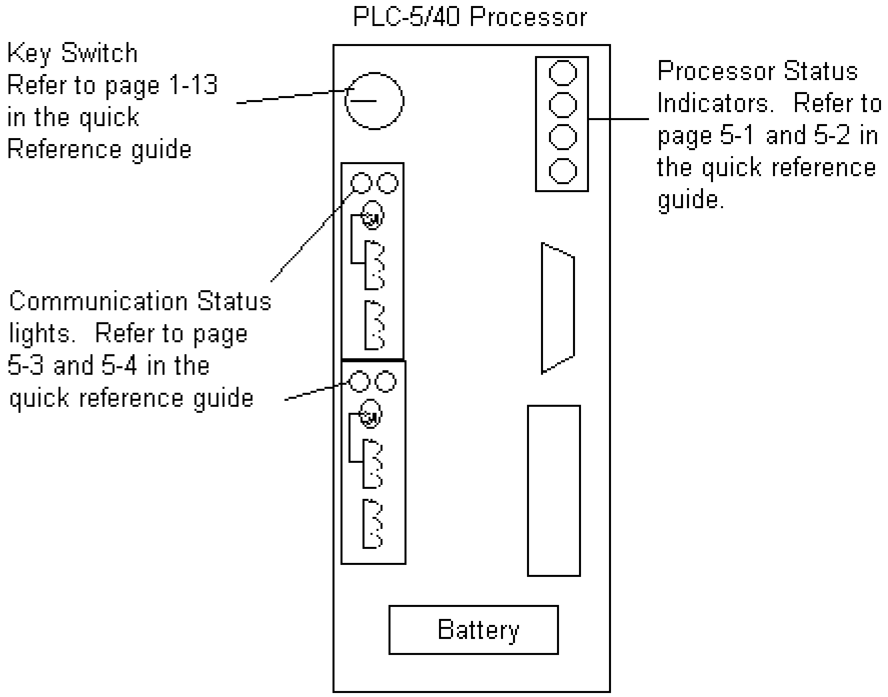
2) Next to the battery, you will find a socket for an EEPROM. The EEPROM is an on board backup of the processor's memory, and must be purchased separately. Changes to the PLC program do not automatically go to the EEPROM chip. You must manually store changes to the EEPROM from the COMMS menu in RSLogix.

3) On the front of the processor, you will find several status lights:
   1. RUN – Indicates when when processor is in RUN Mode
   2. PROC – 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 S2 status file on the ERRORS tab. If the fault light is solid red, this usually indicates the processor lost it's program, or has a hardware problem. You can try the following: reseat the processor, replace eeprom, clear memory and reload program, or replace processor.
   3. BATT-- Indicates the battery is low or missing
   4. FORCE – If flashing indicates forces are installed but not enabled... If solid indicates forces are installed, and enabled in the processor.

Communication lights-- You may also have some indicators to indicate when communications are active depending on the type of processor you have.

For more a more detailed explanation of the PLC status lights, look at page 5-1 and 5-2 in the PLC-5 quick reference guide. The most common communication status lights are found on page 5-3 and 5-4.

The PLC-5 Processor components:



Addressing Procedure

There are three things you must know about every chassis before it can be addressed properly... The Rack # the chassis is set up for, the starting module group (SMG) of the chassis, and the addressing mode. For remote chassis, you will need to also determine the baud rate before setting the chassis up in the I/O Configuration.

Local Chassis:

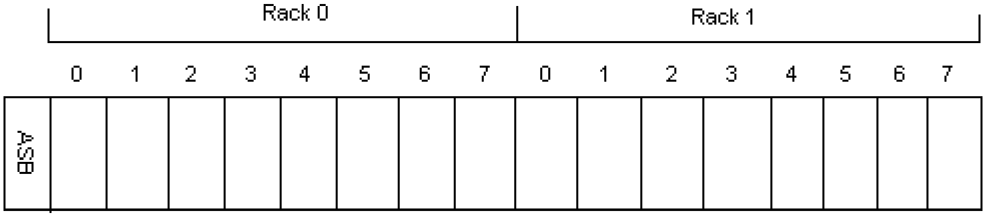
The local chassis is the chassis where the processor resides that you are going to be on line with.

1) The Rack # will almost always be 0, except in very special cases for PLC-2 compatibility.
2) The SMG will be 0 also.
3) The addressing mode can be derived using one of the following methods:
   1. Look at the backplane dip switches, comparing them to page 4-1 in the Quick Reference guide.
   2. If you know the Rack size (ie: ¼ ½ ¾ or full), and the number of slots in the chassis, refer to page 2-8 of the PLC-5 Quick Reference Guide to derive the addressing mode.

Numbering the Groups

4) If the addressing mode is 1 slot, write a zero above the first slot, and number every slot consecutively.
5) If the addressing mode is 2 slot, draw a bracket around every two slots on the chassis. Write a 0 on the first bracket, and then number each BRACKET consecutively.
6) If the addressing mode is ½ slot, then write a 0 and 1 on the first slot, and assign two module groups to every consecutive slot.

Note: ***Remember the only possible module groups are 0 to 7. Do not number a module past 7. If you get to module group 7, and still have modules left in the chassis, increment the rack number (octally), and restart the module groups at 0. See the example below for 1 slot addressing:***



Remote Chassis

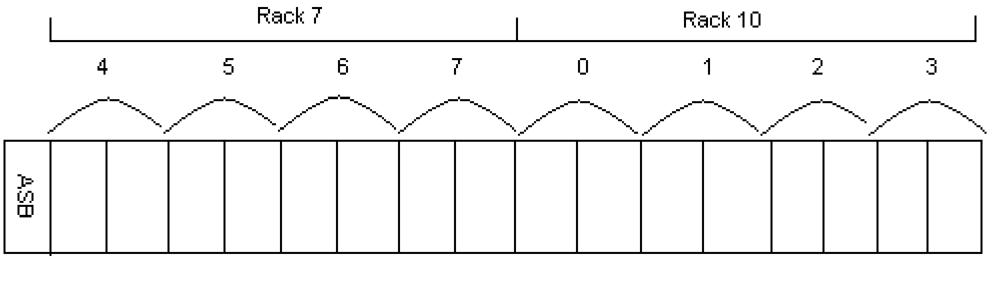
There are three things you must know about a remote chassis for proper addressing in addition to the baud rate.

1) Rack #. For a remote PLC-5 chassis, pull out the adapter, and use page 4-4 to 4-6 to derive the rack # if the chassis is not labeled. You will look at switches 1 to 6 of switch bank 1.
2) Starting Module Group (SMG): Pull out the adapter and look at switches 7 and 8 of switch bank 1. Compare these switches to page 4-6.
3) Baud rate: You may want to write down the baud rate for later use. This can be found on switches 1 and 2 of switch bank 2. Compare these switches to page 4-6 in the quick reference guide.
4) Addressing Mode: There are two ways to determine the addressing mode:
    1. Pull out the adapter, and compare the switches on the backplane to page 4-2 in the quick reference guide.
    2. If you know the Rack Size, and how many slots are in the chassis, look at page 2-8 in the quick reference guide.

Numbering the Groups

5) If the addressing mode is 1 slot, write the SMG above the first slot, and number every slot consecutively.
6) If the addressing mode is 2 slot, draw a bracket around every two slots on the chassis. Write the SMG on the first bracket, and then number each BRACKET consecutively.
7) If the addressing mode is ½ slot, then write the SMG and the next module group on the first slot, and assign two module groups to every consecutive slot.

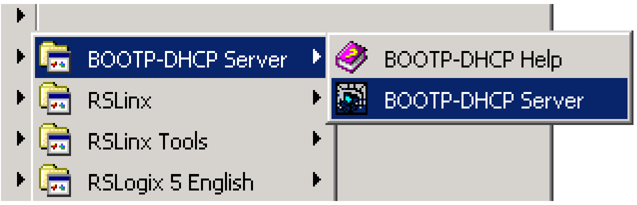
Note: ***Remember the only possible module groups are 0 to 7. Do not number a module past 7. If you get to module group 7, and still have modules left in the chassis, increment the rack number (octally), and restart the module groups at 0. Below is an example of 2 slot addressing, crossing a rack border. Notice the next rack # after 7 is 10 in an octal environment.***



RSLinx -- Utilizing the BootP/DHCP Server

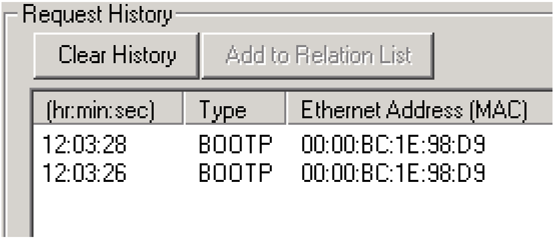
1) Write down the Ethernet Hardware Address of the device you wish to configure. This is also called the MAC address. Here is an example of a hardware address:
   00:00:BC:1E:98:D9

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.



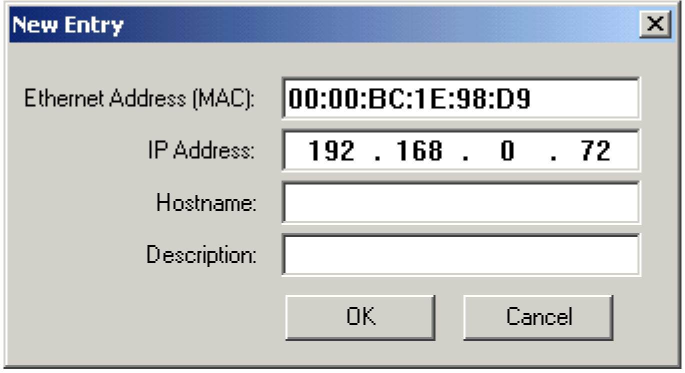
3) Once the server is open, it may as for some specific network information if this is the first time the BootP server has been run. You can obtain this information from your network administrator. For this example, we just set the subnet mask to 255.255.255.0 then click OK.

4) Power up your processor, and you should see the Ethernet device begin to request and address.



(hr:min:sec)	Type	Ethernet Address (MAC)
12:03:28	BOOTP	00:00:BC:1E:98:D9
12:03:26	BOOTP	00:00:BC:1E:98:D9

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.



Ethernet Address (MAC): 00:00:BC:1E:98:D9

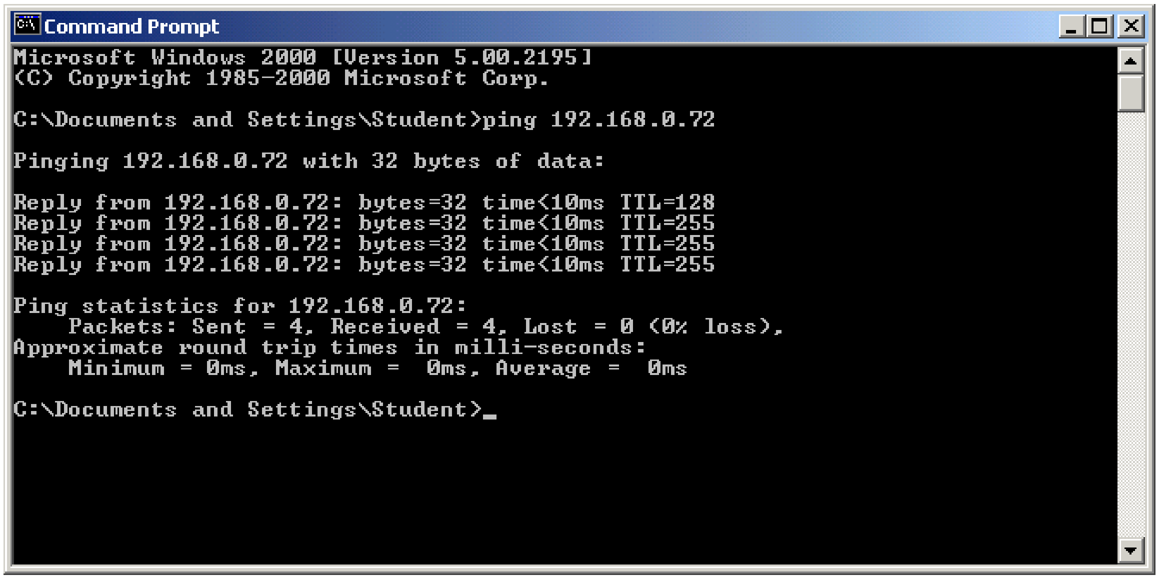
IP Address: 192 . 168 . 0 . 72

Hostname:

Description:

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 2000 [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>_
```

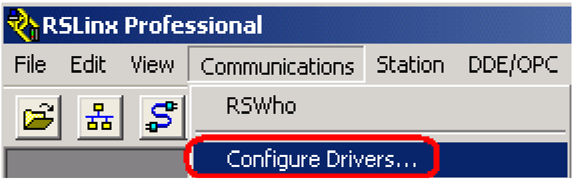
You should get replies. To ping continuously, use the -t flag after the ping command. A control-C will stop the ping command.

Configuring the DF1 Driver in RSLinx

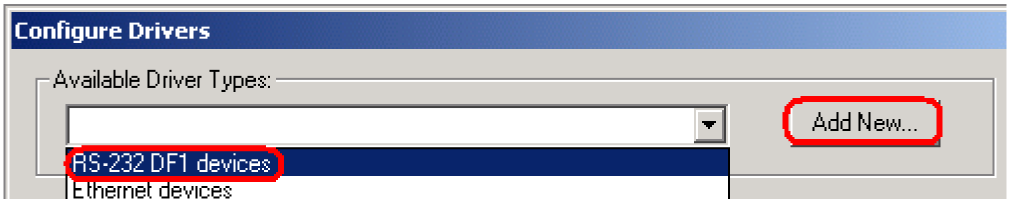
The DF1 Driver is used for point to point communication over RS232 between a COM port on a PC, and the serial port (Channel 0) of a processor. The following steps will take you through a sample configuration of the DF1 RS232 driver.

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

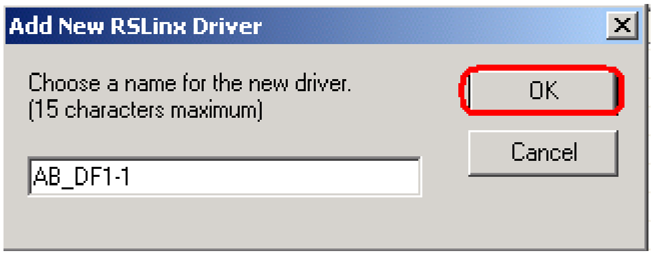
2) Click 'Communication' on the menu bar, then choose 'Configure Drivers'.



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



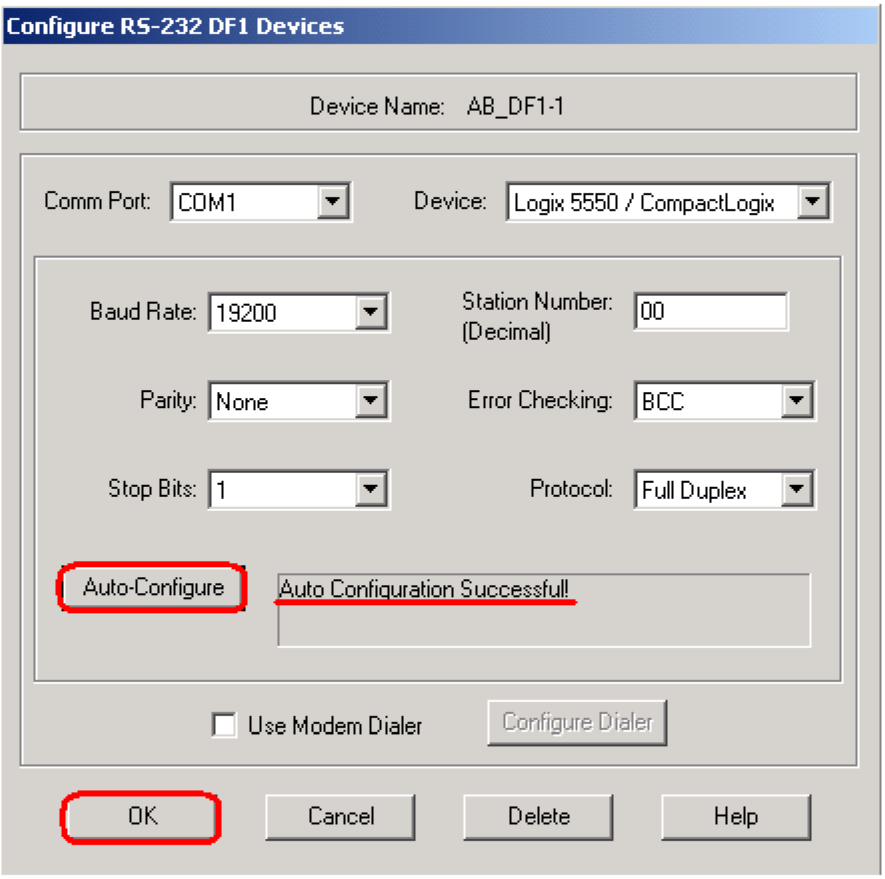
4) For this example, the name can be left at default. Press OK.



Choose a name for the new driver.
(15 characters maximum)

AB_DF1-1

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: (Decimal) 00

Parity: None

Error Checking: BCC

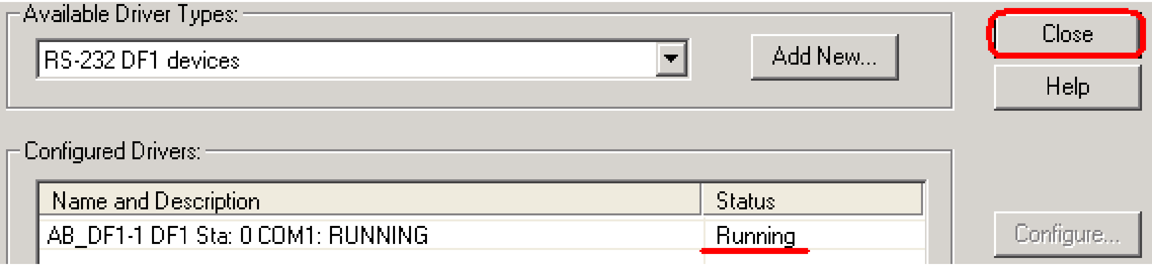
Stop Bits: 1

Protocol: Full Duplex

Auto Configuration Successful!

☐ Use Modem Dialer

6) You will see the driver is now running. Close the “Configure Drivers' screen.

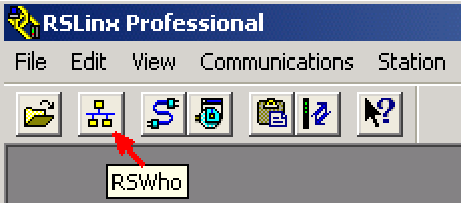

Available Driver Types:

RS-232 DF1 devices

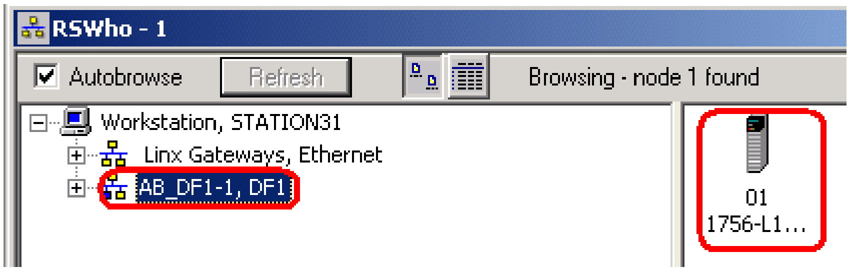
Configured Drivers:

Name and Description	Status
AB_DF1-1 DF1 Sta: 0 COM1: RUNNING	Running

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.



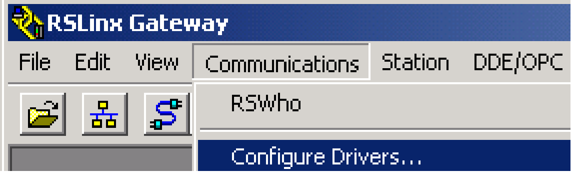
To go on line, you must go to RSLogix at this point.

Configuring the Ethernet Driver in RSLinx

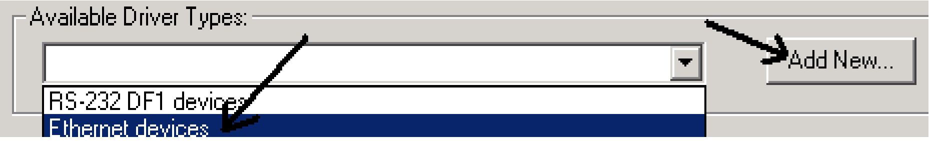
The Ethernet driver is used to make a connection to Ethernet Devices, such as an Ethernet PLC-5, or a ControlLogix system. The following steps will walk you through a sample configuration of the Ethernet driver in RSLinx.

1) Open RSLinx communication server

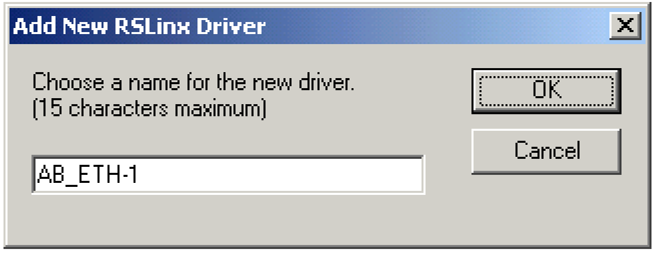
2) Click 'Communication' on the menu bar, and then choose 'Configure Drivers'.



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



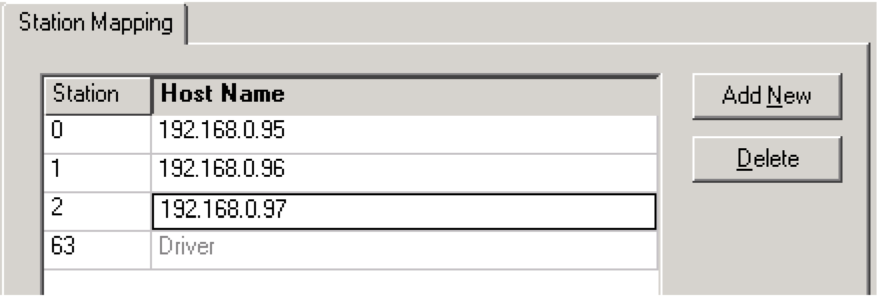
4) For this example, the name can be left at default. Press OK.



Choose a name for the new driver.
(15 characters maximum)

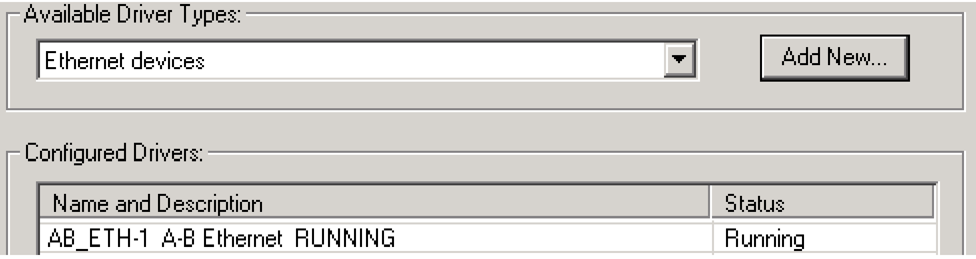
AB_ETH-1

5) Populate the list of hostnames. If you do not have a way to resolve host names, 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 off line project, or in some cases, the IP address is displayed on the front of the module.



Station	Host Name
0	192.168.0.95
1	192.168.0.96
2	192.168.0.97
63	Driver

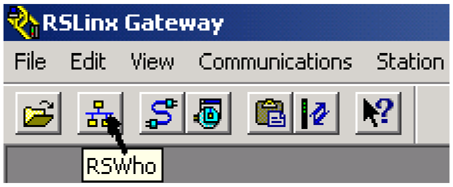
6) Press Apply, then OK. You will see the driver is now running. Close the 'Configure Drivers' screen.



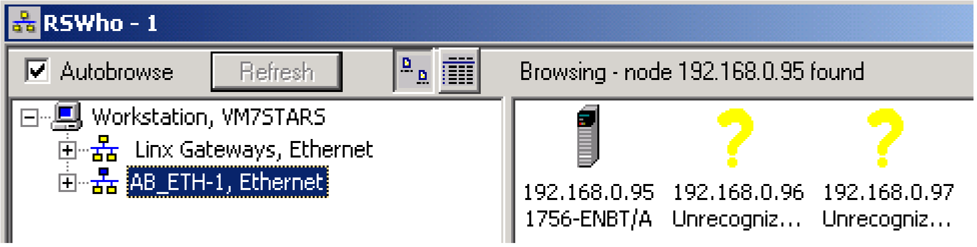
Available Driver Types: Ethernet devices

Name and Description	Status
AB_ETH-1 A-B Ethernet RUNNING	Running

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.



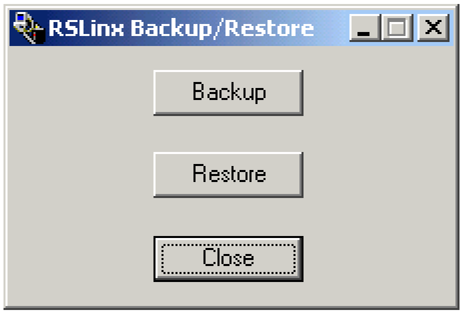
9) To go on line with a PLC, you must go to RSLogix at this point.

RSLinx Backup Restore Utility

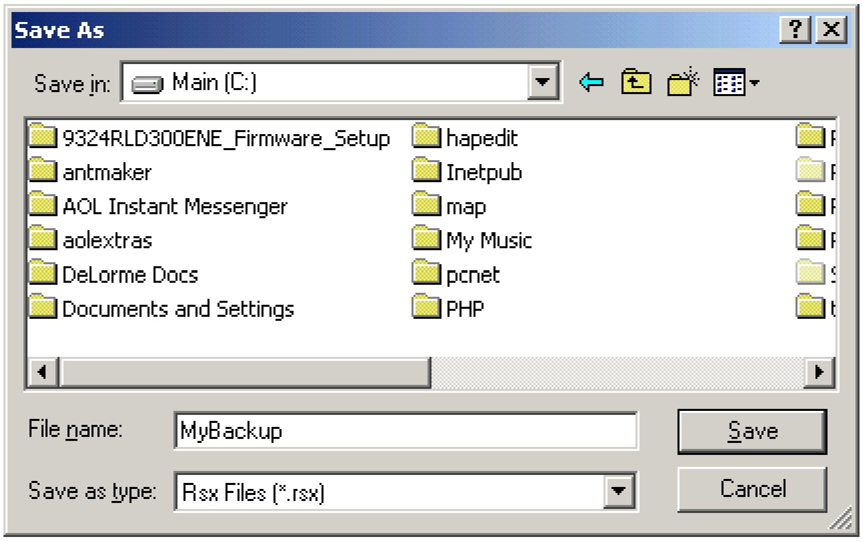
The RSLinx Backup Restore Utility can be used to backup the current driver configuration of RSLinx, or restore the configuration from a previous backup at an earlier time.

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.



Save in: Main (C:)

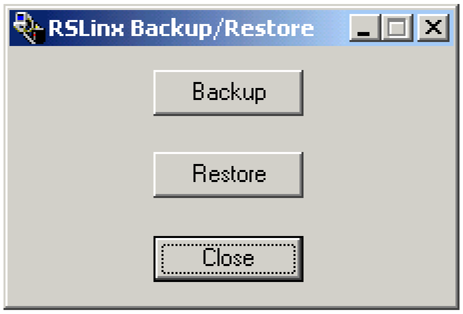
File name: MyBackup

Save as type: Rsx Files (*.rsx)

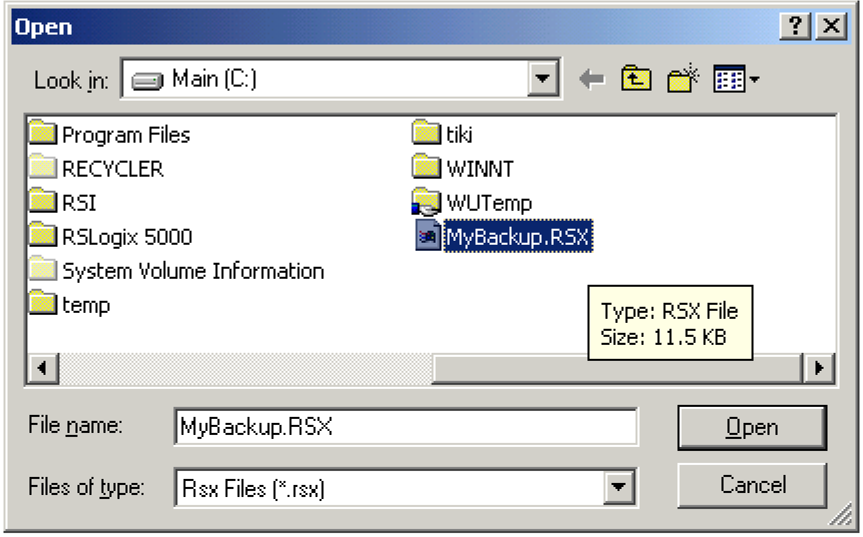
3) You will then get a message that the operation completed successfully. Press OK, then you can close the RSLinx Backup Restore Utility.

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, the press 'Open'.



Look in: Main (C:)

Type: RSX File
Size: 11.5 KB

File name: MyBackup.RSX

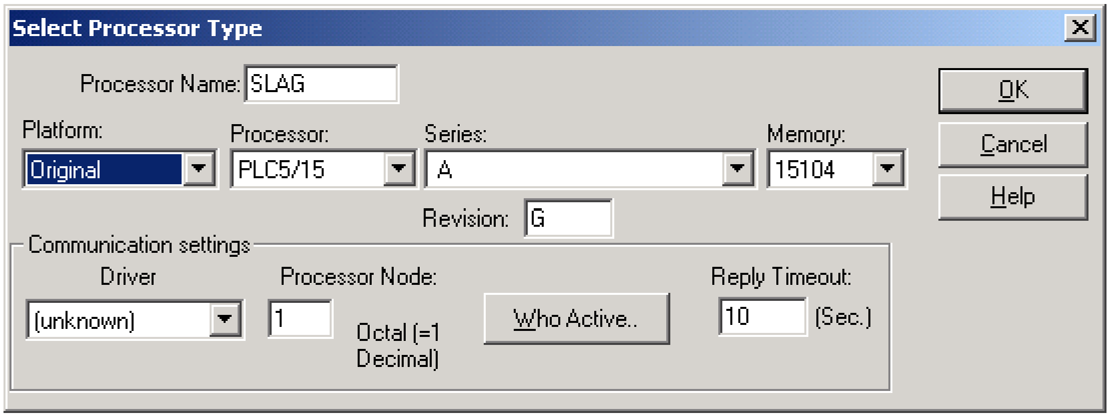
Files of type: Rsx Files (*.rsx)

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.

Creating a new project in RSLogix 5

1) Open RSLogix 5 – You may have a short cut on the desktop, or under Start | Programs | Rockwell Software | RSLogix 5 English | RSLogix 5 English.

2) Next, you will select the processor type.
   a. Enter your processor name. This name will describe the process you are developing the project for.
   b. Select the Platform of your processor. The older processors that do not have a 25 pin serial port (just 9 pin for DH+ are on the original platform). Other options are Enhanced (processors that have a 25 pin serial port and DH+/RIO ports), Protected (Not used quite as often), Ethernet (with an AUI port), or ControlNet (BNC connectors).
   c. The series, memory, and revision are written on the side of the processor.
   d. If you know the RSLinx driver you will be using, and the node of your processor, enter it manually. Otherwise, press the Who Active button, and choose your PLC from the RSWho screen. The driver and node would then be automatically filled in.
   e. For this example leave the Reply Timeout at 10 seconds. If you have questions on any dialog screen, use the help button.
   f. Press OK.



Select Processor Type

Processor Name: SLAG

Platform: Original

Processor: PLC5/15

Series: A

Memory: 15104

Revision: G

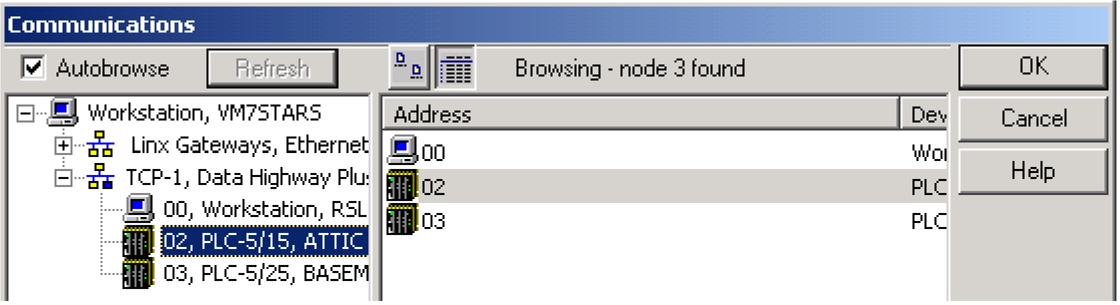
Communication settings

Driver: (unknown)

Processor Node: 1 Octal (=1 Decimal)

Reply Timeout: 10 (Sec.)

Example of Who Active Screen:



Communications

☑ Autobrowse

Browsing - node 3 found

- Workstation, VM7STARS
  - Linx Gateways, Ethernet
  - TCP-1, Data Highway Plu:
    - 00, Workstation, RSL
    - 02, PLC-5/15, ATTIC
    - 03, PLC-5/25, BASEM

Address	Dev
00	Wo
02	PLC
03	PLC

Adding Ladder Logic Off line

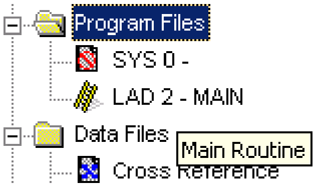
Ladder logic is still the primary programming language for Allen Bradley PLC's. Ladder logic was designed to simulate relay logic when PLC's were first introduced into the industry. With the wide use of Object Oriented concepts in today's programming languages, and the limitations of ladder logic for programming, one day ladder logic may be a language of the past.

Ladder 2 is the only main routine in the SLC, and by default Ladder 2 is the only Main Routine in the PLC-5. Other ladders can be added called Subroutines. As with many other programming languages, if you are going to have subroutines, there has to be some means of calling those subroutines so they will be executed by the processor. In the PLC-5 and SLC 500, this is the JSR instruction (jump to subroutine). The maximum routine number is 255 for the SLC, and 999 for the PLC-5.

Usually the JSR instruction is placed in the main routine, but JSR instructions can also be placed in subroutines if the subroutine itself is being called from a JSR. For example... The main routine might jump to ladder 3, and ladder 3 could jump to ladder 4. This is called nesting subroutines. Each processor has limitations on how deep the programmer can nest routines. Consult the help file for your processor.

For this example, Ladder 2 will be the main routine. We will also add other routines: Ladder 3, Pumps; Ladder 4, Timers; Ladder 5, Counters; Ladder 6, Data; Ladder 7; Analog.

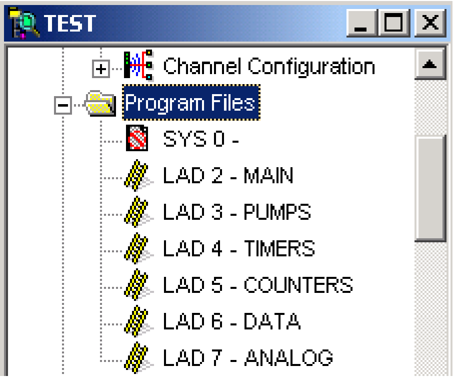
1) Rename Ladder 2 to Main. In the project tree, right click Ladder 2, and select Rename. Type MAIN as the routine name. Press enter to accept. Right click on MAIN, and select properties. Type Main Routine as the description. Now, move your mouse over the main routine. You will see the description appears as a tool tip.



2) Next Right click on the program file folder and select 'new' to add other routines to your project. Set up your routines as follows:

Number	Name	Descriptions
3	Pumps	Logic for All Pumps
4	Timers	Project Timers
5	Counters	Project Counters
6	Data	Calculations
7	Analog	Block Transfers for Analog

When finished, your project tree will look as shown below:



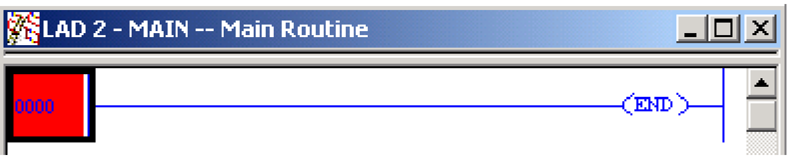
TEST

- Channel Configuration
- Program Files
  - SYS 0 -
  - LAD 2 - MAIN
  - LAD 3 - PUMPS
  - LAD 4 - TIMERS
  - LAD 5 - COUNTERS
  - LAD 6 - DATA
  - LAD 7 - ANALOG

3) Now double click on main to ensure you are looking at the main routine in the ladder window to the right.

4) One method of adding logic is the ASCII method. You can just type the 3 character instruction followed by certain parameters. We will need 5 JSR statements... One for each subroutine. Do not jump to the main routine from the main routine. A processor fault may result.

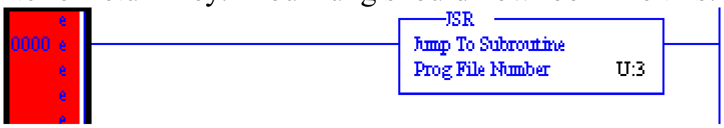
5) Highlight Rung 0 so it is red as shown below:



LAD 2 - MAIN -- Main Routine

6) Type the following text: jsr 3

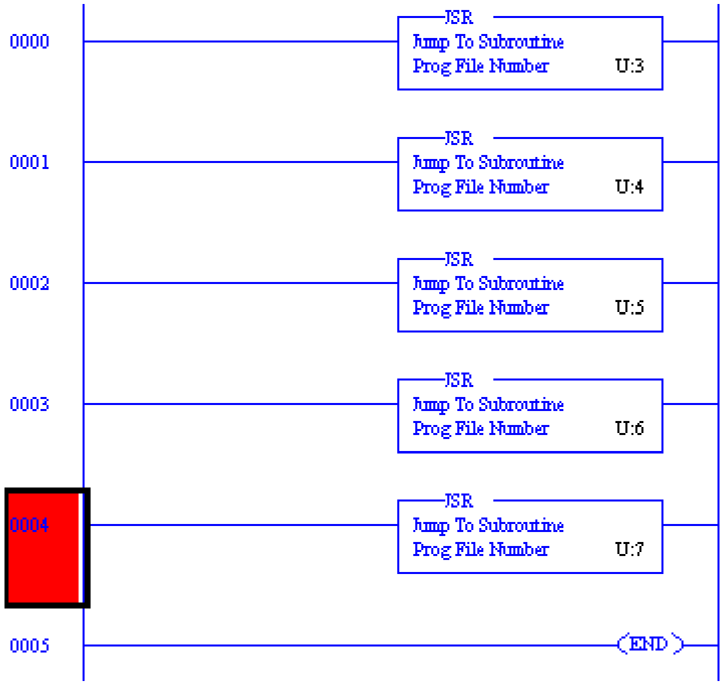
Now press the enter or return key. Your rung should now look like this:



7) Now, click on the end rung again, and type the next jump statement: "jsr 4", then press enter. Go ahead and all the rest of the jump statements (jsr 5, jsr 6, and jsr 7) Be sure to click on the end rung each time or you will over write your work.

8) Notice the e's in the margin. This indicates that we have edits that have not yet been verified by RSLogix. Verifying edits will make sure the PLC can understand the instructions we are giving it. Verifying ***only tests to see if the processor can understand what we are telling it to do. It will not ensure that we are telling it to do the right thing!*** Move your mouse over these two icons on the standard tool bar:

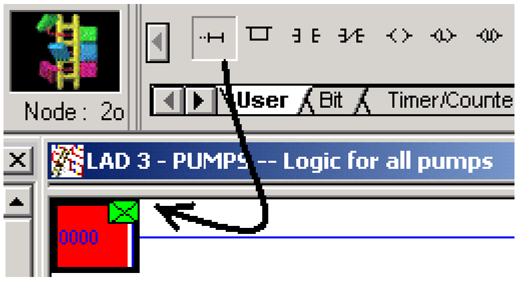
The first icon will verify edits too the file we are currently in (ladder 2). The second icon will check all ladders. Click the first icon, and you will notice the e's no longer appear in the margin. Your ladder should now look as shown below:



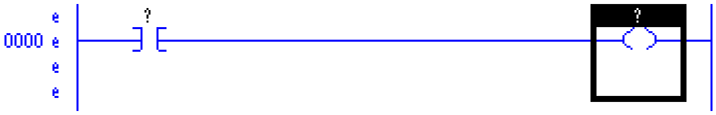
Note: Advanced users of the JSR instruction may use extra parameters to pass information to and from subroutines. For our examples, we will keep our logic simple.

9) Next, double click Ladder 3. This is where our pump logic belongs. Look in the title bar of the ladder view to ensure you are entering logic into ladder 3. You will also notice at the bottom of the ladder view a new tab is created each time a new routine is opened. This allows you to quickly switch between routines you are interested in.

10)The first method you learned when adding logic was the ASCII method. For the next few rungs, let's look at the Drag and Drop method. You cannot add any logic to the end rung, so you will need to drag a new rung into your ladder view. The first icon on the instruction tool bar is 'new rung'. Click on the new rung, and hold the left mouse button down. Drag the new rung down into the ladder view, and you will see red boxes indicating where you can drop the new rung. Move close to a red box, and it will turn green. You can then let go of the left mouse button, and the new rung should be in place.



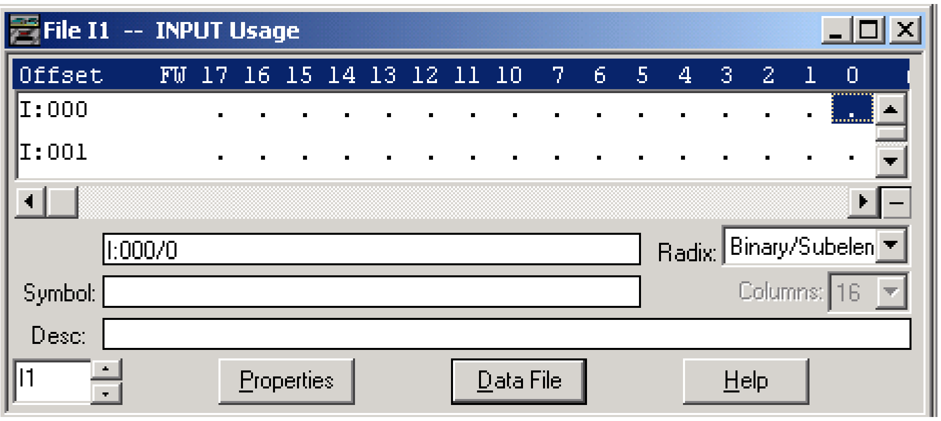
11)Use the same drag and drop method to add an xic and an ote to the new rung. When you are finished your rung should appear as shown below:



Note: An easier method for you might be the single click method. Be sure rung 0 is highlighted so the rung number is red. Click the new rung icon one time, then click the XIC icon, then the OTE icon. You will get the same result.

12)Notice the question marks above the instructions. We have to associate the instructions with addresses. There are several methods you can use for this. One method would be to click on the instruction and start typing the address, or open a data table, and drag the addresses from the data table. For this example, we will use a PLC project, and drag the addresses onto the instructions.

13)Open the input data table. Although we can drag directly from the data table, let's click the usage button at the bottom of the data table, so we can see what addresses are already in use.



File I1 -- INPUT Usage

```
Offset    FW 17 16 15 14 13 12 11 10  7  6  5  4  3  2  1  0
I:000           .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
I:001           .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

I:000/0 Radix: Binary/Subelem

Symbol:

Desc:

Columns: 16

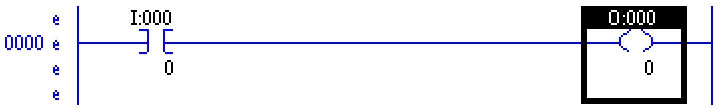
I1

14)Drag an address from the data table into your logic window. When you see a green square on the XIC instruction, drop the address.

15)In the bottom left corner of the data table, click the down arrow to navigate to data table 0 (output table). Drag an address from the output table onto the OTE instruction.

Note: The XIC instruction can read most any bit in any data table, and the OTE instruction can write to most any bit in any data table. We are using the input and output tables just for example.

When finished your logic should look similar to this (with your own addresses):

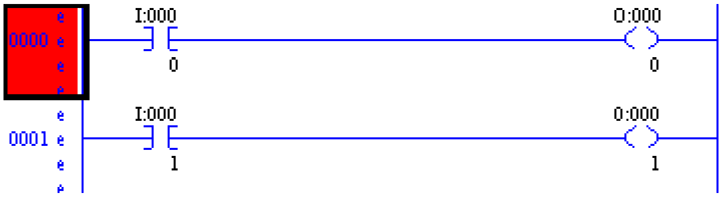


16)Download and test your work.

Finding Errors

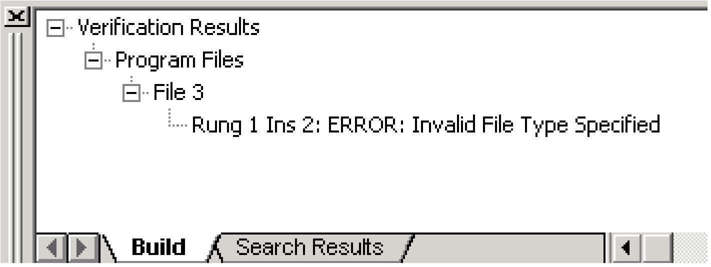
Errors you make are sometimes tricky to find. RSLogix has a results window that will show you where your mistakes are. Mistakes are usually found when verifying a project. If there is an error in your program, a results window will appear at the bottom of the ladder view, informing you where the error is, and a description of the error. By double clicking the error itself in the results window, RSLogix will take you to the exact location of the problem. Sometimes this might be behind the results window, so you may have to close the results window to see it.

Can you spot the error with this logic?

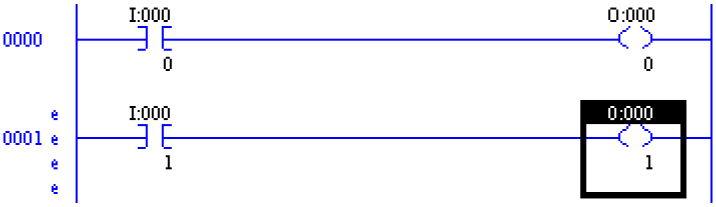


This error is very difficult to find. I will verify the project, and the results window will appear.

I had to click the + on file 3 to see what the error was, but here is the result:

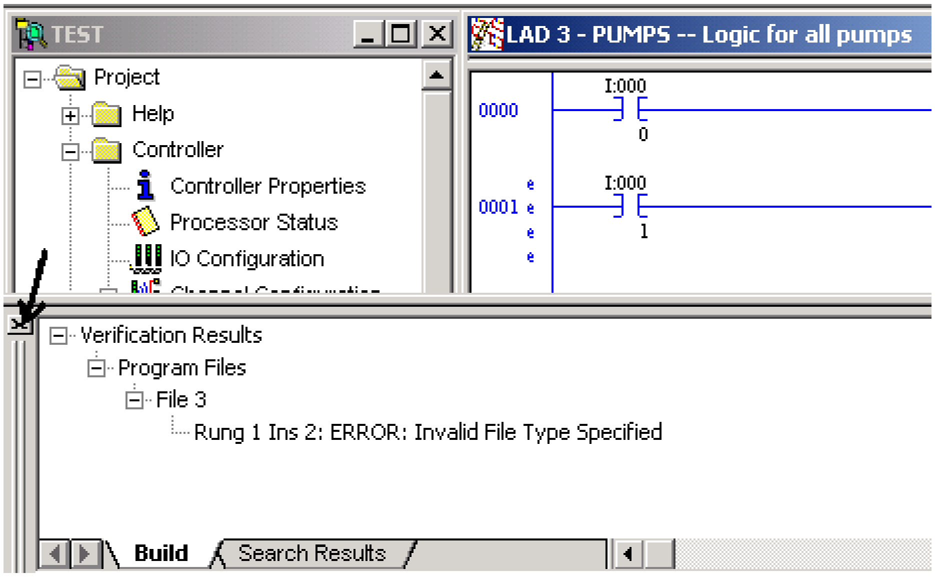

Verification Results
Program Files
File 3
Rung 1 Ins 2: ERROR: Invalid File Type Specified

Still can't see the problem, I will double click the error, and RSLogix will highlight where the problem is:



Do you see the problem yet? The first character is the number 0, and should be the letter O.

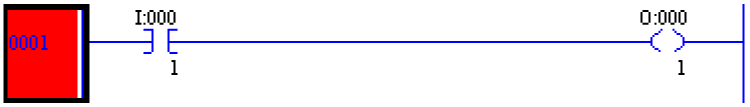
Note: You should exit the results window by clicking the X in the upper left hand corner of the window. If you resize the results window, you may not see it next time you have an error in your logic.



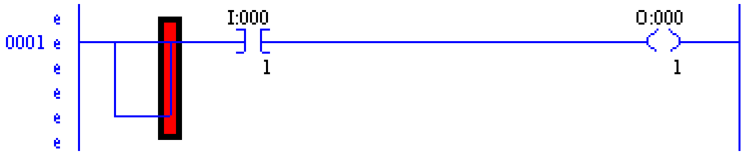
Verification Results
Program Files
File 3
Rung 1 Ins 2: ERROR: Invalid File Type Specified

Branching

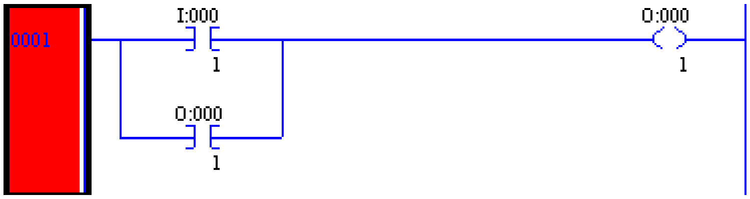
Branching can also be accomplished in ASCII mode using the BST (Branch Start), NXB (Next Branch), and BND (Branch End) commands. For this lesson, we will discuss the drag and drop method. In the example below, Let's seal around the input when the output is energized.



17)The next step is to drag a new branch from the instruction tool bar. You can drop the new branch at any location a red box appears. Be sure to move close to any red box, and when it turns green, your branch can be dropped. For this example, drop off line.

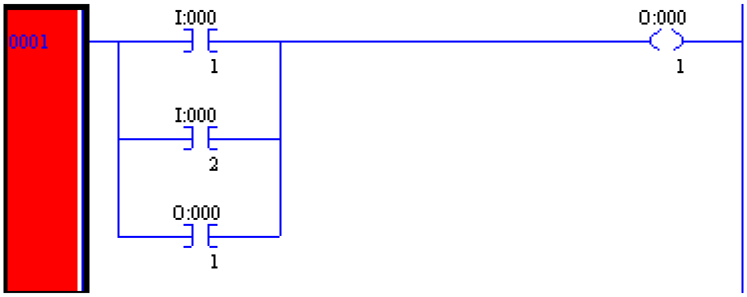


18)Only the right side of the branch can be moved. We have two options: You can move the XIC inside the branch, or you can drag the right branch leg to the other side of the XIC instruction. When finished, the rung will look like this:

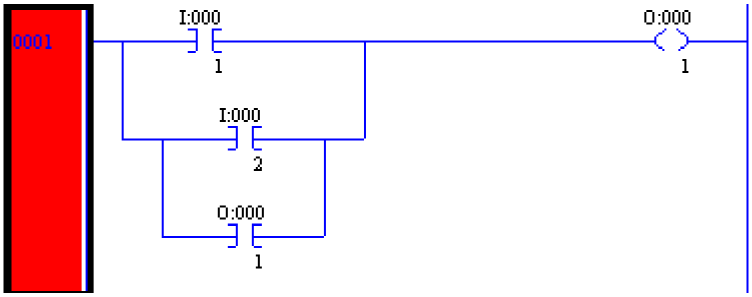


19)If other inputs are required, you can continue to extend the existing branch down. Do not drag new branches down from the instruction tool bar unnecessarily. When possible, extend the existing branch. Nesting branches has limitations, and increases the processor scan time. Look at the examples below:

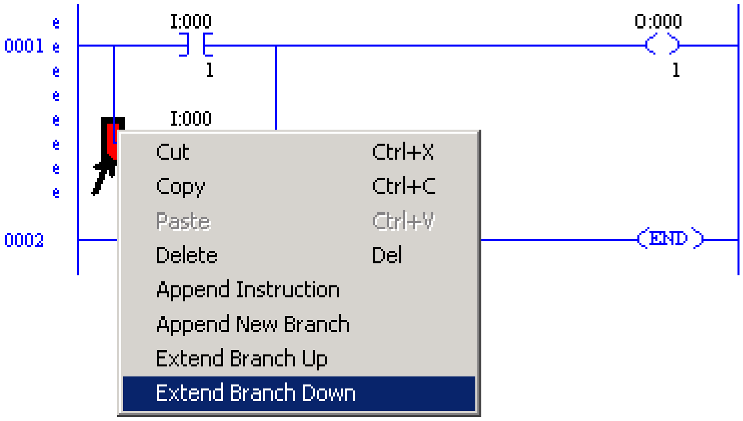
Correct



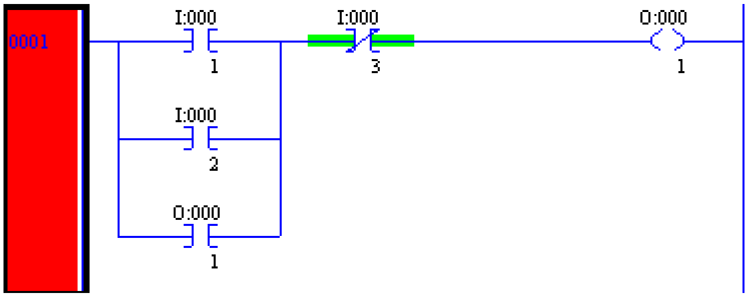
Incorrect



To extend an existing branch, right click on the bottom left corner of the branch leg, and choose 'extend branch (up or down)'.



20)There is only one thing left to do. Once this branch seals in, there is no way to make the output false again without restarting the processor. Here is what our final rung might look like:



On Your Own!

Write logic that will perform the following actions:

1. If input 2 (is on), then (energize) light 2.

2. If input 3 AND 4, then (energize) light 3.

3. If input 3 OR 4 OR 5, then light 4.

4. If input (4 OR 5) and (NOT 6) then light 5.

Basic Instructions

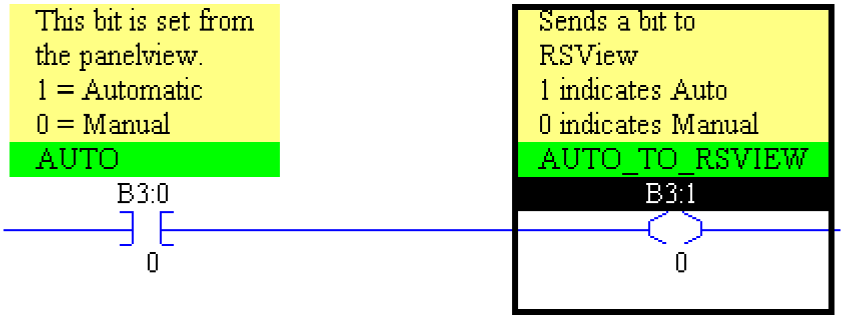
A Little History

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:

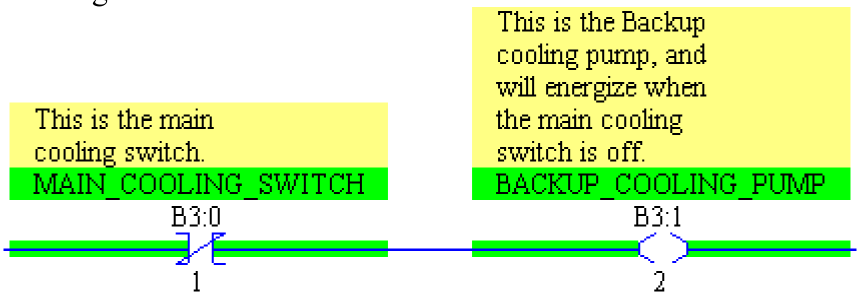
Examine If Closed (XIC)

You will find that most instructions in the SLC, PLC-5, and ControlLogix consist of three character pneumonics. The XIC looks at any 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 can look at any 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:

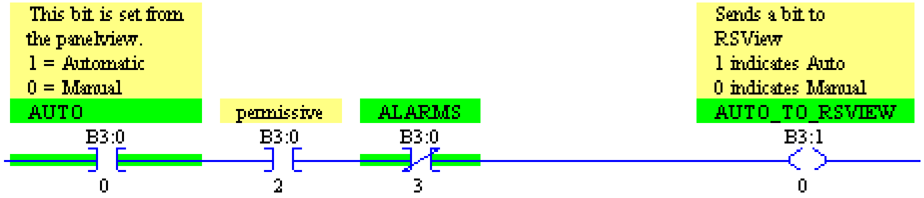

This is the main cooling switch.
MAIN_COOLING_SWITCH
B3:0
1
This is the Backup cooling pump, and will energize when the main cooling switch is off.
BACKUP_COOLING_PUMP
B3:1
2

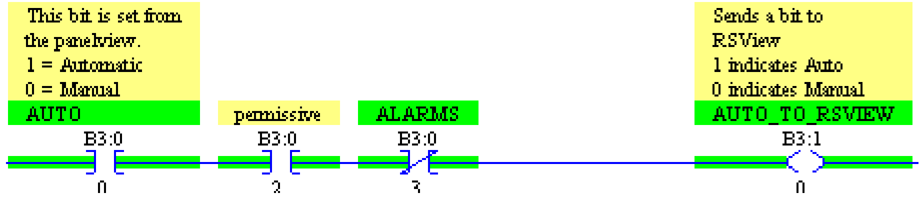
In the above example, you can see that as soon as the Main Cooling Switch is shut off, a bit is set to run the backup cooling pump.

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.

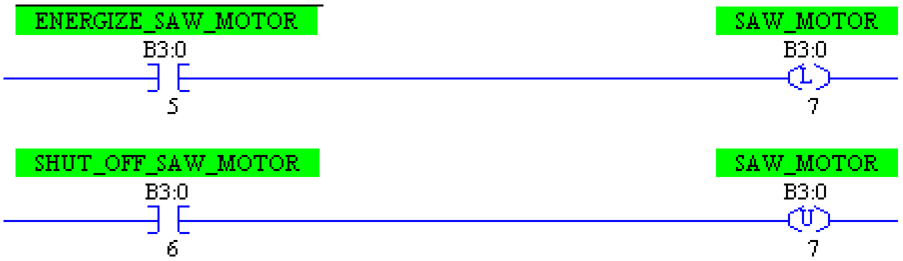
Below you will find two different states of the same rung. The first state shows the rung as false, so a zero is written to B3:1/0. The second state is true, and a 1 will be written to B3:1/0.



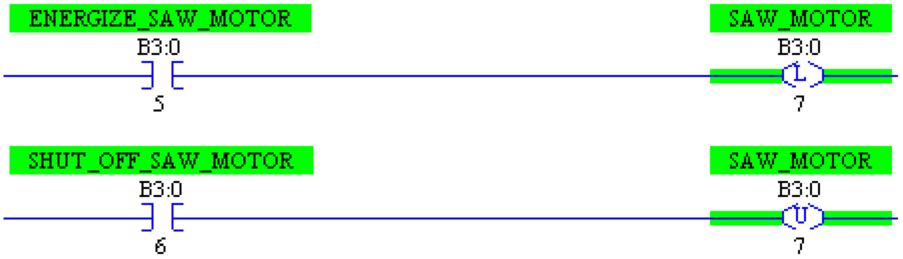


Output To Latch (OTL) and Output To Unlatch (OTU)

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.

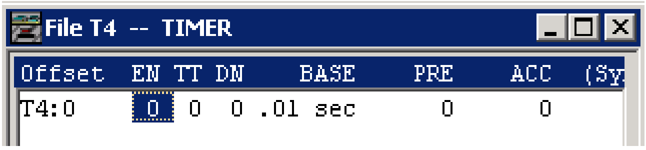


Due to the processor scan cycle, since the unlatch is placed after the latch, if both inputs were to go true, the Unlatch instruction would win, and the output address will be shut off. If the latch was after the unlatch, then the latch would be the last instruction scanned, and therefore the bit would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

By default, timers are stored in the T4 Data file, however other time files can be created.. A timer 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.



File T4 -- TIMER

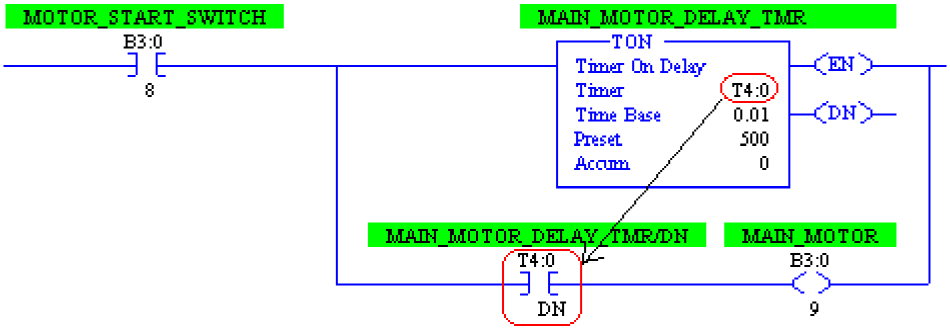
Offset	EN	TT	DN	BASE	PRE	ACC	(Sy
T4:0	0	0	0	.01 sec	0	0	

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 .01, therefore 500 (preset) times .01 (timebase) = 5 second delay.

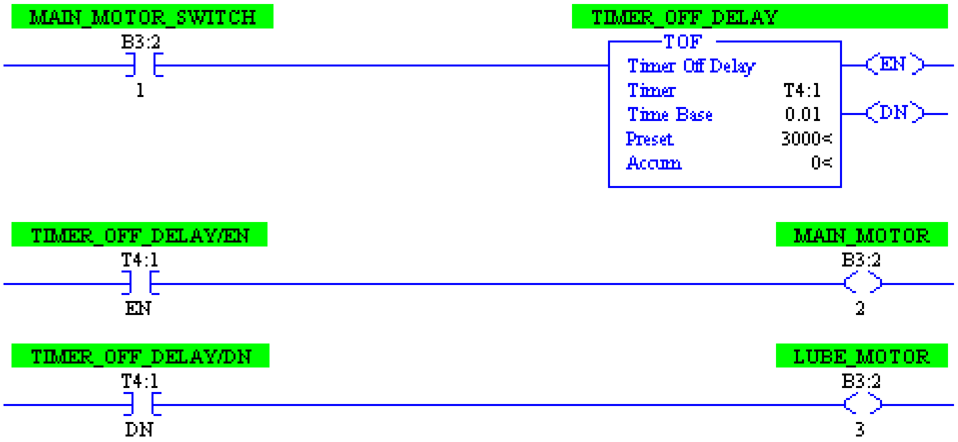
Timer Off Delay (TOF)

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:

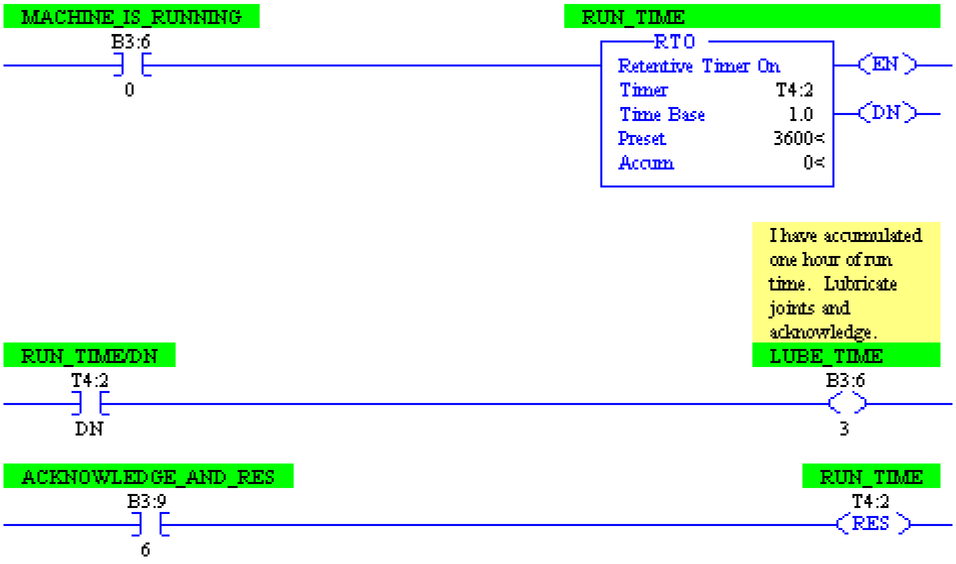


When the motor switch is energized, both the main motor and the lube motor will energize immediately. When the main motor switch is shut off, the main motor shuts off immediately, but since the TOF delays the DN bit from shutting off, the Lube motor will shut off 30 seconds later. Warning: Using the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

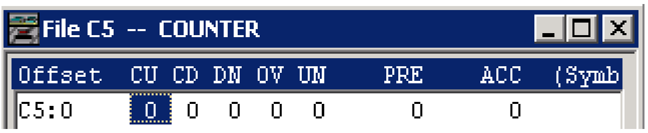
Counters

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

By default, data file C5 stores counters, however, other counter files can be added as well. Below is how the C5 Data file would appear:



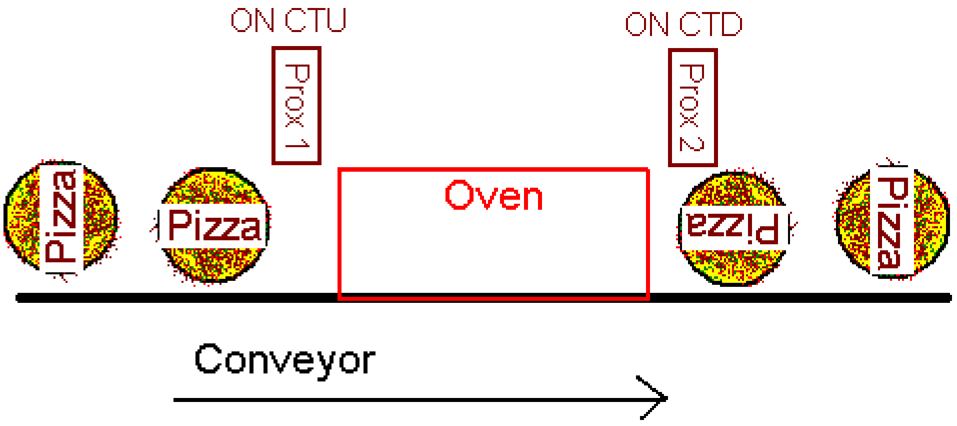
File C5 -- COUNTER

Offset	CU	CD	DN	OV	UN	PRE	ACC	(Symb
C5:0	0	0	0	0	0	0	0	

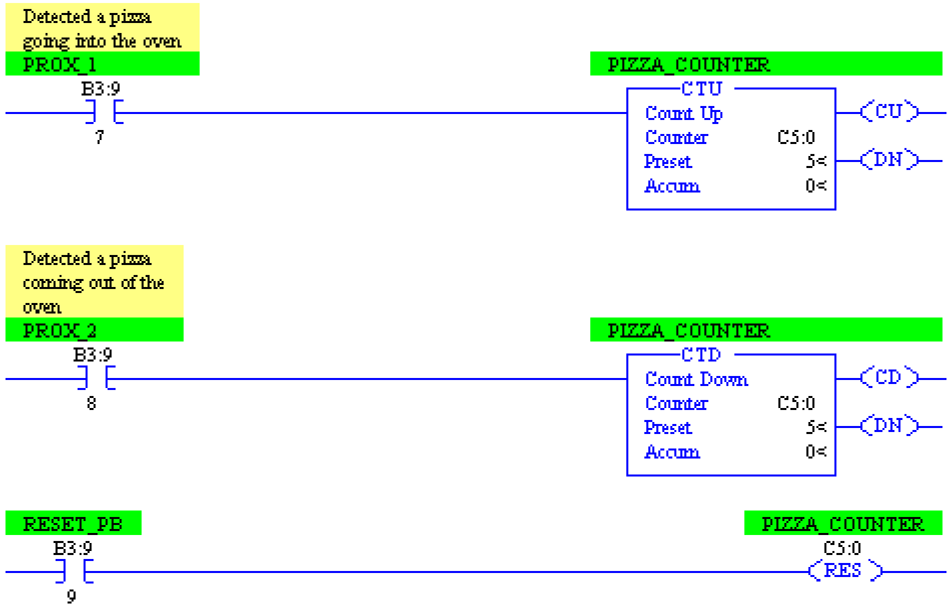
For the CTU instruction: The CU bit is high when the CTU instruction is true. The ACC value increments by the value of 1 each time the CU bit goes high. When the ACC reaches the PRE, the DN bit will be set. The CTU will continue to increment the accumulated value until it reaches the maximum possible value for a 16 bit signed integer (32767). If the CU bit goes high one more time, the OV bit will be set, and the ACC value will go to -32768. 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 16 bit signed integer (-32768). If the CD bit goes high one more time, the UN bit will be set, and the ACC value will go to 32767. 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!



Documentation

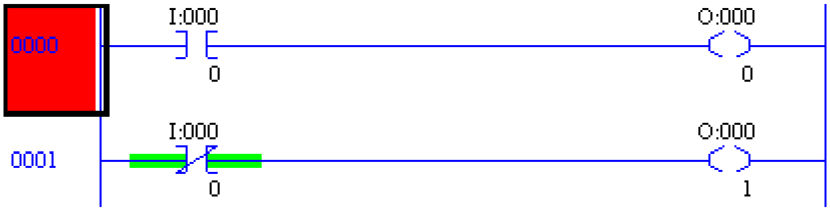
Documentation is very important in any project to help the programmer keep the project organized, and an essential aid to anyone who has to troubleshoot a project after it has been developed. In a properly documented project, a user can quickly locate an output, and easily navigate through a project to find what field devices are preventing an output from energizing. Four main forms of documentation are available for the project: symbols, descriptions, comments, and titles. Symbols and descriptions document individual addresses while comments and page titles document entire rungs or a group of rungs.

Symbols

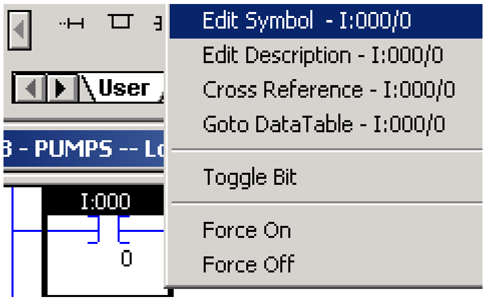
The first documentation we will discuss is the symbol. Symbols are synonymous with address. Once a symbol is attached to a bit in memory, no other bit can be assigned the exact same symbol. When writing logic, the symbol can be used as an address. Symbols can also be used to reflect designators such as the brass tags on valves in a system.

Symbols can also be grouped into categories. These methods will be discussed later in this document.

1) Enter these two rungs of logic (Use address that apply to your system). These two rungs control a main and backup vacuum pump. No one can know this without documentation, so let's start putting some symbols on the addresses.



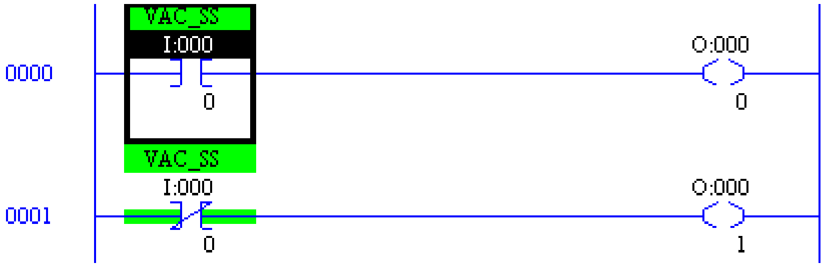
2) One way to add a symbol to an address is to right click the bit in logic, and choose 'Edit Symbol'.



3) Type the symbol name you would like to attach to the bit, and press enter. There are a few rules to follow when entering symbols.

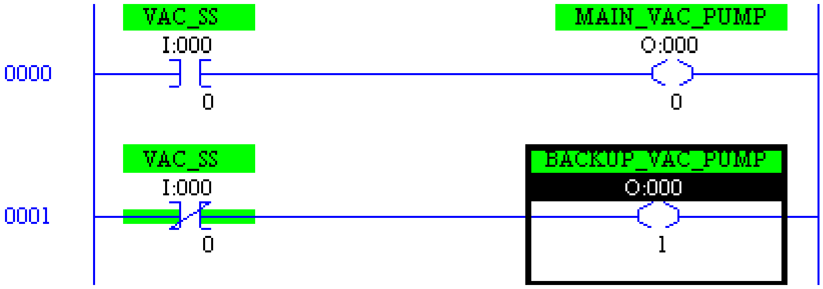
- 20 character limit
- No spaces (underscores are allowed)
- Cannot be only numbers
- Cannot end in a reserved bit or word name such as _pre _acc or _dn
- My not contain the following characters: ~ ` ! @ # $ % ^ & * ( ) + - = [ ] { } \ : ; " < > ' , . / ? |
- Cannot consist of a number followed by a radix designator: D, O, H, E, or B (The software will interpret this as a number with a specific radix)
- The letters O, I, and S cannot be followed by only a number. This will be interpreted as an address

Notice what happens when you type Vac_SS as the symbol:

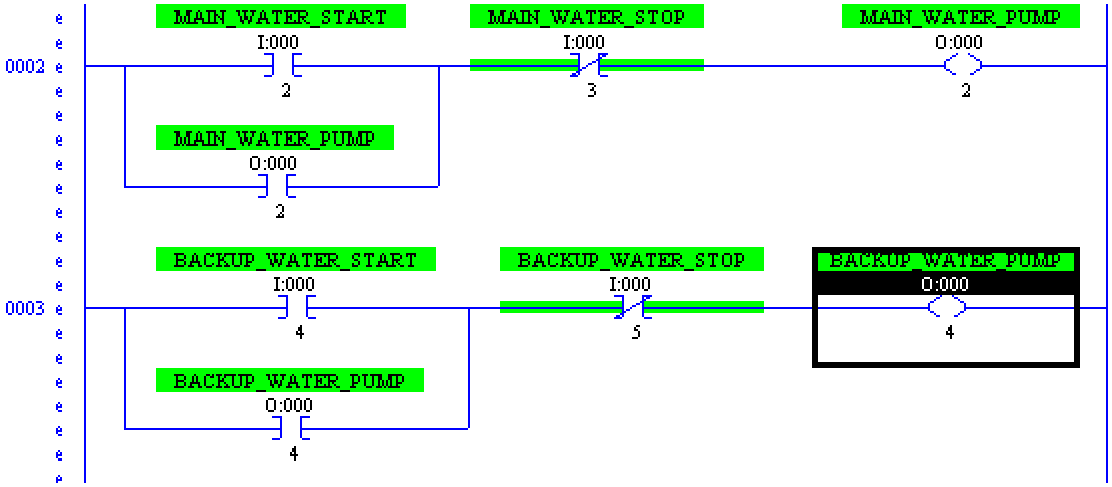


Wherever the address appears in logic, you will also see the symbol.

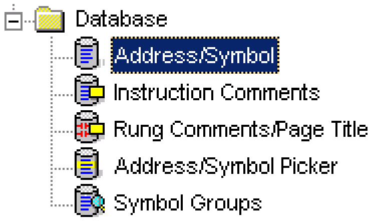
4) Go ahead and document the outputs as follows:



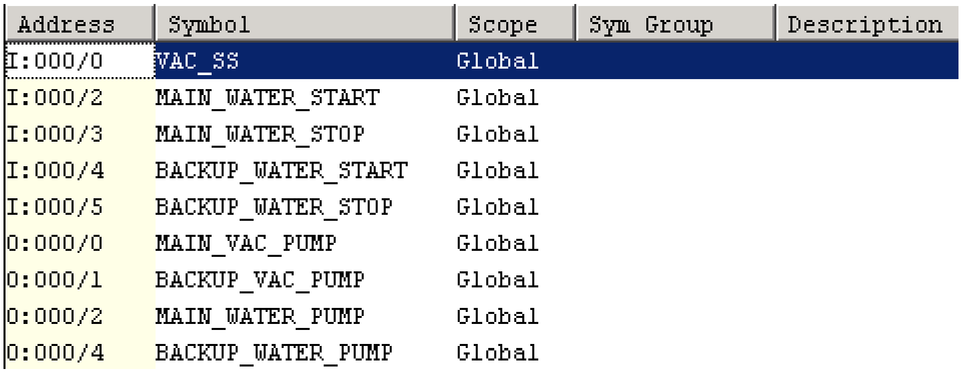
5) For more practice, enter the following logic and documentation into your project. (Use addresses that will work on your system)



Another method to add symbols is to use the Address/Symbol editor found in the Database folder at the bottom of your project tree.

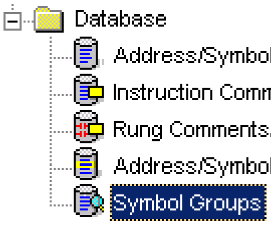


6) When entering a symbol in logic, the symbol actually goes into the documentation database. Open the Address/Symbol editor, and locate the symbols we typed in the ladder view.

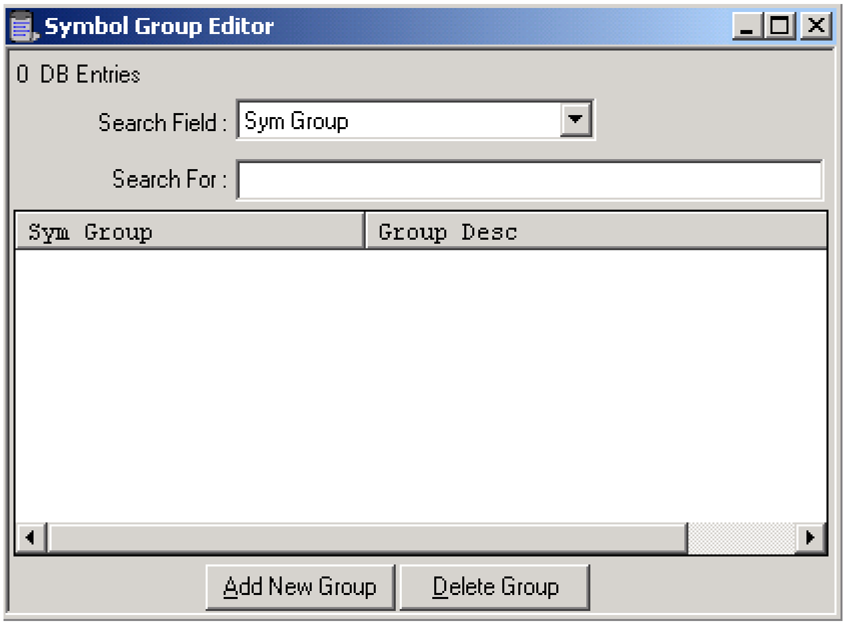


Address	Symbol	Scope	Sym Group	Description
I:000/0	VAC_SS	Global		
I:000/2	MAIN_WATER_START	Global		
I:000/3	MAIN_WATER_STOP	Global		
I:000/4	BACKUP_WATER_START	Global		
I:000/5	BACKUP_WATER_STOP	Global		
O:000/0	MAIN_VAC_PUMP	Global		
O:000/1	BACKUP_VAC_PUMP	Global		
O:000/2	MAIN_WATER_PUMP	Global		
O:000/4	BACKUP_WATER_PUMP	Global		

7) You will notice the symbol editor allows you to enter or delete records from the data base directly. Notice a field for the symbol group. If you are going to add logic using symbols only as your form of addressing, you will need a way to organize your tags into groups so they are easier to find. You do not have the ability to change the group yet because we have not created any. Look at the symbols you have so far, and think about how you would organize them. For this example, we'll create two groups. One for Vacuum symbols, and another for Water symbols. Close out of the Address/Symbol editor, and open 'Symbol Groups' from the database folder in the project tree.



8) You will see a screen similar to the one shown below. Click 'Add new Group' at the bottom of the window.


Symbol Group Editor

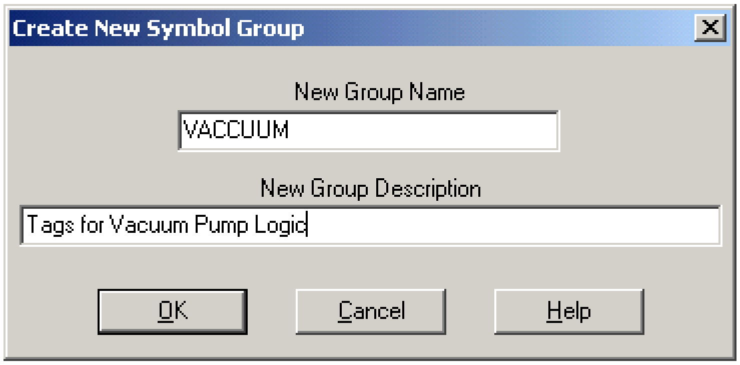
0 DB Entries

Search Field : Sym Group

Search For :

Sym Group	Group Desc

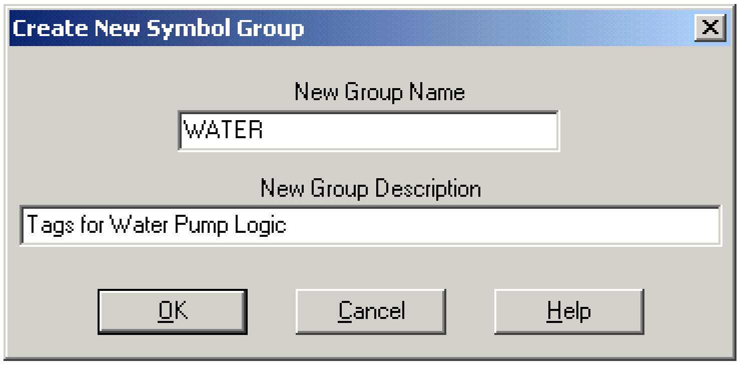
9) Set up your new group as shown below for Vacuum pumps, then press OK.



New Group Name: VACCUUM

New Group Description: Tags for Vacuum Pump Logic

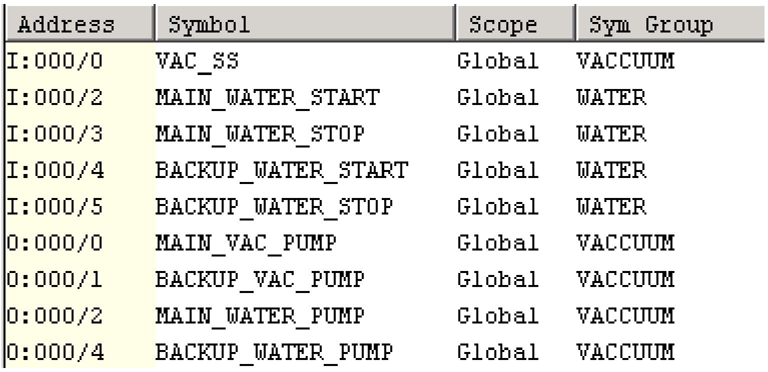
10)Next, Add another new group as shown below:



New Group Name: WATER

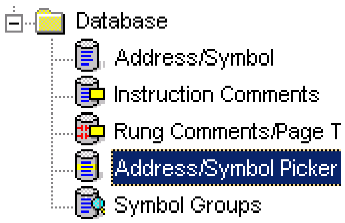
New Group Description: Tags for Water Pump Logic

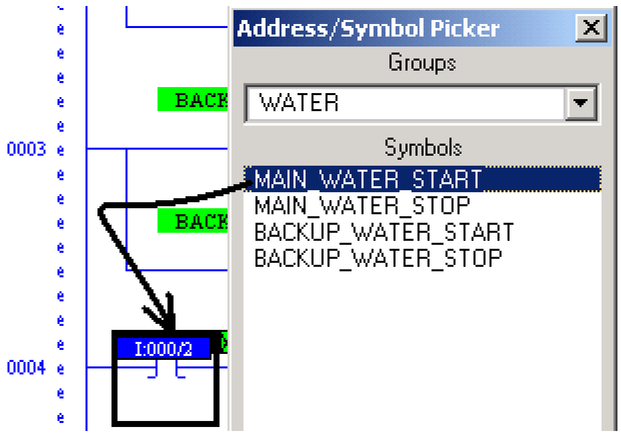
11)Now go back to the Address/Symbol editor, and you will be able to assign each tag to a group as shown.



Address	Symbol	Scope	Sym Group
I:000/0	VAC_SS	Global	VACCUUM
I:000/2	MAIN_WATER_START	Global	WATER
I:000/3	MAIN_WATER_STOP	Global	WATER
I:000/4	BACKUP_WATER_START	Global	WATER
I:000/5	BACKUP_WATER_STOP	Global	WATER
O:000/0	MAIN_VAC_PUMP	Global	VACCUUM
O:000/1	BACKUP_VAC_PUMP	Global	VACCUUM
O:000/2	MAIN_WATER_PUMP	Global	VACCUUM
O:000/4	BACKUP_WATER_PUMP	Global	VACCUUM

12)Now that the symbols are in groups, you could drag symbols from the Address/Symbol picker, and drop them in your logic. Go ahead and open the Address/Symbol picker, and create a test rung to give this method a try.



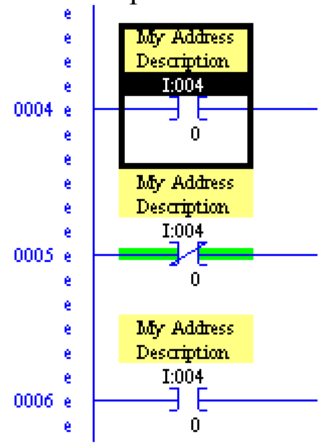


13)Delete your test rung when you see how this feature works. Be sure to explore the other options on the symbol picker such as config, and show descriptions.

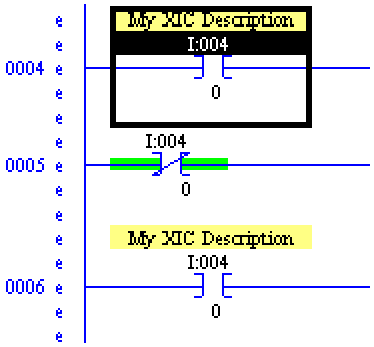
Descriptions

Symbols were synonymous with the addresses, and had many limitations. Symbols were just a designator for a bit, but descriptions have fewer limitations, and can be used multiple times throughout a project. There are two forms of descriptions... Instruction descriptions, and address descriptions. Address descriptions are probably the most commonly used, and appear wherever the address is used in logic. Descriptions can be no longer than 5 lines with 20 characters on each line. Instruction descriptions appear wherever the address is used with a particular instruction. Look at the example below:

Address Descriptions work like this

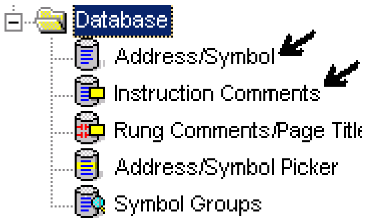


Instruction Descriptions work like this

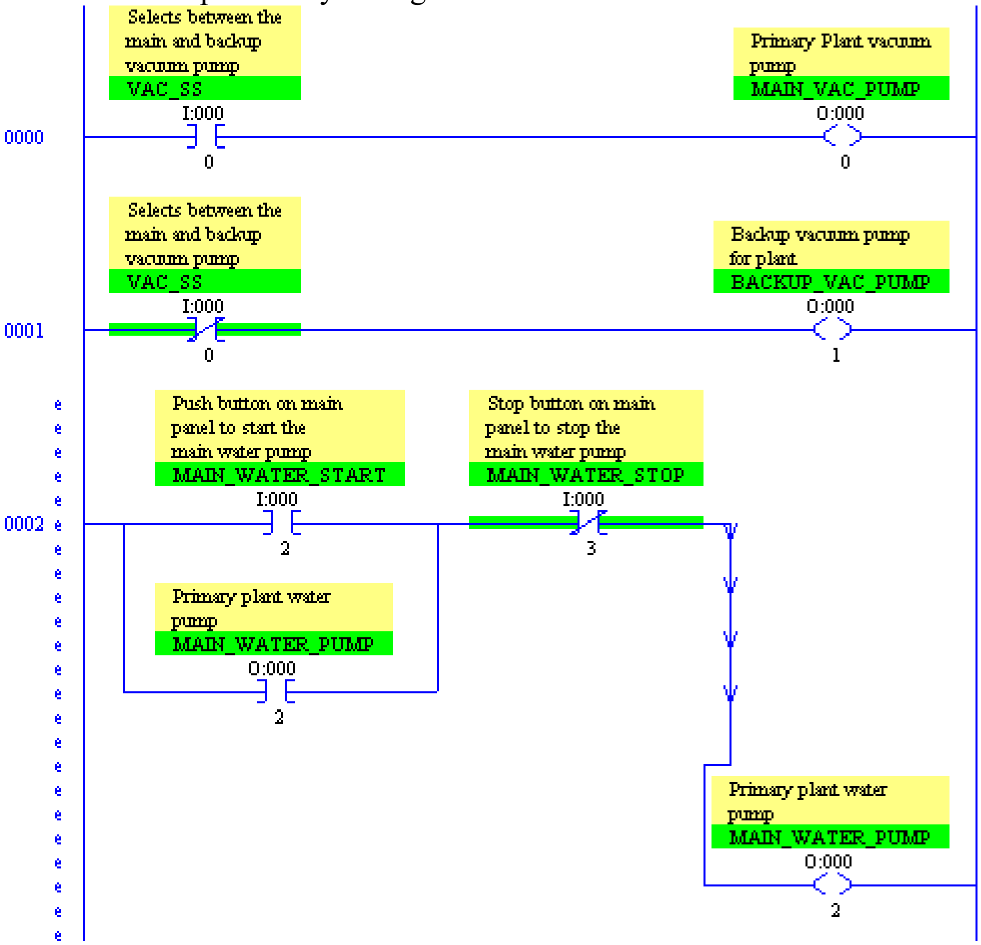


In each case, the description was only applied to the first XIC. When an address description was applied, it appeared at all three locations where the address was used. When the instruction description was applied to the first instruction, it appeared every where the address was used on an XIC instruction throughout the project.

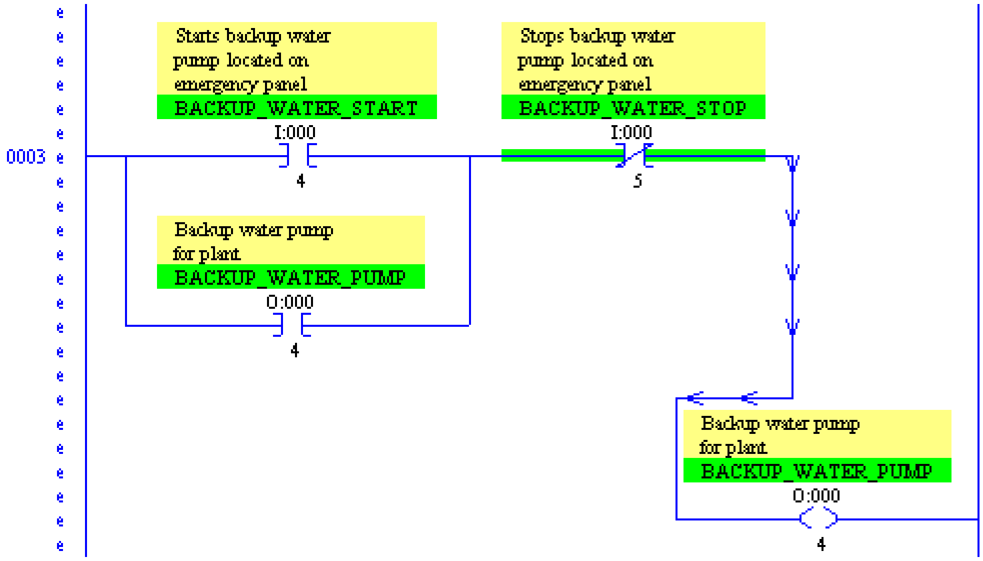
There are several ways to add descriptions to bits. One way is to right click the bit in logic. Another way is go use the database. Address descriptions appear in the Address/Symbol Editor, and Instruction descriptions appear in the Instruction Comments editor.



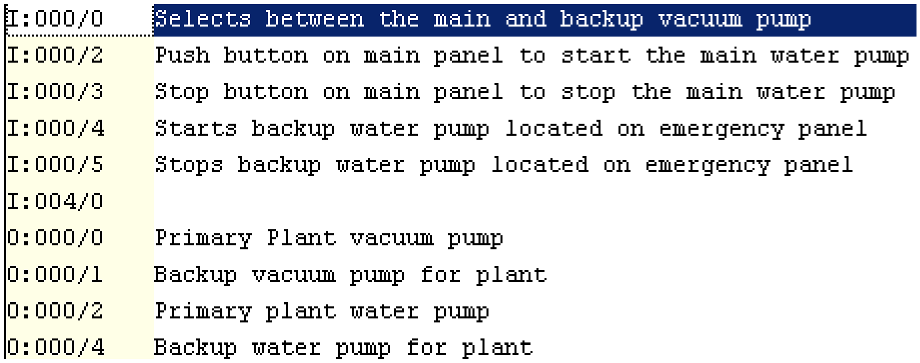
1) For our example, Right click on the instructions in logic, and choose 'Edit Description'. Use Address descriptions in your logic as follows:


Selects between the main and backup vacuum pump
VAC_SS
I:000
0000
0
Primary Plant vacuum pump
MAIN_VAC_PUMP
O:000
0
Selects between the main and backup vacuum pump
VAC_SS
I:000
0001
0
Backup vacuum pump for plant
BACKUP_VAC_PUMP
O:000
1
Push button on main panel to start the main water pump
MAIN_WATER_START
I:000
0002
2
Stop button on main panel to stop the main water pump
MAIN_WATER_STOP
I:000
3
Primary plant water pump
MAIN_WATER_PUMP
O:000
2
Primary plant water pump
MAIN_WATER_PUMP
O:000
2

Continued on next page


0003
Starts backup water pump located on emergency panel
BACKUP_WATER_START
I:000
4
Stops backup water pump located on emergency panel
BACKUP_WATER_STOP
I:000
5
Backup water pump for plant
BACKUP_WATER_PUMP
O:000
4
Backup water pump for plant
BACKUP_WATER_PUMP
O:000
4

If you look in the Address/Symbol editor, you will see your changes have been logged to the database. Be sure to save your work.

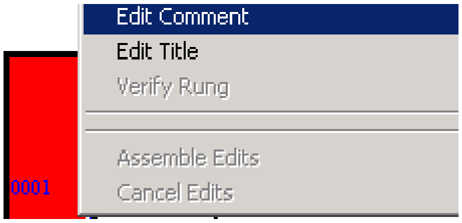


I:000/0	Selects between the main and backup vacuum pump
I:000/2	Push button on main panel to start the main water pump
I:000/3	Stop button on main panel to stop the main water pump
I:000/4	Starts backup water pump located on emergency panel
I:000/5	Stops backup water pump located on emergency panel
I:004/0	
O:000/0	Primary Plant vacuum pump
O:000/1	Backup vacuum pump for plant
O:000/2	Primary plant water pump
O:000/4	Backup water pump for plant

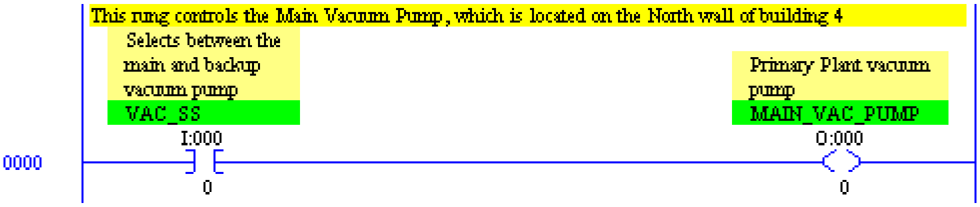
Rung Comments

Rung comments describe the purpose of a rung in ladder logic. Rung comments can be up to 64,000 characters long, and can be attached to the rung number or the output. If the comment is attached to the output address, the rung comment will appear wherever the address appears as an output on a rung. With good programming practice, your program will probably only have one OTE instruction for any given address, so the comment will appear only one time. However, in the case of a Latch and Unlatch instructions, the comment would appear at both locations if the Latch and Unlatch are in separate rungs.

1) To add a description, right click the rung number just to the left of the rung, and choose 'Edit comment'.



2) Go ahead and add comments to all four rungs of logic explaining the purpose of the rung. Below is an example of rung 0.


This rung controls the Main Vacuum Pump, which is located on the North wall of building 4

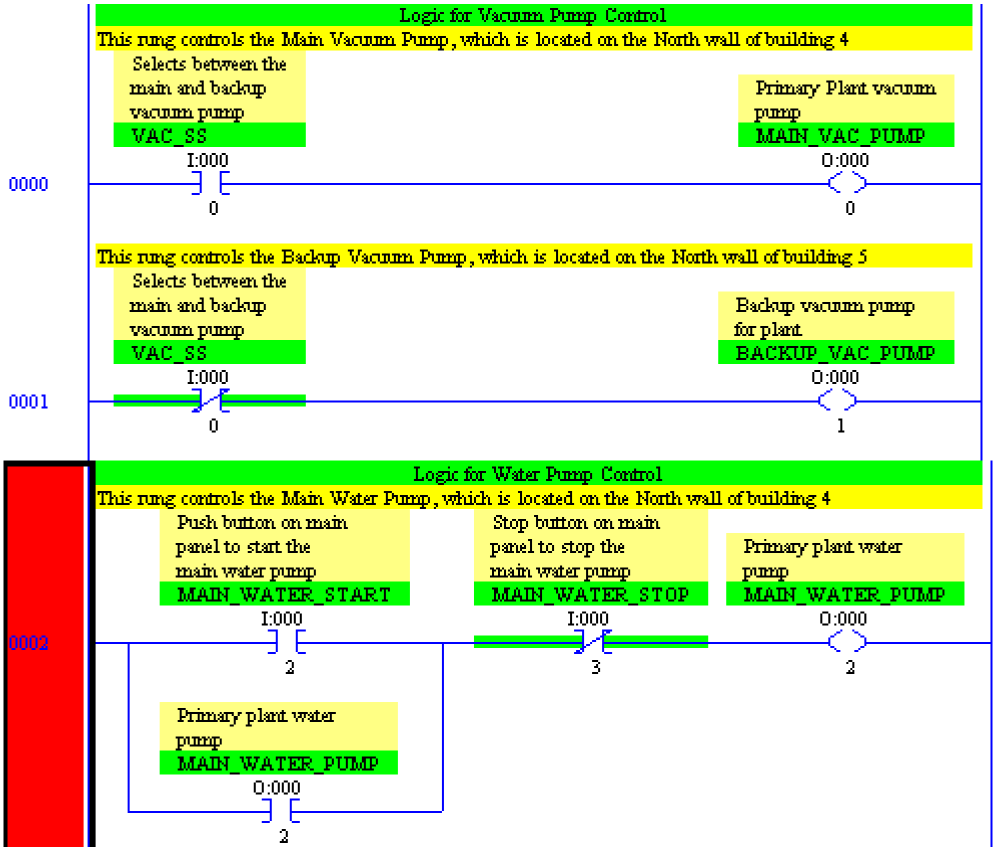
Selects between the main and backup vacuum pump

Primary Plant vacuum pump

Page Titles

Page titles are a very important part of documentation. Page titles should be used as 'thumb tabs' in routines to mark sections or blocks of logic that perform a specific purpose. Advanced Diagnostics uses page titles to quickly locate an output in the program, so the troubleshooter can then use cross referencing to find what field devices have failed. In later versions of RSLogix, Page Titles (Advanced Diagnostics) can be integrated into the project tree. With page titles, you are limited to 80 characters.

1) Adding Page Titles are very similar to adding rung comments. Right click the rung number and 'Edit Title'. Titles and comments can be modified from the same window. In our sample logic, there are two sections. One for Vacuum Pumps, and the other for Water pumps. Add your Titles to reflect the purpose of each section as shown below.



Logic for Vacuum Pump Control

This rung controls the Main Vacuum Pump, which is located on the North wall of building 4

0000

Selects between the main and backup vacuum pump
VAC_SS
I:000
0

Primary Plant vacuum pump
MAIN_VAC_PUMP
O:000
0

This rung controls the Backup Vacuum Pump, which is located on the North wall of building 5

0001

Selects between the main and backup vacuum pump
VAC_SS
I:000
0

Backup vacuum pump for plant
BACKUP_VAC_PUMP
O:000
1

Logic for Water Pump Control

This rung controls the Main Water Pump, which is located on the North wall of building 4

0002

Push button on main panel to start the main water pump
MAIN_WATER_START
I:000
2

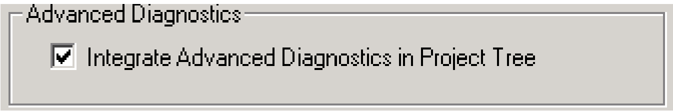
Stop button on main panel to stop the main water pump
MAIN_WATER_STOP
I:000
3

Primary plant water pump
MAIN_WATER_PUMP
O:000
2

Primary plant water pump
MAIN_WATER_PUMP
O:000
2

2) To integrate page titles into the project tree, click Tools | Options on the menu bar.

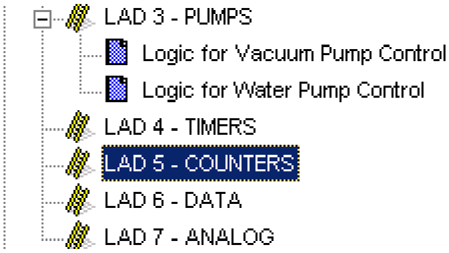
In the middle of the window, choose 'Integrate Advanced Diagnostics Into Project Tree'. Apply your changes then click OK.



Advanced Diagnostics

☑ Integrate Advanced Diagnostics in Project Tree

3) Now look in your project tree. You see a + next to ladder 3. Expanding ladder 3 reveals the sections (page titles) we just created. Double click any page title, and RSLogix will take you directly to that section of logic.



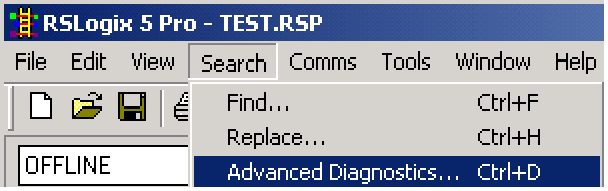
LAD 3 - PUMPS
Logic for Vacuum Pump Control
Logic for Water Pump Control
LAD 4 - TIMERS
LAD 5 - COUNTERS
LAD 6 - DATA
LAD 7 - ANALOG

Advanced Diagnostics

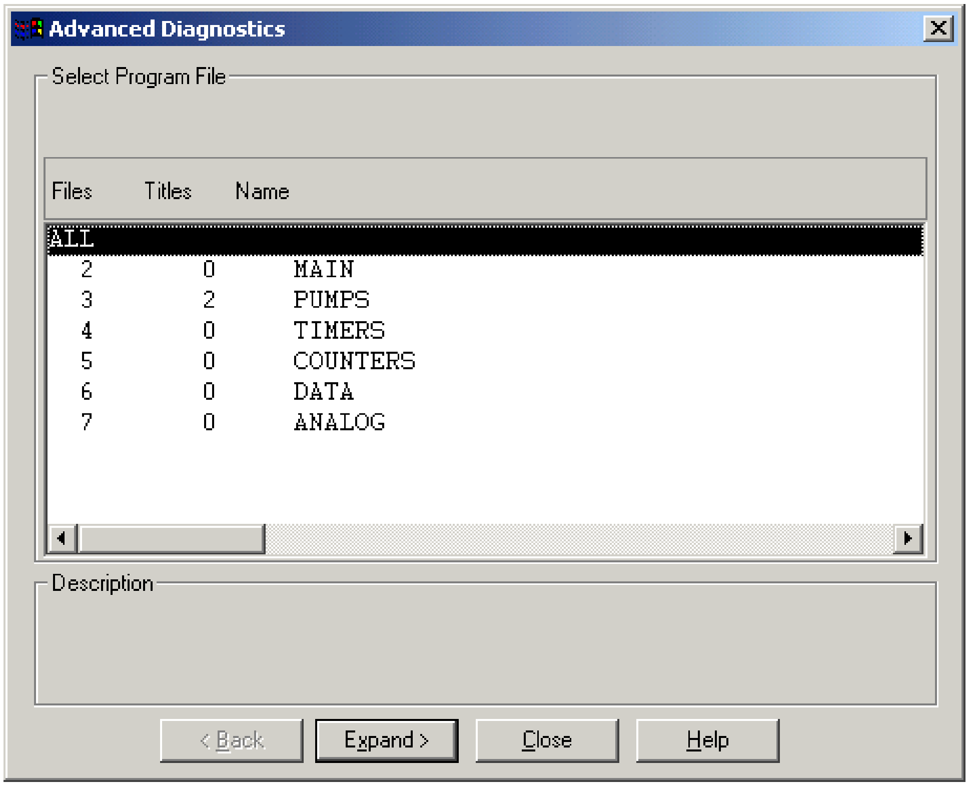
Advanced Diagnostics is a power troubleshooting tool built into RSLogix. Advanced diagnostics helps a user quickly located an output in the ladder logic. There are three steps: 1) Choose from a list of routines; 2) Choose from a list of page titles within the routine; and 3) Choose the output within the page title (block of logic) that you are looking for.

In this example, we want to figure out what needs to happen for the backup water pump to run.

1) To start Advanced Diagnostics, you must have a project open then click Search | Advanced diagnostics from the menu bar. You will notice the short cut is to press CTRL-D.

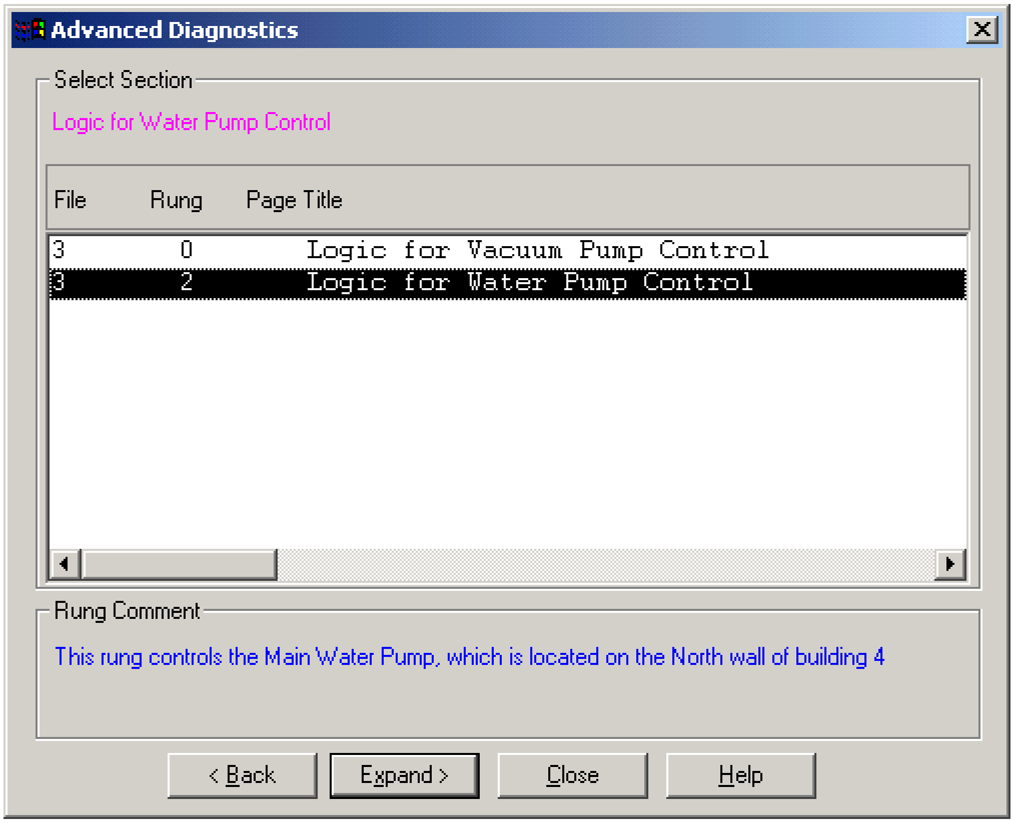


2) The first screen that appears is asking what routine you want to work with. From this screen, you can also see how many titles are in each routine. Since our Water Pump is down, double click 'PUMPS', or highlight 'PUMPS', then click 'Expand'.



Files	Titles	Name
ALL		
2	0	MAIN
3	2	PUMPS
4	0	TIMERS
5	0	COUNTERS
6	0	DATA
7	0	ANALOG

3) Next, RSLogix is asking what page title you would like to view. Notice when you highlight a page title, the rung comment also appears for the rung in which the page title resides. Since our problem is with the backup water pump, open the page title for water pump control.



Advanced Diagnostics

Select Section

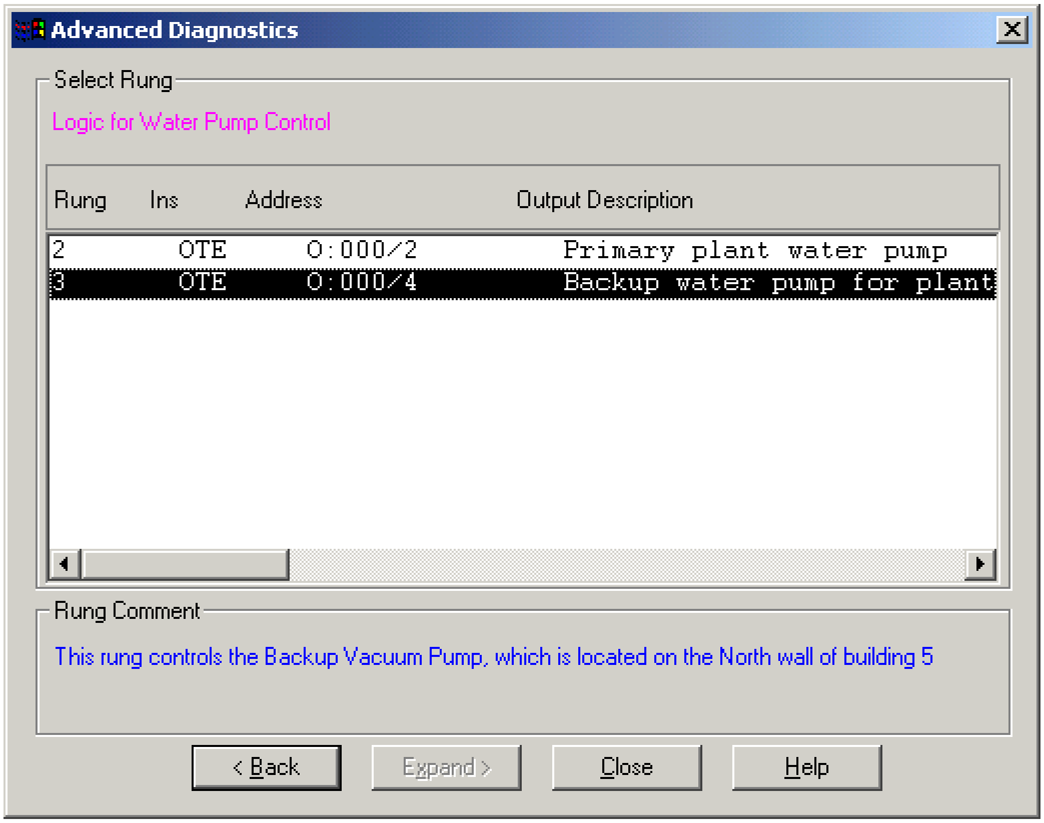
Logic for Water Pump Control

File	Rung	Page Title
3	0	Logic for Vacuum Pump Control
3	2	Logic for Water Pump Control

Rung Comment

This rung controls the Main Water Pump, which is located on the North wall of building 4

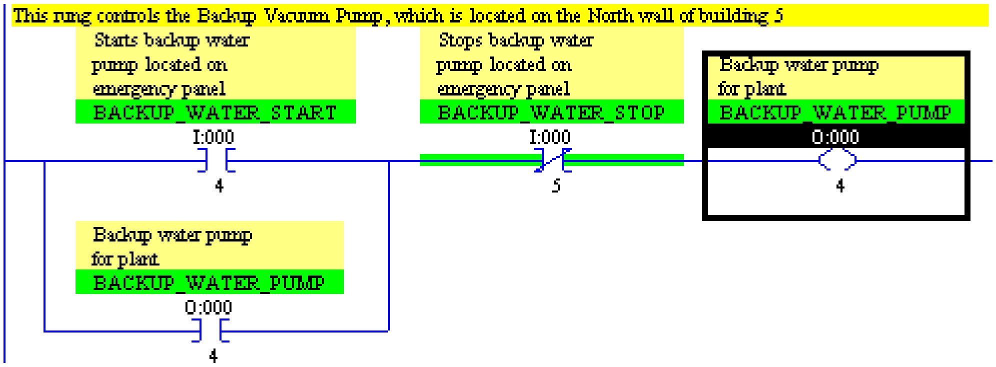
4) Next, you are asked which output to look at within the page title. Notice when you click an output, the rung description appears at the bottom. You also see the description of the output in the description field of the list. If you click on the backup water pump, nothing appears to happen on the advanced diagnostics screen, but something did happen in logic. Close the advanced diagnostics screen, and you at the location in logic where you need to be.



Logic for Water Pump Control

Rung	Ins	Address	Output Description
2	OTE	O:000/2	Primary plant water pump
3	OTE	O:000/4	Backup water pump for plant

This rung controls the Backup Vacuum Pump, which is located on the North wall of building 5

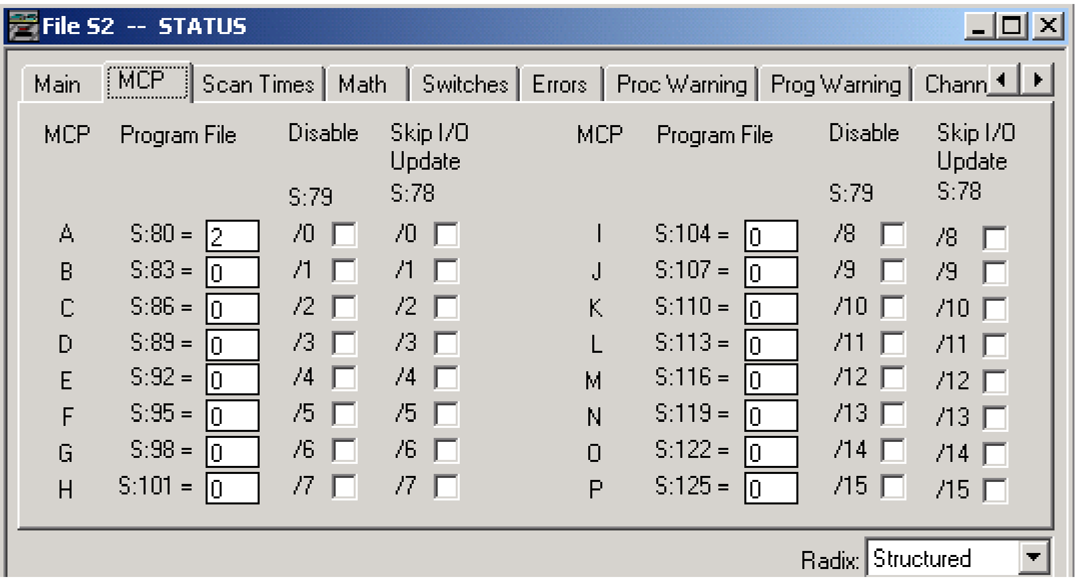

This rung controls the Backup Vacuum Pump, which is located on the North wall of building 5

Starts backup water pump located on emergency panel
BACKUP_WATER_START
I:000
4
Stops backup water pump located on emergency panel
BACKUP_WATER_STOP
I:000
5
Backup water pump for plant
BACKUP_WATER_PUMP
O:000
4
Backup water pump for plant
BACKUP_WATER_PUMP
O:000
4

You can now see why the backup pump is not on. In this example, all instructions point to real devices, but if internal bits were used, you would now have to use Cross-Referencing to trace the field devices.

JSR Instruction

In the S2 status file, you will find an MCP tab which lists all the Main Control Programs that are scheduled to execute every scan. In the snapshot below, ladder 2 is the only routine scheduled to execute every scan. (This is the default setting). Other ladders, such as Ladder 3, 4, or 5 are ignored by the processor unless the processor is instructed to execute those routines by a JSR statement (Jump to Sub Routine).



MCP	Program File	Disable S:79	Skip I/O Update S:78	MCP	Program File	Disable S:79	Skip I/O Update S:78
A	S:80 = 2	/0 □	/0 □	I	S:104 = 0	/8 □	/8 □
B	S:83 = 0	/1 □	/1 □	J	S:107 = 0	/9 □	/9 □
C	S:86 = 0	/2 □	/2 □	K	S:110 = 0	/10 □	/10 □
D	S:89 = 0	/3 □	/3 □	L	S:113 = 0	/11 □	/11 □
E	S:92 = 0	/4 □	/4 □	M	S:116 = 0	/12 □	/12 □
F	S:95 = 0	/5 □	/5 □	N	S:119 = 0	/13 □	/13 □
G	S:98 = 0	/6 □	/6 □	O	S:122 = 0	/14 □	/14 □
H	S:101 = 0	/7 □	/7 □	P	S:125 = 0	/15 □	/15 □

Radix: Structured

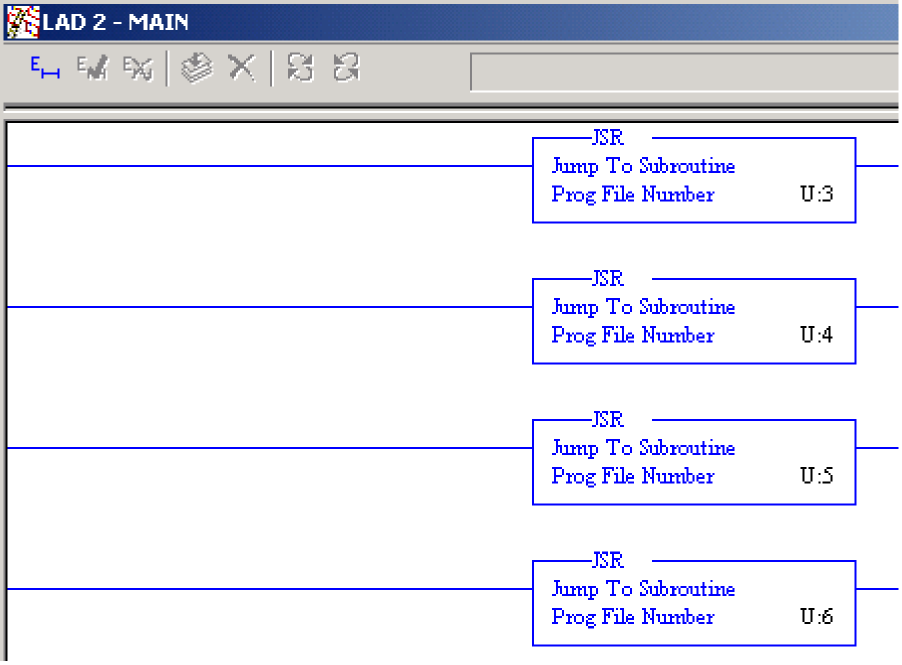
Advantages of Subroutines:

Subroutines can be used to organize a project into sections. All the logic could be placed in the same routine, but doing so would make locating a particular section of logic more difficult. For example: A programmer might create a routine such as Ladder 3 for all the pumps in a project. Ladder 4 could be used for Alarms, and ladder 5 could be used to store all logic that controls valve actions.   Organizing the logic this way would simplify the troubleshooting process because the troubleshooter would have less logic to search through when tracking down the reason why a valve malfunctioned. Organizing logic in such a way also helps the programmer keep track of where all the logic is located as he writes the program.

Organizing a project into subroutines can also decrease the amount of time required to complete one program scan. Many projects require that certain calculations only need to be performed once a day. If this logic is placed in it's own routine, we only need to call this routine with a JSR statement when the calculations need to be performed.

Typical Subroutine Usage:

Many programmers will only have JSR statements in ladder 2 that will execute each subroutine. This is the simplest usage of the JSR instruction. In this situation, each routine is called one at a time.



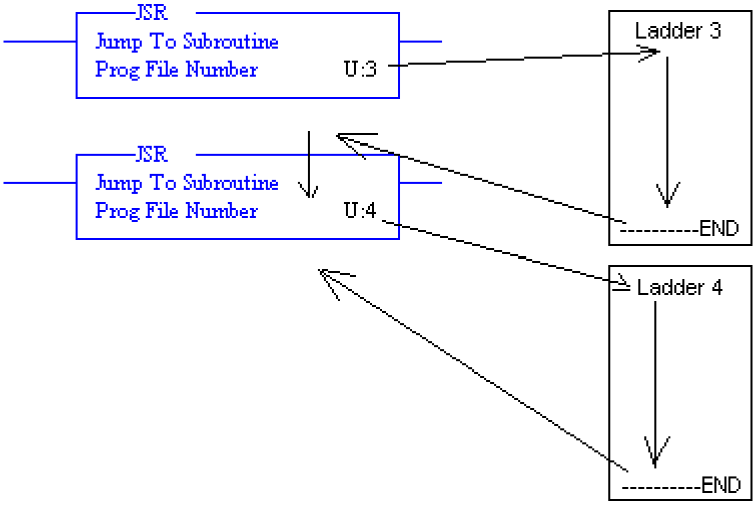
JSR
Jump To Subroutine
Prog File Number U:3

JSR
Jump To Subroutine
Prog File Number U:4

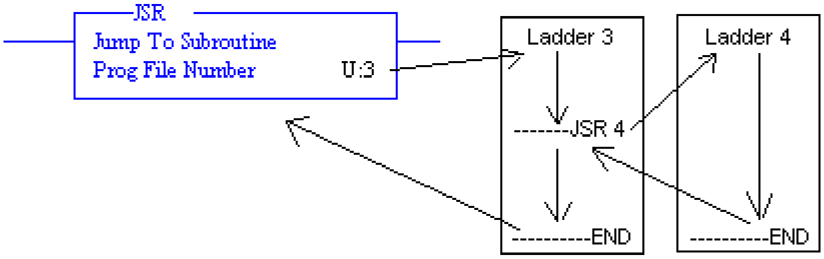
JSR
Jump To Subroutine
Prog File Number U:5

JSR
Jump To Subroutine
Prog File Number U:6

When the processor sees the JSR to ladder 3, the processor will go to the top of ladder 3, execute ladder 3, and when it reaches the end of ladder 3, will come back to the very next instruction in ladder 2. This is the JSR to ladder 4. Each time, we are only going 1 routine deep...



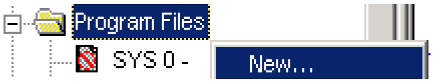
If a JSR would have been placed in ladder 3, this would be called "Nesting" subroutines. You can only nest 8 routines deep before the processor faults. In this example, we are nesting 2 deep.



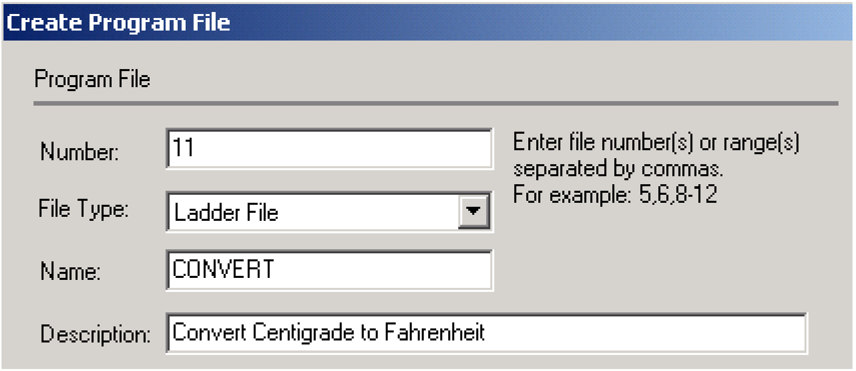
Passing Parameters:

The JSR statement can be used as a function to pass parameters (values) to a routine. The routine will then store it's parameters in it's own workspace to perform operations such as converting centigrade to Fahrenheit, and return an answer to the routine that called the conversion. This exercise will demonstrate such a conversion.

1) Right click the program files folder and create a new routine.



2) For this example, we will create ladder 11. The name will be "Convert", and the description will be "Convert Centigrade to Fahrenheit".



Create Program File

Program File

Number: 11

Enter file number(s) or range(s) separated by commas. For example: 5,6,8-12

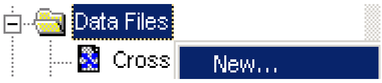
File Type: Ladder File

Name: CONVERT

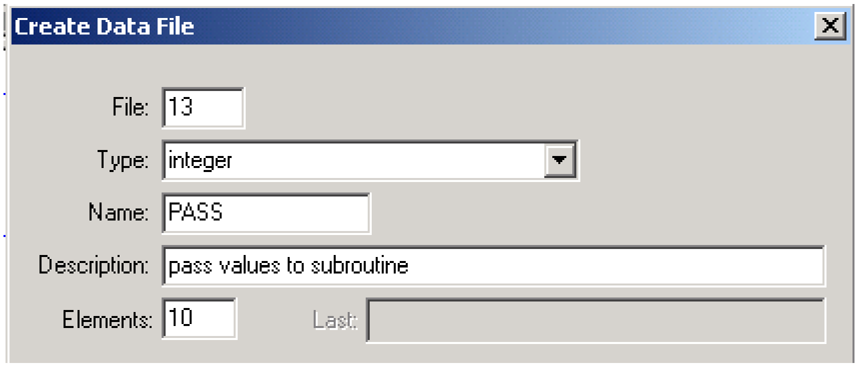
Description: Convert Centigrade to Fahrenheit

3) Next, we will create two data files. The first data file is the file we are going to use to pass data to the subroutine. The second data file will act as a 'workspace' for our conversion subroutine.

4) Right click Data Files, and create a new file.



5) For this example, the first file we will create is going to file file 13. It will be an integer file consisting of 10 elements as shown:


Create Data File

File: 13

Type: integer

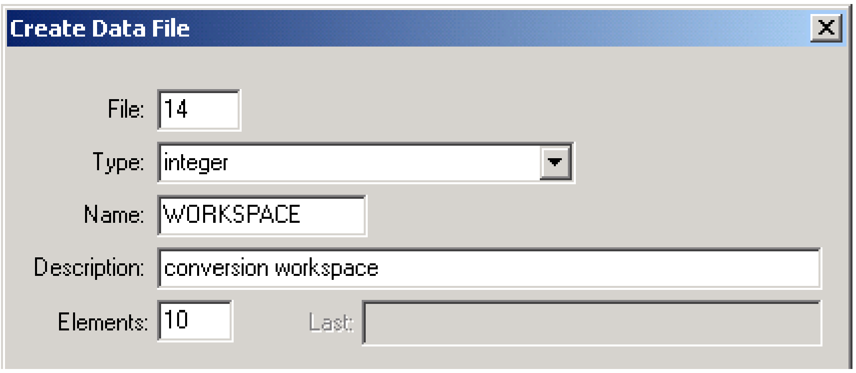
Name: PASS

Description: pass values to subroutine

Elements: 10

Last:

6) Create another file, file 14 as integer consisting of 10 elements as shown. This will be the workspace for our new routine.


Create Data File

File: 14

Type: integer

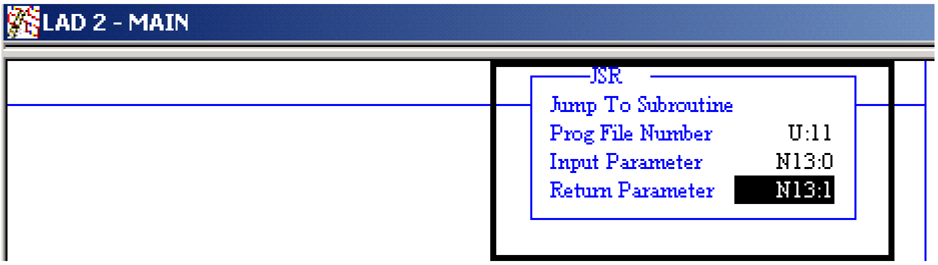
Name: WORKSPACE

Description: conversion workspace

Elements: 10

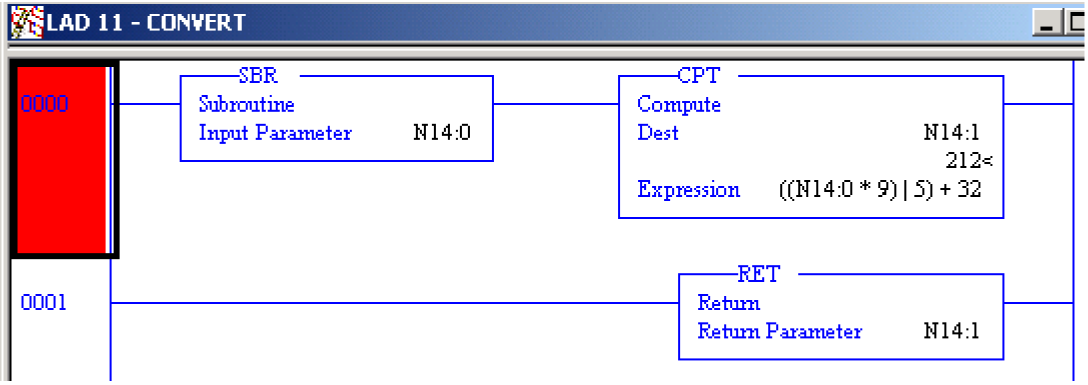
Last:

7) Next, we will add a JSR statement to Ladder 2 that will take data from N13:0, and pass this value to the subroutine it is calling. When the subroutine returns a calculated value, this value will be stored in N13:1. Set up your JSR as shown:


LAD 2 - MAIN

JSR
Jump To Subroutine
Prog File Number U:11
Input Parameter N13:0
Return Parameter N13:1

8) Now we are going to add some logic to ladder 11. The SBR instruction tells the processor where to store the value it was passed from the JSR instruction. The RET statement at the bottom of the routine tells the processor where to get data from after the calculations are complete to send back to the JSR statement, so this value can be stored in it's return parameter. Insert your logic as shown:



LAD 11 - CONVERT

0000

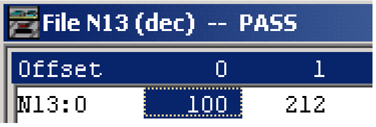
SBR
Subroutine
Input Parameter N14:0

CPT
Compute
Dest N14:1
212<
Expression ((N14:0 * 9) | 5) + 32

0001

RET
Return
Return Parameter N14:1

9) Now, open Data file 13. When you place a value into N13:0, the conversion for this value will be displayed in N13:1. In this example, I have entered the value of 100 degrees Centigrade in N13:0. The value returned by the subroutine is shown in N13:1.



File N13 (dec) -- PASS

Offset	0	1
N13:0	100	212

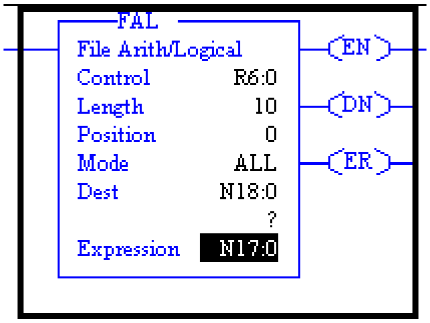
10) Summary:
   1. The JSR called ladder 11, and passed the value of N13:0 (value of 100) to ladder 11.
   2. The SBR statement in ladder 11 instructed the processor to store the value it was passed into a workspace so it can be operated on (N14:0)
   3. The CPT statement in ladder 11 performed the conversion and stored the value into N14:1.
   4. The RET statement instructed the processor to return the value of N14:1 (value of 212) back to the JSR statement that called it.
   5. The JSR statement in ladder 2 instructed the processor to store the value to N13:1 after it was received from the subroutine.

11) Note: In this example, we are only performing one conversion, but if this was in a large project, any data table value can now be converted and returned from a JSR anywhere in the project. When complex calculations are required, the programmer only needs to set up the calculations one time. Any value can be passed to the routine performing the calculations from anywhere in the program at any time (Not to exceed the stack limit of 8 nested routines).

FAL-File Arithmetic Logic

The FAL instruction is very versatile. It can copy data from one file to another, copy data from a file to an element, or an element to a file. The main purpose of the FAL instruction, however is to perform math (or manipulate) the data as it is moved across to another file.

Here is what the FAL Instruction looks like in logic:



FAL
File Arith/Logical
Control R6:0
Length 10
Position 0
Mode ALL
Dest N18:0
?
Expression N17:0

The **CONTROL** element is just a workspace for the FAL to perform it's job. This workspace keeps track of the status of the FAL instruction.

The **LENGTH** is the number of elements in the file you are operating on.

The **POSITION** indicates the progress of the FAL instruction as it operates on the file. The instruction is considered done when the position equals the length.

There are three **MODES** of operation:

ALL – All calculations are performed in the current scan.

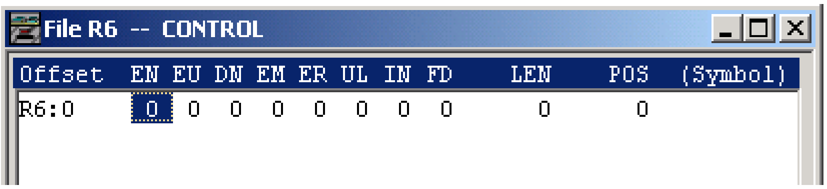
INCREMENTAL – One element is operated on for each false to true rung transition.

NUMERIC – A certain number of elements are operated on per scan once the rung is true.

The **DESTINATION** is where the data is stored after the expression has been executed. This can be in the form of a single element such as N18:0, or a file such as #N18:0 (note the # sign). If you use the # sign before the file type, each operation will be stored to a different element. For example: The first operation's result will be stored to N18:0.... The second operation's result will be stored to N18:1, etc...

The **EXPRESSION** is the 'formula' used in the calculation.

This example uses R6:0 as the control element. If you are going to use multiple FAL statements, each statement must have a separate control elements such as R6:1, R6:2, etc.... Here is a snapshot of the control file:



File R6 -- CONTROL

Offset	EN	EU	DN	EM	ER	UL	IN	FD	LEN	POS	(Symbol)
R6:0	0	0	0	0	0	0	0	0	0	0	

EN – This is the ENABLE bit. It is set when the rung goes from false to true.
In INCREMENTAL mode, this bit will follow the rung condition.
In NUMERIC mode, or ALL mode, the EN bit remains set until the instruction is finished.

DN – This is the DONE bit. It is set when the instruction completes all operations. In NUMERIC mode, the DN bit is reset when the instruction is complete if the rung is false.... Otherwise, the EN bit will reset the DN bit when it goes true again.

ER – This is the ERROR bit. It is set if the instruction generates an overflow, and the instruction stops it's operation. If this happens, the ER bit is reset by the ladder logic. The POS will indicate the position of the value (in the file) that caused the overflow condition to occur.

LEN – This is the length of the number of elements you are operating on.

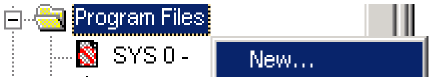
POS – This is the position of the FAL instruction as it operates through the file.

Other bits such as (EM) Empty (EU) Queue (IN) Inhibit (FD) Found, and (UL) Unload are not used for this particular instruction.

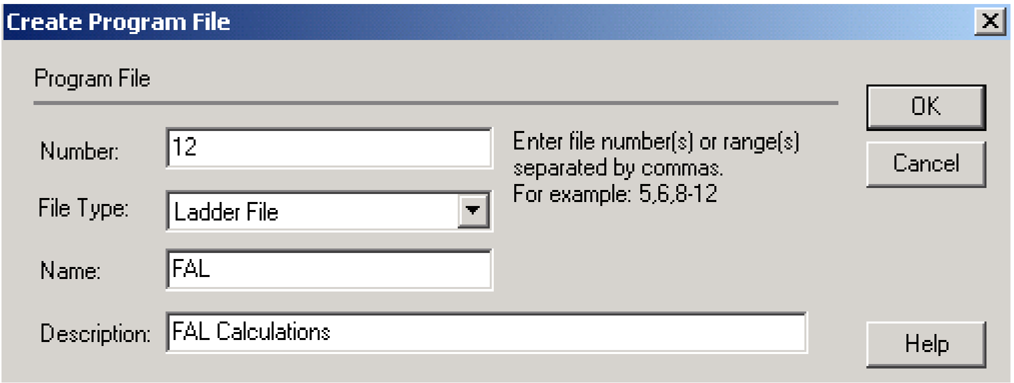
Making it work.

In this example, we are going to take 10 elements, and convert them all from Centigrade to Fahrenheit, and store the result in a different file.

1) First, let's right-click the program file folder to create a new routine. We are going to use this routine just for the FAL instructions.



2) This will be ladder 12, and the name will be FAL as shown.



Create Program File

Program File

Number: 12

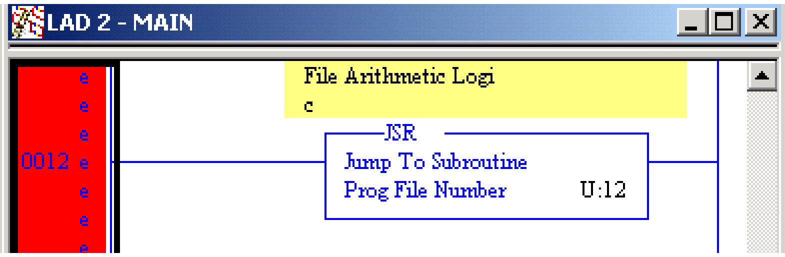
File Type: Ladder File

Name: FAL

Description: FAL Calculations

Enter file number(s) or range(s) separated by commas.
For example: 5,6,8-12

3) Now, we need to go back to ladder 2, and add a JSR statement so ladder 12 will execute.



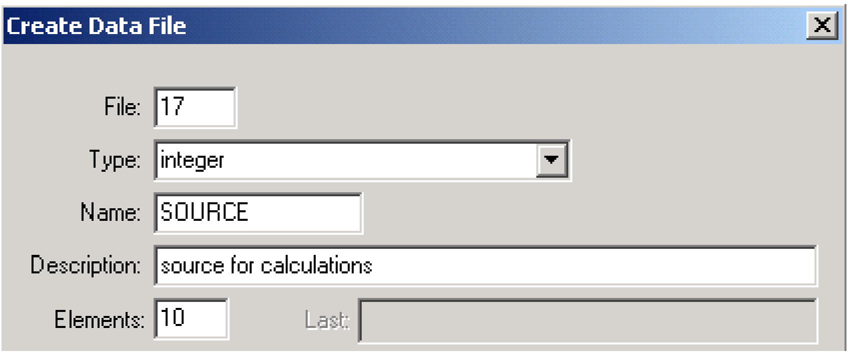
LAD 2 - MAIN

0012

File Arithmetic Logi
c

JSR
Jump To Subroutine
Prog File Number U:12

4) Now let's create a couple data files that we can use for our math. Right click on the Data File folder, then select New to create a new data file.



Create Data File

File: 17

Type: integer

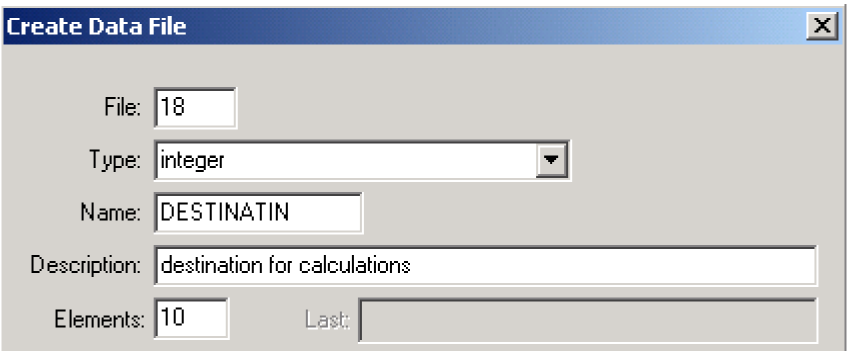
Name: SOURCE

Description: source for calculations

Elements: 10

Last:

5) Create another data file that will act as the destination. We will place the data into this file after we perform our data conversion:



Create Data File

File: 18

Type: integer

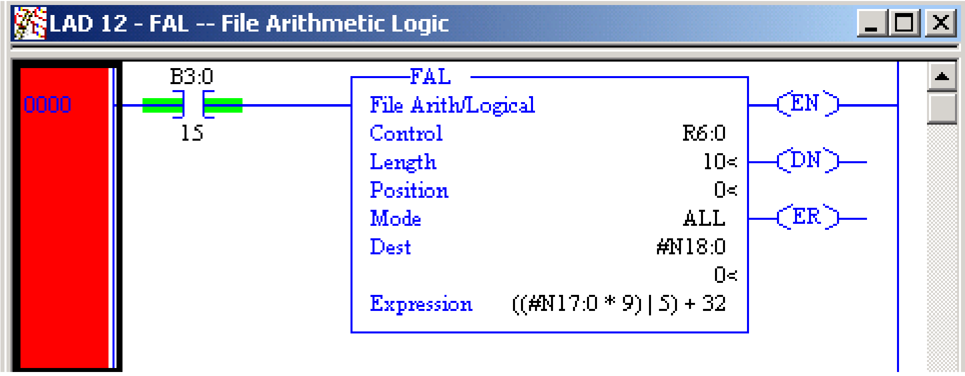
Name: DESTINATIN

Description: destination for calculations

Elements: 10

Last:

6) Now, let's go back to Ladder 12, and add an FAL statement as shown. (Note the # before N18:0 and N17:0. This means that for each operation, the FAL instruction will index through the position of each of these files) Without the # before the destination, all 10 calculations would write to the same element. Without the # before N17:0, all calculations would be performed from the same source element.



LAD 12 - FAL -- File Arithmetic Logic

0000

B3:0

15

FAL

File Arith/Logical

Control R6:0

Length 10<

Position 0<

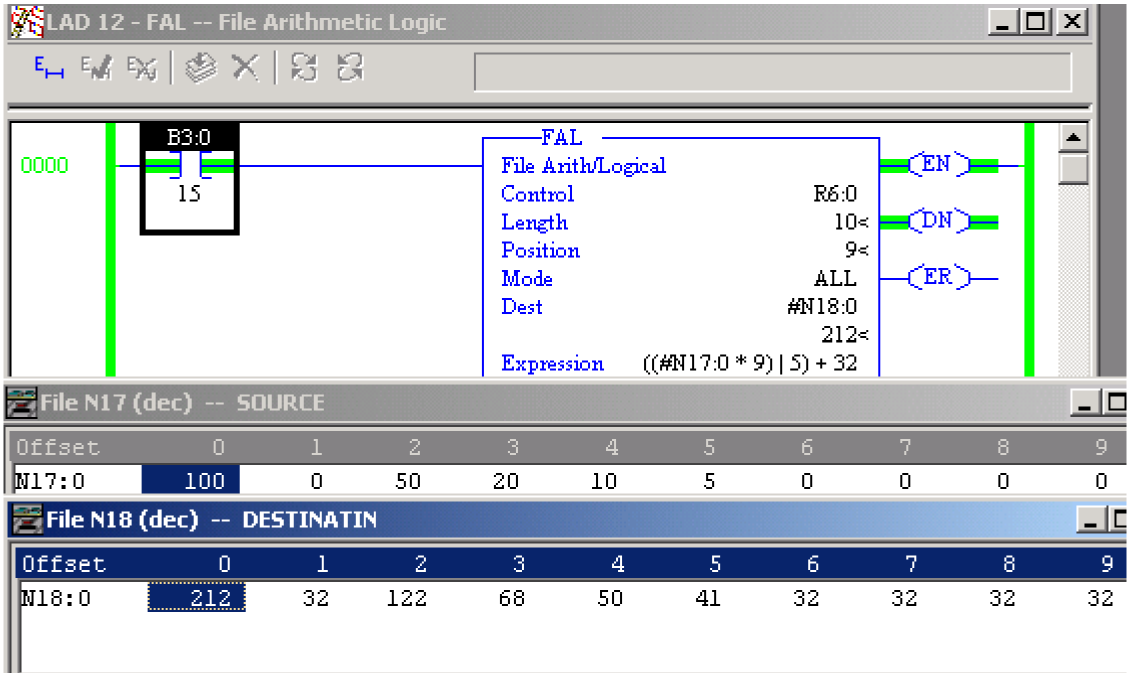
Mode ALL

Dest #N18:0

0<

Expression ((#N17:0 * 9) | 5) + 32

7) Download your work to the processor and go on line. Manually populate the N17 data file with values. Toggle the internal bit to enable the FAL instruction. Once you open the N18 file, you will see that all the values you entered in N17 are now converted into Fahrenheit. You can set both data files side by side and watch the conversion take place.



LAD 12 - FAL -- File Arithmetic Logic

0000 B3:0 15

FAL	
File Arith/Logical	
Control	R6:0
Length	10<
Position	9<
Mode	ALL
Dest	#N18:0
	212<
Expression	((#N17:0 * 9) | 5) + 32

File N17 (dec) -- SOURCE

Offset	0	1	2	3	4	5	6	7	8	9
N17:0	100	0	50	20	10	5	0	0	0	0

File N18 (dec) -- DESTINATIN

Offset	0	1	2	3	4	5	6	7	8	9
N18:0	212	32	122	68	50	41	32	32	32	32

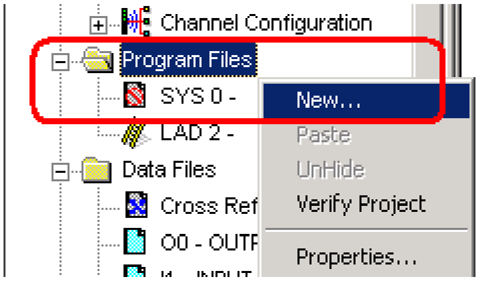
Special Routines

There are three types of special routines we will cover in this section: The Fault Routine, the PII (Processor Input Interrupt), and the STI (Selectable Timed Interrupt)

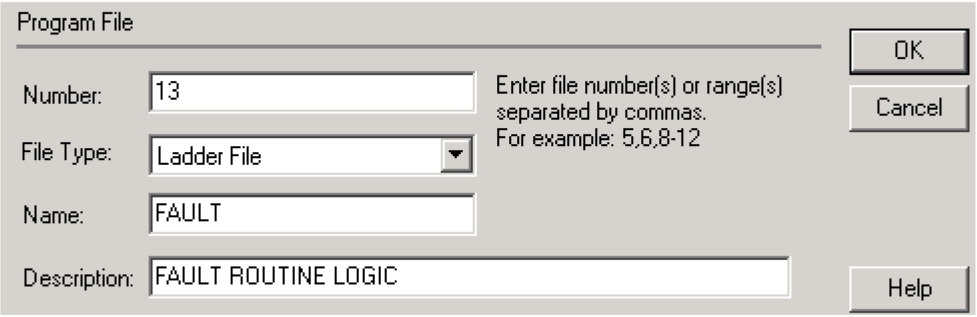
The Fault Routine

Fault routines can be used to execute a set of instructions (ladder logic) when a processor fault occurs. This logic might be for the purpose of clearing data tables, setting alarms, or even attempt to recover from the fault. This fault routine is executed before the system shuts down. In this example, we will move the value of "13" into memory location N7:0 when the processor faults by using a fault routine.

1) To create a fault routine, right click the program file folder while in program mode or off line to create a new file.



2) We will make the file number 13, and name the routine "FAULT" as shown. Press OK when finished.


Program File

Number: 13

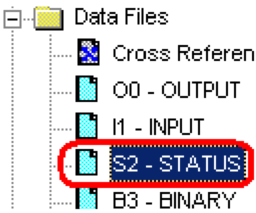
Enter file number(s) or range(s) separated by commas. For example: 5,6,8-12

File Type: Ladder File

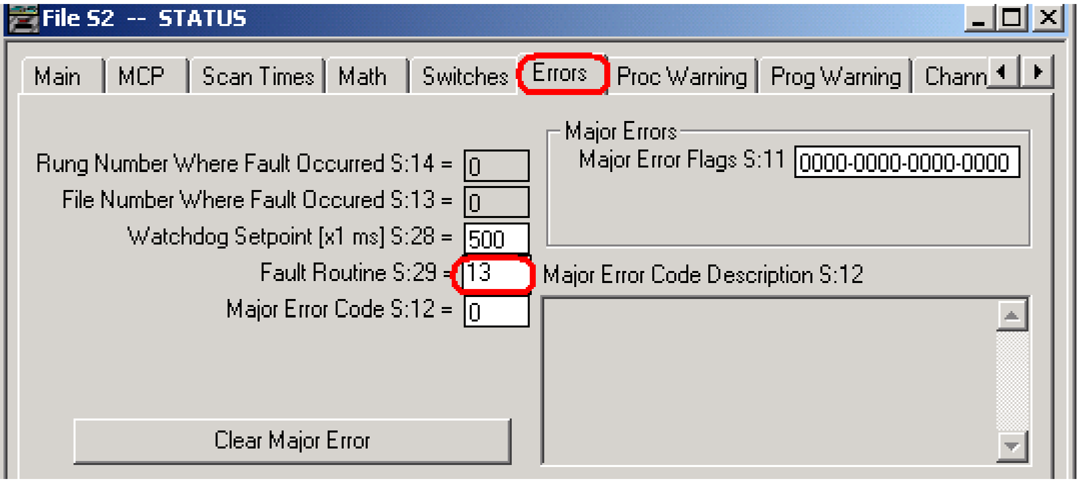
Name: FAULT

Description: FAULT ROUTINE LOGIC

3) To set up the processor to execute this routine on a major fault, open the S2 Data file.



4) Click the "Error" Tab on the processor status window, and specify the fault routine as file #13 as shown.



Rung Number Where Fault Occurred S:14 = 0

File Number Where Fault Occured S:13 = 0

Watchdog Setpoint [x1 ms] S:28 = 500

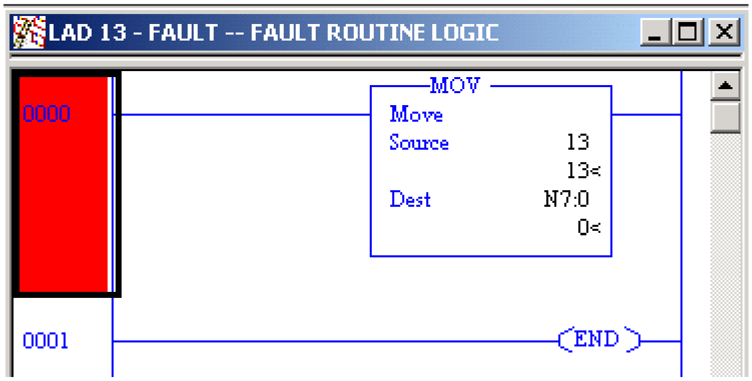
Fault Routine S:29 = 13

Major Error Code S:12 = 0

Major Error Flags S:11 0000-0000-0000-0000

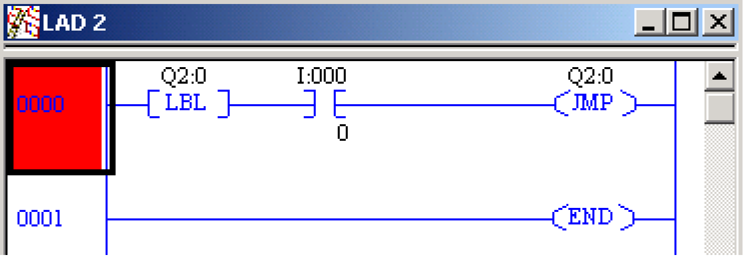
Major Error Code Description S:12

5) Close out of the processor status window, and enter the sample fault logic in ladder 13 as shown below:

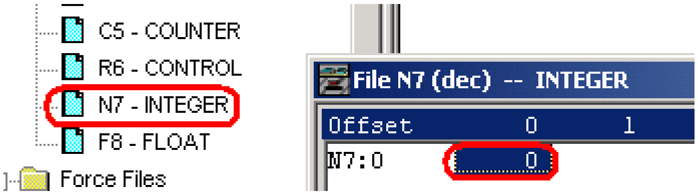


```
0000                                  MOV
                                      Move
                                      Source        13
                                                    13<
                                      Dest        N7:0
                                                    0<
0001                                              (END)
```

6) Now we need to write logic that will fault the processor to test this routine. Enter the following logic into ladder 2. This will cause the processor to fault when a switch is energized. This logic will create an infinite loop that will cause the processor to fault on a watchdog time-out.

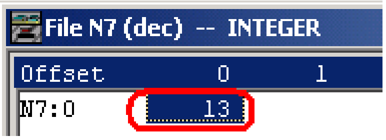


7) Open the N7 Data file, and you will see that the value of N7:0 is currently at 0 with the processor running (the fault has not occurred yet)



Offset	0	1
N7:0	0	

8) If you were off line, download your work and go to run mode. If you were in program mode, just place the processor in “Run” or “Remote Run mode”. Energize switch 0, and you will see that a processor fault has occurred. (The PROC light is now flashing on your processor) You will also notice the value of 13 was moved into N7:0 as well because the fault routine was executed.

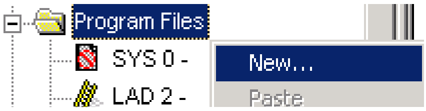


Offset	0	1
N7:0	13	

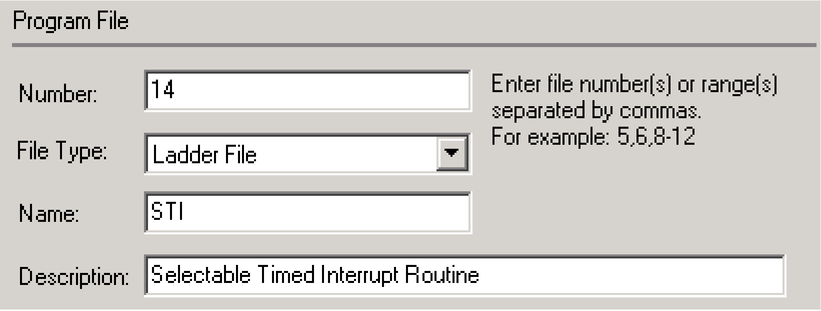
STI (Selectable Timed Interrupt)

The Selectable Timed Interrupt routine will execute at exact interval (in milliseconds). When it's time to execute the STI routine, the processor is interrupted from it's normal scan cycle to execute the STI routine. A common instruction you will usually find in the STI routine is the PID (Proportional Integral Derivative). In this example, we will prove the use of the STI by setting up a routine that will add the value of 1 to N7:0 every half second. When the preset is raised to 1000 ms, you will see that the value of N7:0 will then only increment once a second.

1) Right click on "Program Files" in the project tree while the processor is in program mode, or if you are off line. Select "New" (Note: If you are off line, you will need to download your changes to the processor before they become active.)



2) For this example, we'll make this file #14, and name it "STI" as shown. Press OK when finished.


Program File

Number: 14

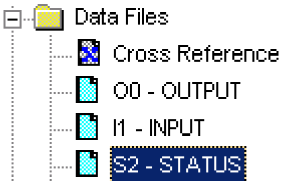
File Type: Ladder File

Name: STI

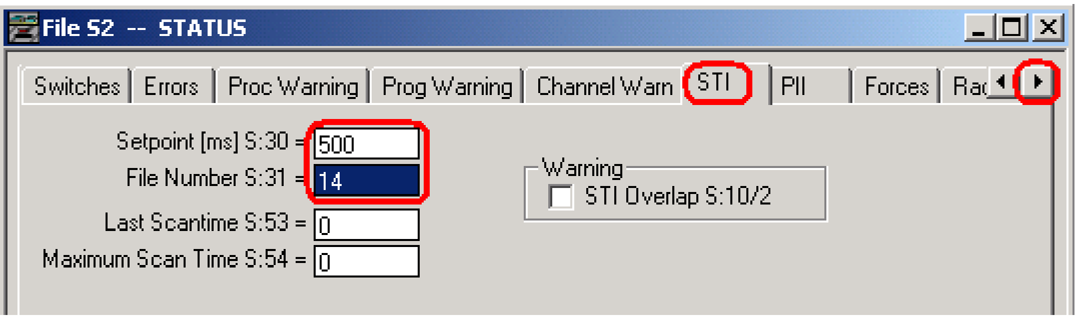
Description: Selectable Timed Interrupt Routine

Enter file number(s) or range(s) separated by commas.
For example: 5,6,8-12

3) Next, we need to tell the processor that ladder 14 is our STI, and we are going to execute ladder 14 at 500ms intervals. Open the S2 (Status) file in the project tree.



4) Next Scroll to the right on the processor status tabs until you find a tab labeled “STI”. Schedule ladder 14 to execute every 500ms as shown.



Setpoint [ms] S:30 = 500

File Number S:31 = 14

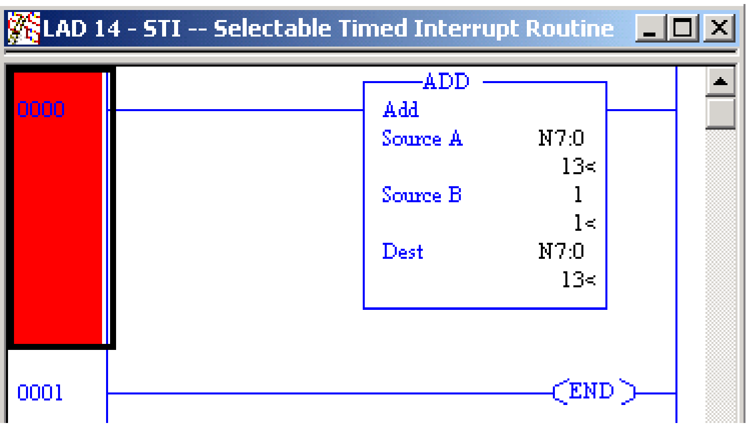
Last Scantime S:53 = 0

Maximum Scan Time S:54 = 0

Warning

□ STI Overlap S:10/2

5) Now we need to add some logic to ladder 14 to prove to ourselves the STI is executing properly. Open Ladder 14 from the project tree, and add logic as shown that will increment the value of N7:0. Remember this ladder is executing every 500ms, so we should see the value of N7:0 increment twice a second (one for each time the routine is executed).



```
0000                      ADD
                          Add
                          Source A       N7:0
                                          13<
                          Source B          1
                                           1<
                          Dest           N7:0
                                          13<

0001                                    END
```

6) If you were off line, download your work, and go to “Run” Mode. If you were in Program Mode, simply accept your edits and go to “Run” or “Remote Run” Mode.

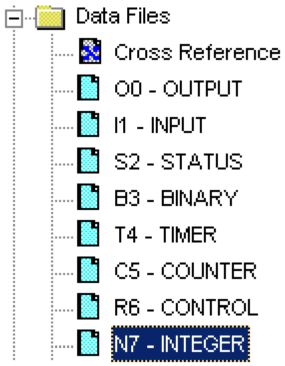
Warnings:

1) Outputs remain in their last state when you leave an STI routine, so be sure you are leaving your outputs in a safe state

2) Unpredictable operation could result with 5/40(L) and 5/60(L) processors if they are Series A Revision B or earlier when utilizing remote block transfers. Do not use remote block transfers within an STI if you have one of these early processors.

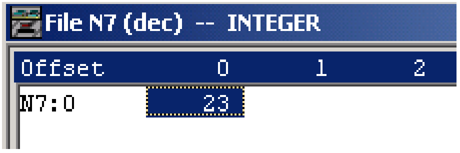
3) STI's increase your scan time. Be sure you do not exceed the watchdog setting.

7) Open the N7-Integer Data File from the Project Tree.



- Data Files
  - Cross Reference
  - O0 - OUTPUT
  - I1 - INPUT
  - S2 - STATUS
  - B3 - BINARY
  - T4 - TIMER
  - C5 - COUNTER
  - R6 - CONTROL
  - N7 - INTEGER

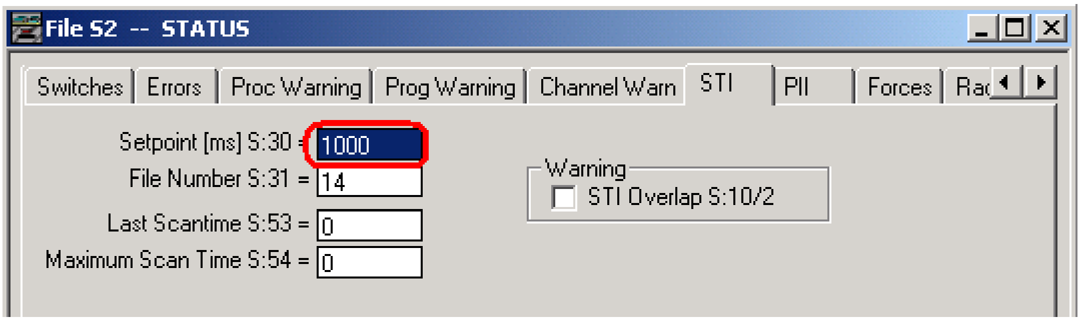
8) Observe the value of N7:0. You will see that it increments every 500ms.



File N7 (dec) -- INTEGER

Offset	0	1	2
N7:0	23		

9) Now go back to the S2 Data file (on the STI tab), and increase the set point to 1000ms.



File S2 -- STATUS

Setpoint [ms] S:30 = 1000

File Number S:31 = 14

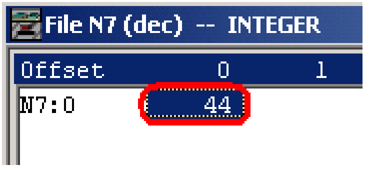
Last Scantime S:53 = 0

Maximum Scan Time S:54 = 0

Warning

□ STI Overlap S:10/2

10) Now go back to the N7 Data file, and you will see the value of N7:0 only increments once each second now.



File N7 (dec) -- INTEGER

Offset	0	1
N7:0	44	

PII (Processor Input Interrupt)

The PII is used to execute a certain routine when certain input conditions are met on a discrete input module on the LOCAL chassis with an Enhanced PLC-5 processor. This is considered an event-driven interrupt. Every 100 microseconds, the processor checks for transitions on this module. The PII cannot be used on a remote chassis, or an extended local chassis. The PII routine has a higher priority than STI routines (Selectable Timed Interrupt)

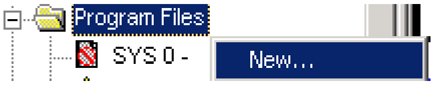
Warnings:

1) Outputs remain in their last state when you leave a PII routine. Be sure you are leaving your outputs in a safe state.
2) PII changes in the status file may not take effect until the processor goes from program mode to run mode. Unpredictable operation could occur before this happens.
3) Be sure you don't exceed the watchdog set point.
4) Do not use 2-slot addressing in the local chassis when using PII's
5) IIN (Immediate Input) and IOT (Immediate Output) instructions should not be looking at the same slot as the PII.
6) The PII may function unpredictably with noisy input signals.
7) Do not put a module requiring block transfers (such as an Analog Module) next to the module being monitored for PII, and if possible don't even put these modules in the same chassis as the module being monitored for PII.
8) Limit the logic you place in the PII. Do not add logic to the PII that can easily be placed in another routine.

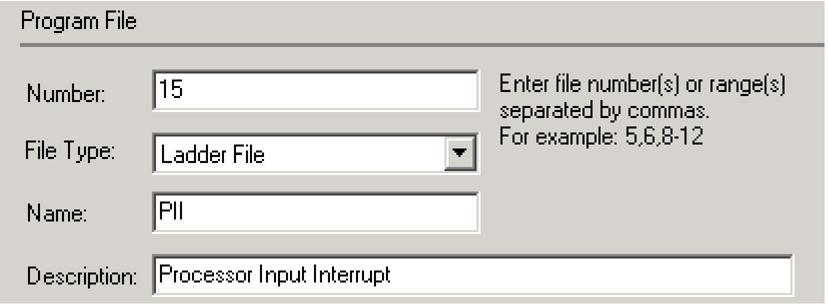
PII's can be used to shut down equipment, or re-initialize data files (such as a recipe) when an event (or certain number of events) occur on the input module that is being monitored.

For this example, we will energize bit B3/0 for every 5 true to false transitions we see on the simulator module in Group 0 of your local chassis. Once the bit is energized, we'll know the PII executed, then we will manually toggle it back off. This will prove the operation of the PII routine.

1) First, lets create the PII routine. You must be in program mode or off line to do this. Right click the program file folder in the project tree, and select “New”.



2) This will be file #15, and we will name it PII as shown. Press OK when finished.



Program File

Number: 15

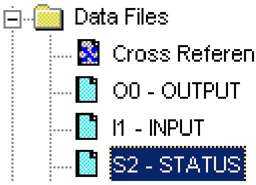
File Type: Ladder File

Name: PII

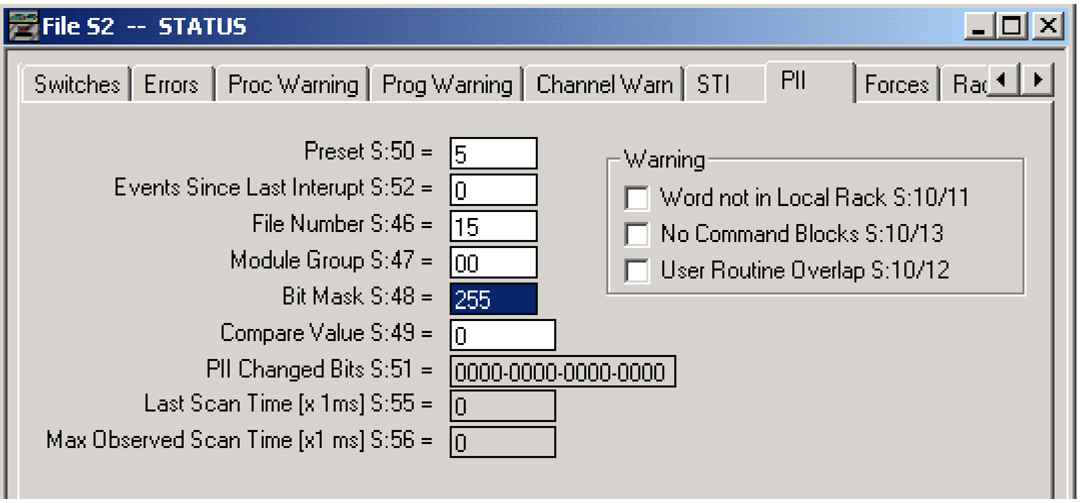
Description: Processor Input Interrupt

Enter file number(s) or range(s) separated by commas.
For example: 5,6,8-12

3) Next, we need to configure the status file, so the processor knows to look at this module for events. Open the S2 Data file.



4) Now scroll over to the PII tab. Let's discuss some of the options on the PII tab:



Preset S:50 = 5

Events Since Last Interupt S:52 = 0

File Number S:46 = 15

Module Group S:47 = 00

Bit Mask S:48 = 255

Compare Value S:49 = 0

PII Changed Bits S:51 = 0000-0000-0000-0000

Last Scan Time [x 1ms] S:55 = 0

Max Observed Scan Time [x1 ms] S:56 = 0

Warning

□ Word not in Local Rack S:10/11

□ No Command Blocks S:10/13

□ User Routine Overlap S:10/12

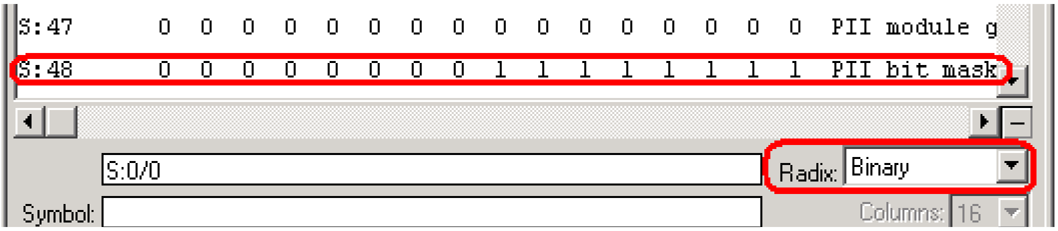
5) The preset (S:50) is the number of events that are to happen before the PII is executed.

6) The events since the last interrupt (S:52) is the processor's internal counter. This will update as events occur on the input module.

7) The file number (S:46) is the ladder number of our PII routine. In this case, it will be 15.

8) The module group (S:47) is the group number at which the processor will look for transitions. The address for switches on our SIM card is I:000 (Rack 00, Group 0), so we will enter 00 as the group number.

9) The bit mask (S:48) indicates which bits on the module the processor will look for transitions. To truly understand this mask, let's change the RADIX to BINARY, and scroll down to S:48. You will see that when we are looking at bits 0-7 (all being the binary value of 1), this translates into a decimal value of 255.



```
S:47     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  PII module g
S:48     0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1  PII bit mask
```

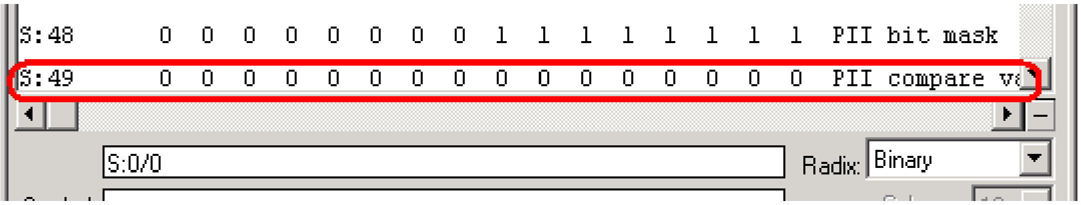
S:0/0

Radix: Binary

Symbol:

Columns: 16

10) The Compare value (S:49) indicates what transition will cause the event trigger. For each bit, enter 1 for false to true transitions, and 0 for true to false transitions. We are going to leave them all at 0, so our trigger will be on the true to false transition.



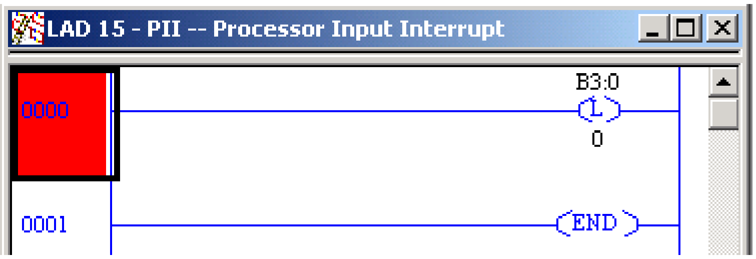
```
S:48     0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1  PII bit mask
S:49     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  PII compare va
```

S:0/0

Radix: Binary

11) Change your RADIX back to STRUCTURED.

12) Now let's open Ladder 15, and add our latch instruction to prove the PII is executing this routine. Download your work and go to run mode, or just accept your changes, and go to run mode if you were previously in program mode.



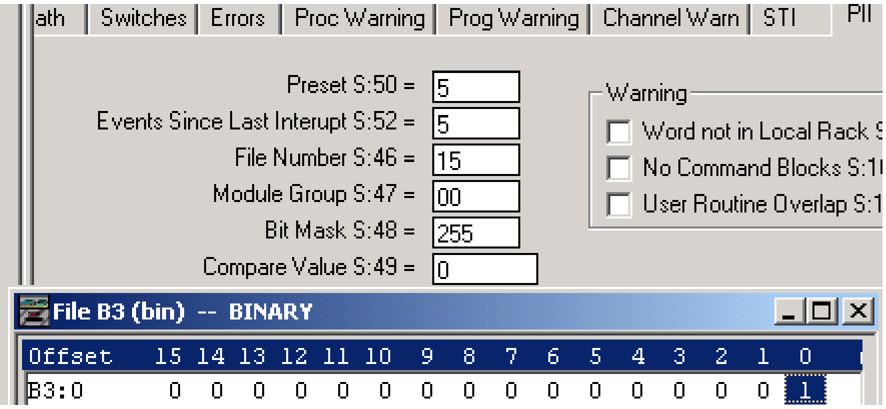
LAD 15 - PII -- Processor Input Interrupt

```
0000 ----------------------------------------- B3:0
                                               -(L)-
                                                 0
0001 ----------------------------------------- (END)
```

13) Energize all 8 switches on the SIM card.

14) Now, let's open the status file (on the PII tab, and open the B3 data file as well). Keep both windows visible.

15) When you open a switch, the processor will see that an event (or several) have occurred. It's difficult to make a clean break on the switches on the SIM module. When you see that 5 events have occurred, B3/0 will go high because the PII routine was executed. If you want to try this again, manually reset B3/0 for now. When the PII is implemented in real situations, other logic would be executed to take appropriate action on the event. The PII is commonly associated with the IOT instruction (Immediate Output) that will update the output immediately (before the normal output update during the regular scan cycle)



Preset S:50 = 5
Events Since Last Interupt S:52 = 5
File Number S:46 = 15
Module Group S:47 = 00
Bit Mask S:48 = 255
Compare Value S:49 = 0

Warning
□ Word not in Local Rack S
□ No Command Blocks S:1
□ User Routine Overlap S:1

File B3 (bin) -- BINARY

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PID Demonstration—PLC-5

Terminology

CV – Control Variable – This is the output (control) from the PID loop. The control variable can be used to control a valve or heating bands. The PID loop will continuously read the input (process variable), and decide where to set the control variable to achieve the set point.

PV – Process Variable – This is the input (status) of the process. The process variable can read a tank level or a temperature. The PID loop continuously reads this value to determine how much output to provide to the process to achieve the set point.

SP – Set Point – The value of the process variable we desire to achieve. The set point can be described in terms of a temperature we want to maintain, a tank level we desire, a particular flow rate, etc.

ERROR – The difference between the current value of the set point and the process variable. (In other words, the difference between where we are and where we should be)

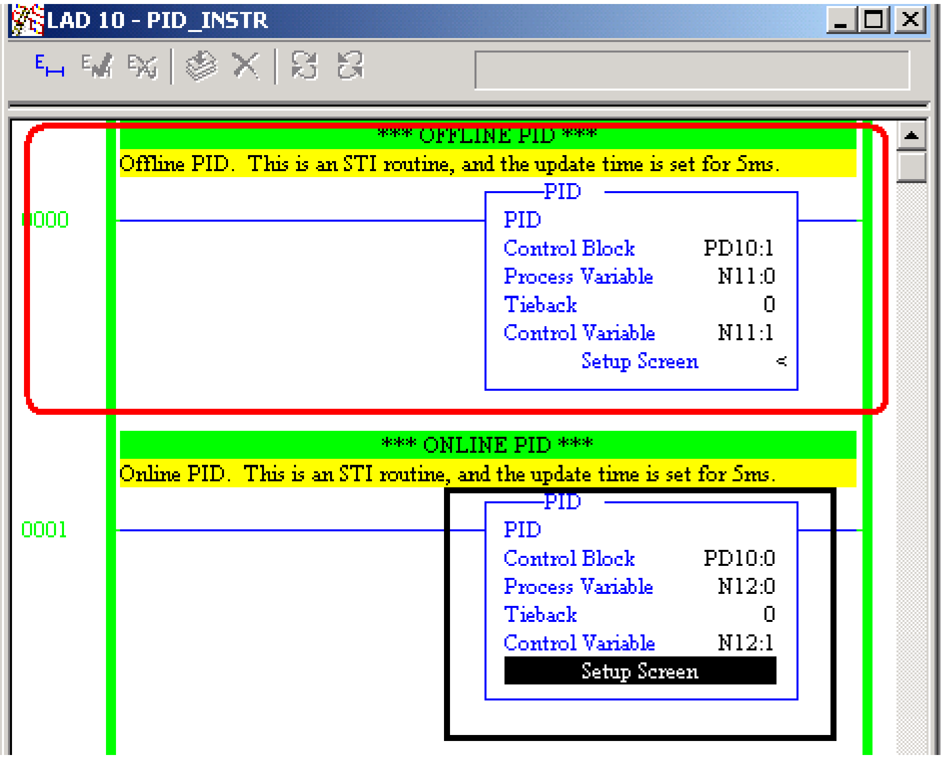
Proportional – Output that is proportional to the amount of error. For example: as the process variable approaches the set point the output (CV) will decrease (or in some cases increase) proportionally.

Integral – Repeats the proportional influence over time. For example: The longer we stay below the set point, the more output is provided (or less in some cases). Integral is based on time.

Derivative – Output influence that is based on the rate of change of error. For example, if we have a huge steam demand, a heating unit may start to cool down rapidly. At first, the error is not enough to provide a large output based on proportional influence, and not enough time has passed for integral influence to provide enough output to quickly pull the process back to the set point. Derivative is not used in every process. For many processes, just the proportional and integral alone is enough to control a loop.

Tieback – A value used to implement “bumpless transfer” when the loop is in manual mode. This helps to ensure a more stable output when the loop is placed into manual mode.

1) In Ladder 10, you will find an OFFLINE PID instruction. This instruction is not currently controlling any process. We are going to use this instruction simply for the purpose of seeing how the proportional controller reacts to error.



LAD 10 - PID_INSTR

*** OFFLINE PID ***

Offline PID. This is an STI routine, and the update time is set for 5ms.

0000

PID
PID
Control Block PD10:1
Process Variable N11:0
Tieback 0
Control Variable N11:1
Setup Screen

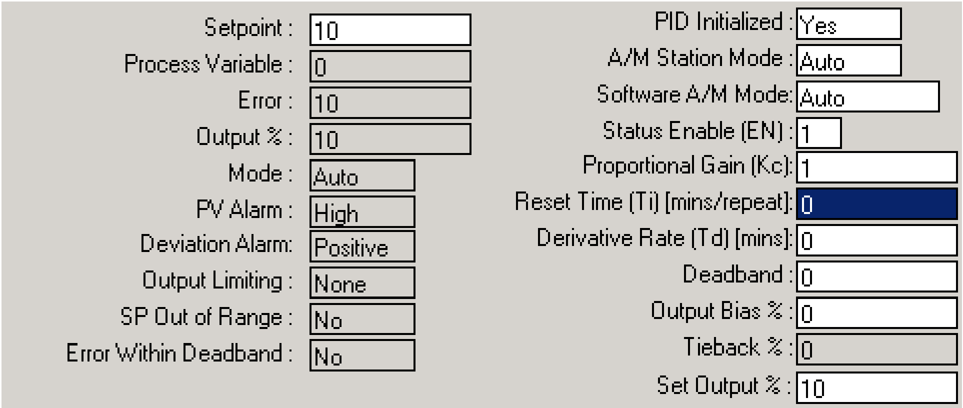
*** ONLINE PID ***

Online PID. This is an STI routine, and the update time is set for 5ms.

0001

PID
PID
Control Block PD10:0
Process Variable N12:0
Tieback 0
Control Variable N12:1
Setup Screen

2) Click "Setup Screen" within the PID instruction, and you will see various parameters to control the PID instruction. PV is the process variable. This is the value returned from a sensor. We will also be discussing the CV (control variable) later in this lesson. This is the output for the PID loop. Set up the status parameters as follows:



Setpoint :	10	PID Initialized :	Yes
Process Variable :	0	A/M Station Mode :	Auto
Error :	10	Software A/M Mode:	Auto
Output % :	10	Status Enable (EN) :	1
Mode :	Auto	Proportional Gain (Kc):	1
PV Alarm :	High	Reset Time (Ti) [mins/repeat]:	0
Deviation Alarm:	Positive	Derivative Rate (Td) [mins]:	0
Output Limiting :	None	Deadband :	0
SP Out of Range :	No	Output Bias % :	0
Error Within Deadband :	No	Tieback % :	0
		Set Output % :	10

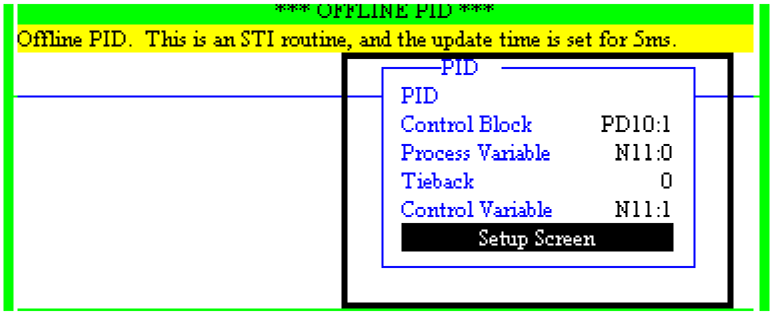
3) You will notice that the Controller Gain (Kc) is going to be our proportional gain. Be sure this gain is set for the value of 1 for our first experiment. Ti and Td are your integral and derivative timing values. We will work with these later.

4) Next we are going to change the SP to the value of 10. When you changed the Set point to the value of 10, how much output did the controller provide? What is the error?

5) If we change the SP to the value of 30, how much output would you expect from the controller? If the PV is still at 0, what is the error going to be?

6) Try changing the SP to 30 and see if you get the results you would expect.

7) Now lets imagine for a moment that the output we are providing is starting to cause things to happen in the real world.... Such as a flow control valve that is starting to open. We are continuously reading the flow rate from this system. The sensor that is measuring the flow rate is calibrated in such a way that if we get a value of 4095 (20mA), then our flow rate is 100%. The value of 0 returned from the sensor will reflect a flow rate of 0. We can find the conversion from amperage to raw data in the manual for the analog module. Look at your PID equation. What memory location represents the value being returned from the sensor?

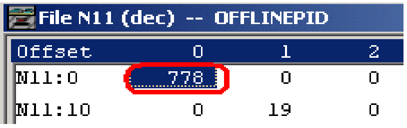


*** OFFLINE PID ***

Offline PID. This is an STI routine, and the update time is set for 5ms.

PID
PID
Control Block PD10:1
Process Variable N11:0
Tieback 0
Control Variable N11:1
Setup Screen

8) Remember that the value we are sending to the flow valve is called the Control Variable. The value being returned from a sensor which is measuring actual flow is called the Process Variable. The process variable is going to show up in memory location N11:0. Remember how our sensor is calibrated. If we see the value of 4095 at N11:0, then we know we have 100% flow. We are going to simulate feedback from a flow meter by entering the value 778 into N11:0. After entering the value of 778 at N11:0, go back to the Setup Screen for your PID instruction.



File N11 (dec) -- OFFLINEPID

Offset	0	1	2
N11:0	778	0	0
N11:10	0	19	0

9) With your Set point still at 30:

    1. What is the PV reading?

    2. What is the CV reading?

    3. What is the ERROR?

    4. Note that 778 is 19% of 16383.

10) Increase N11:0 to the value of 942 (23%). What happens to the output as the Process Variable begins to reach the set point?

11) Now lets see how a change in Kc gain causes the controller to react. Reset N11:0 back to 0, go to the setup screen for the PID loop in Ladder 9 and change the Kc Gain to the value of 2. If you were to change the set point to the value of 10, how much output would you expect from the controller?

12) Change the SP to 10. Did you get the value you expected to see?

13) What output would you expect at the CV if you change the SP to 30?

14) Change the SP to 30. Did you get the output you expected?

15) Change N11:0 to 942 (23%). What output would you expect to see with a SP of 30?

16) Go back to the Setup Screen for the OFFLINE PID instruction in Ladder 10. Was your answer correct?

17) Now you understand how the proportional controller alone works. The controller provides an output that is proportional to the error. Think about how the proportional controller alone works. Under normal circumstances, can the proportional controller alone bring the process variable up to the set point?

18) Why or Why Not?

19) Think about a heating process:

When you start the heating process, the controller will provide a very large output. As you approach the set point, the output will decrease, until the output is zero at the set point. With an output of zero, you are not putting any heat into the system. Due to ambient and load losses, the system will begin to cool. Eventually, at some point below the set point, the amount of heat the controller is adding to the system will be equal to the amount of heat being removed from the system. At this point your system has reached steady state. The error between the steady state temperature and the set point is called the OFFSET.

If the load is increased, then more heat is being drawn out of the system. The controller is no longer providing sufficient output to maintain the previous offset. A greater error must be present in order to increase the amount of heat the controller is adding to the system.

Now verify your answer to question 10. We will demonstrate this effect in the next example.

1--SETPOINT----------------PV=SP (CV=0)

2____PV little below SP (CV =10)

3____PV much below SP (CV=20)

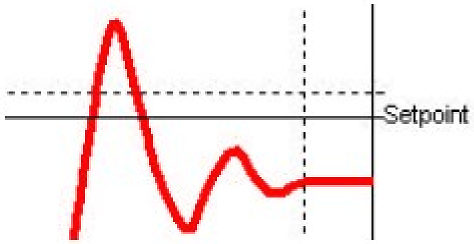
Position 1) If the PV was at the SP, then the controller is not adding any power to compensate for ambient and load losses. Temperature cannot be sustained.

Position 2) If our PV was at this point, we have an error of 10, and therefore the processor is calling for 10% power. Here we can reach steady state if the losses and loads combined are not taking out more heat energy than we are putting into the system. If the load increased, we could not maintain this temperature since our output can only be 10% at this position.

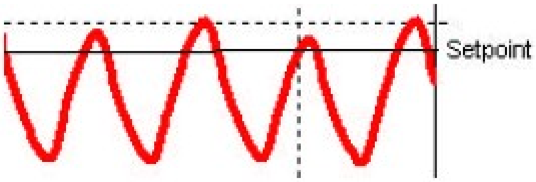
Position 3) If we was at position 2 and our load began to increase, we are taking more heat out of the system than what we are putting in. The temperature begins to decrease. This causes an increase in the error, and the controller will increase power. The system will reach steady state at a lower temperature. This will be the temperature at which the controller is providing enough output to make up for the increased load.

20) If the Kc is increased, then the offset is going to be decreased because we can have the same output with a smaller error. However, if Kc is increased too high, the controller will become unstable and start to oscillate. Allen Bradley states in the Instruction Set Reference Manual that a good rule of thumb used to set Kc is half the value which causes oscillations. This will vary depending on the process.

21) In this example Kc is set okay. The PV overshot the SP, but did eventually reach steady state. Since this is is an example of proportional control only. Steady state is actually being reached at a lower temperature than the set point. The reason is because the temperature at which we reached steady state provided the exact error we need for the proportional controller to generate enough output to make up for losses and load. If Kc is increased, then the error will be decreased, and the steady state temperature will be closer to SP.



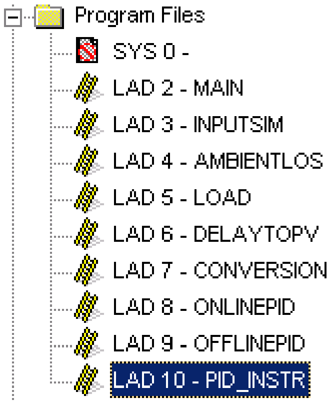
22) In this example, we increased Kc too much. This caused the controller to become unstable and start to oscillate. The minimum value for Kc to cause this type of oscillation is important for calibration. In many cases, once you find the value of Kc that causes the controller to oscillate, set Kc to half this value. The amount of time required for one complete oscillation is called the "natural period" of a sine wave. This natural period will be useful when setting the integral and derivative later on in this lesson.



23) In the next lesson, we are going to simulate a real world process. During this lesson you are going to determine what is the maximum value we can use for Kc without the controller becoming unstable.

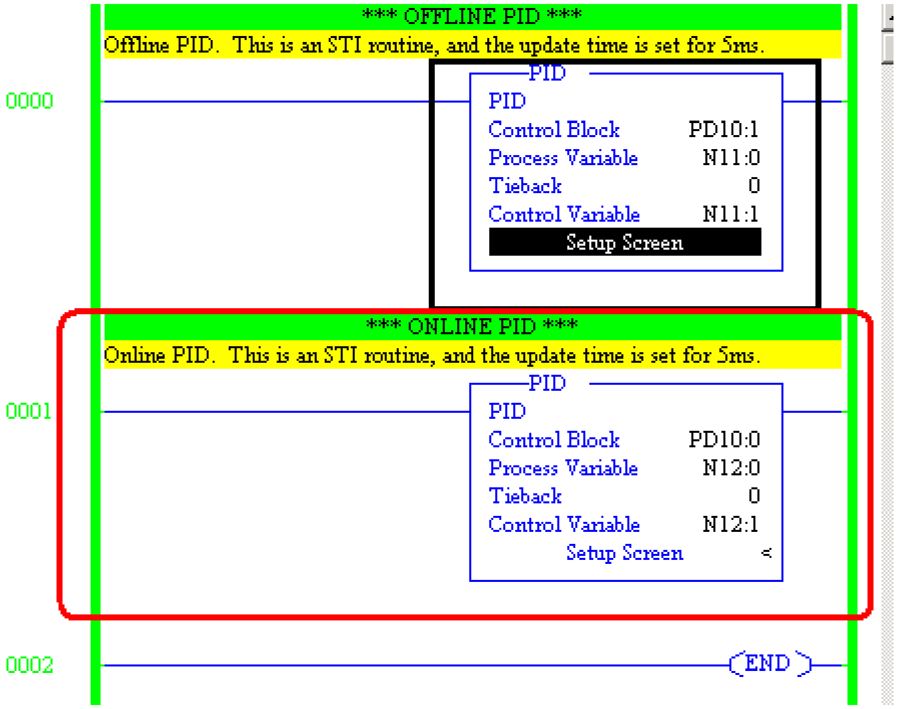
Proportional Gain Worksheet

24) To start this exercise, double-click ladder 10 in the project tree of RSLogix 5. On the right hand side of your screen, you will see the PID controller for this lesson.



Program Files
- SYS 0 -
- LAD 2 - MAIN
- LAD 3 - INPUTSIM
- LAD 4 - AMBIENTLOS
- LAD 5 - LOAD
- LAD 6 - DELAYTOPV
- LAD 7 - CONVERSION
- LAD 8 - ONLINEPID
- LAD 9 - OFFLINEPID
- LAD 10 - PID_INSTR

25) Notice the ONLINE PID instruction in your ladder diagram:



*** OFFLINE PID ***

Offline PID. This is an STI routine, and the update time is set for 5ms.

0000

PID	
Control Block	PD10:1
Process Variable	N11:0
Tieback	0
Control Variable	N11:1
Setup Screen	

*** ONLINE PID ***

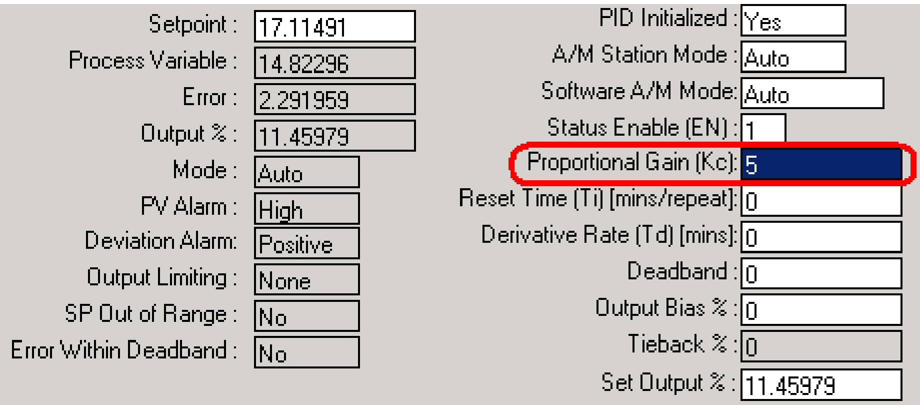
Online PID. This is an STI routine, and the update time is set for 5ms.

0001

PID	
Control Block	PD10:0
Process Variable	N12:0
Tieback	0
Control Variable	N12:1
Setup Screen	<

0002

26) Double-click "Setup Screen" within your PID instruction



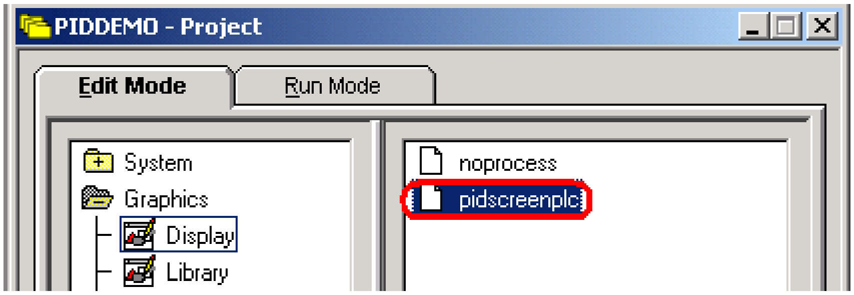
Setpoint : 17.11491
Process Variable : 14.82296
Error : 2.291959
Output % : 11.45979
Mode : Auto
PV Alarm : High
Deviation Alarm: Positive
Output Limiting : None
SP Out of Range : No
Error Within Deadband : No

PID Initialized : Yes
A/M Station Mode : Auto
Software A/M Mode: Auto
Status Enable (EN) : 1
Proportional Gain (Kc): 5
Reset Time (Ti) [mins/repeat]: 0
Derivative Rate (Td) [mins]: 0
Deadband : 0
Output Bias % : 0
Tieback % : 0
Set Output % : 11.45979

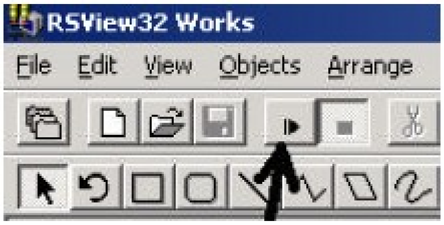
27) Set the Controller Gain (Kc) to 8.0

28) Next, follow your instructor to open a project in RSView. RSView will be the Graphical User Interface (GUI) for a heating process. The program in your PLC-5 will simulate the process. You can find RSView by clicking Start| Programs|Rockwell Software|RSView32|RSView 32 Works. Once RSView is open, Click File, and then open the project called piddemo.rsv

29) Next, Double click on the + next to Graphics. The single click display. A list of the screens available for this project show up on the right hand side. Double click the display called pidscreensplc.



30) Click the Start button on your tool bar.



31) Now we are set up for the next exercise. In this next exercise you are going to adjust the proportional gain (Kc) to find out what value causes the loop to become unstable. Use the following chart: For each row, go to RSLogix and enter the Kc value into the PID controller in Ladder 8. Come back to RSView, With your load at 0, change the load on the system to 100%. Watch the response. Circle whether the controller is still stable after the transition, or whether it has gone unstable. Then change your load from 100% to 0% and watch the graph in RSView. Record the stability, then move to the next row.

Proportional Gain Kc	Load transition 0 to 100%	Load transition 100 to 0%
8.0	Stable | Unstable	Stable | Unstable
10.0	Stable | Unstable	Stable | Unstable
12.0	Stable | Unstable	Stable | Unstable
15.0	Stable | Unstable	Stable | Unstable

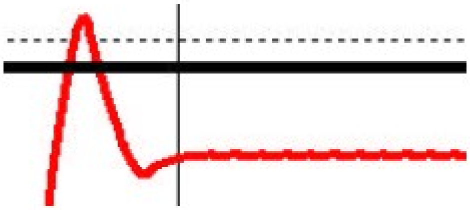
32) What value of Kc started causing instability during certain load transitions?

____

33) What is the natural period of the oscillations (in minutes)?

____

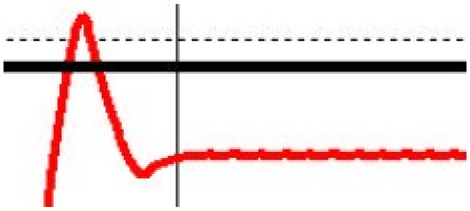
34) Your Set point is 700 degrees. Record the steady state temperature under 0 load with your new Kc. _______You will see that the steady state temperature is significantly below the set point. That is because of the error required to generate enough output to make up for ambient losses. Put your load at 100% and record the steady state temperature again. _________You will see that it is even lower. Because we are taking more heat out of the system, we must have greater error to generate more output.



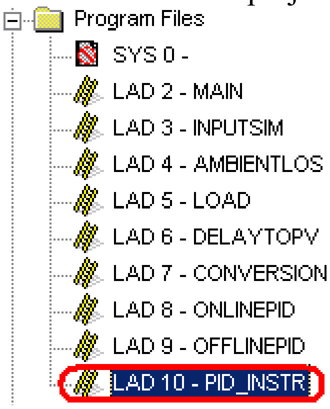
35) We will compensate for this offset in the next lesson

Integral Worksheet

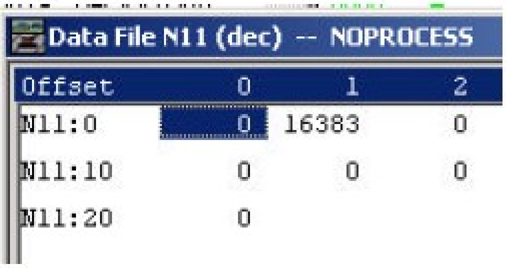
36) You noticed in the last lesson that by using the proportional control alone, we cannot achieve the desired set point, and sometimes cannot even get the PV close to the SP without the loop becoming unstable. We need another component of the controller that can increase the output to push the PV up to the SP. The integral controller controls proportional repeats per minute. Therefore, the longer the PV stays below the SP, the more output is generated. The closer the PV gets to the SP, the slower the output increases. Remember proportional control generates an output simply based on the amount of error. The integral output takes this one step further. If we still have an error, then we are going to continuously keep adding output ever so slightly until we reach the set point.



37) Ladder 10 has a PID instruction set up that will allow us to see the effect of the controller alone when it is not attached to a real world process. Lets bring up RSLogix and double click ladder 10 in the project tree.



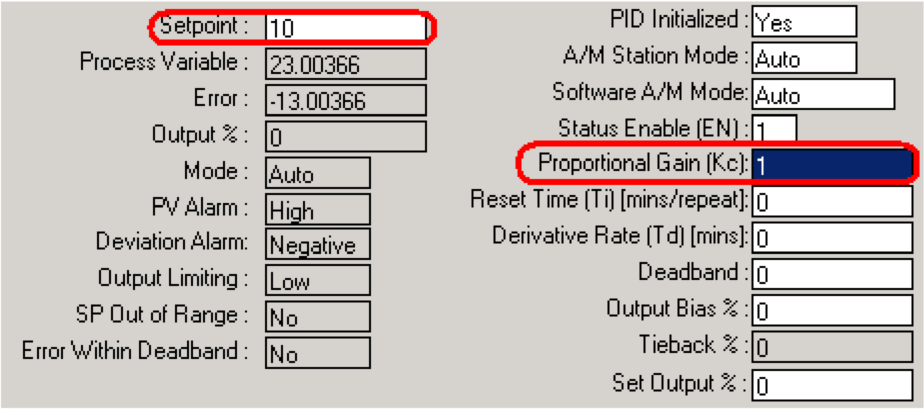
38) Before we start, lets set the Process Variable to 0 in the N11 Data File. If this were a real process, the feedback from the process would show up in N11:0. We will do this manually for this example.



Data File N11 (dec) -- NOPROCESS

Offset	0	1	2
N11:0	0	16383	0
N11:10	0	0	0
N11:20	0		

39) Click 'Setup Screen' on the PID instruction. Configure your PID instruction as follows:

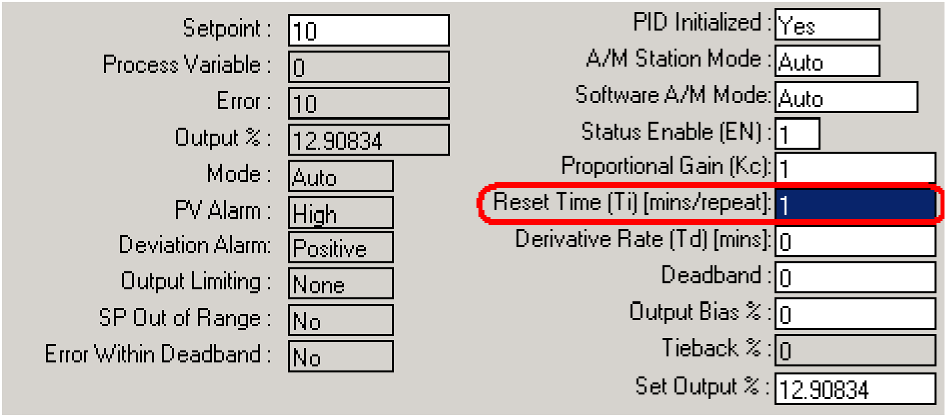


Setpoint : 10
Process Variable : 23.00366
Error : -13.00366
Output % : 0
Mode : Auto
PV Alarm : High
Deviation Alarm: Negative
Output Limiting : Low
SP Out of Range : No
Error Within Deadband : No

PID Initialized : Yes
A/M Station Mode : Auto
Software A/M Mode: Auto
Status Enable (EN) : 1
Proportional Gain (Kc): 1
Reset Time (Ti) [mins/repeat]: 0
Derivative Rate (Td) [mins]: 0
Deadband : 0
Output Bias % : 0
Tieback % : 0
Set Output % : 0

40) Recall that from a previous lesson, using the proportional control only, if our error is 10%, and the Controller gain is 1.0, then the output is going to be 10%. Here we have a condition where our PV is below the set point and is not increasing (just like in our online heating process once it reached steady state with proportional gain only).

41) Change your Reset Ti to the value of 1.0. If you make a mistake in entering this value (such as entering .1), enter 0 to clear the integral influence on the output, then enter 1.0 again.



Setpoint : 10
Process Variable : 0
Error : 10
Output % : 12.90834
Mode : Auto
PV Alarm : High
Deviation Alarm: Positive
Output Limiting : None
SP Out of Range : No
Error Within Deadband : No

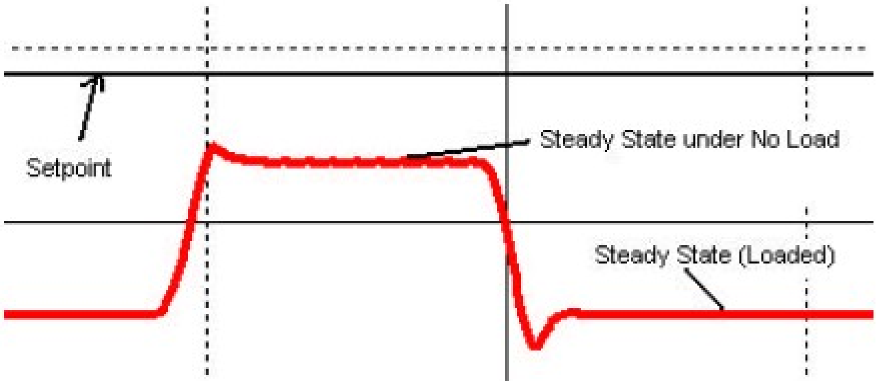
PID Initialized : Yes
A/M Station Mode : Auto
Software A/M Mode: Auto
Status Enable (EN) : 1
Proportional Gain (Kc): 1
Reset Time (Ti) [mins/repeat]: 1
Derivative Rate (Td) [mins]: 0
Deadband : 0
Output Bias % : 0
Tieback % : 0
Set Output % : 12.90834

42) Observe what is happening to the CV%. Since we entered Ti as repeats per minute, after 1 minute, the proportional gain will be repeated, and you will have an output value of 20. After 2 minutes you will have an output value of 30. If the PV starts to increase, the error will decrease, and the CV% will increase more slowly. Even if the PV is only 1% below the SP, the CV% will still be increasing (although very slowly by this time) until the PV is equal to the SP.

43) Change your Reset Ti to 0 to remove the integral influence, then set Ti to .5. You will see the Integral control is repeating the proportional gain every 30 seconds now which is twice as fast. If the value of Ti is set too high it could cause the system to become unstable. The Instruction Set Reference Manual recommends that the Integral value be set to the natural period of the oscillations that occurred when calibrating the Kc.

44) Now that you understand how integral control works, lets try this on your heating process.

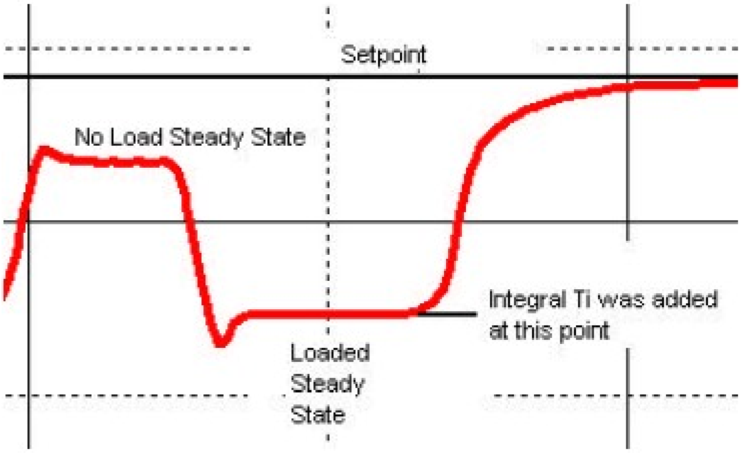
45) Go back to RSView, and for review, change your load to 0% and observe how the loop reacts. Then change your load to 100% and observe how the loop reacts. Also notice that the steady state process variable is well below the setpoint.



46) We are going to set the Ti value to the natural period of the Kc calibration that you recorded earlier. This should push the PV (temperature) up to the setpoint.

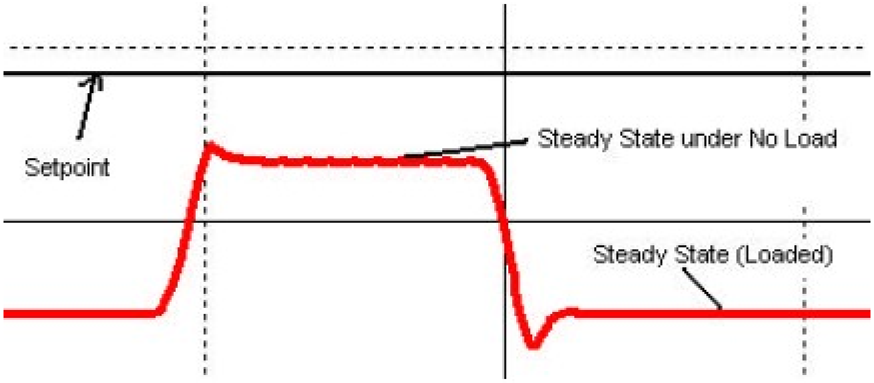
47) Go to RSLogix, and look at Ladder 10. Set the Ti value of the ONLINE PID instruction. Immediately go back to RSView and observe what effect Integral control is having on your system.

48) You can see that right away when Ti is added to the equasion, the output increases sharply due to the large error. The integral control continues to add output to the control variable at increasing slower rates until the PV is driven up to acheive the Setpoint. Now change your load between 0 and 100 to see how stable the loop is. You will see fluctuations immediately, however, no matter what the load, the Setpoint is acheived and the loop becomes stable within a reasonable period of time.



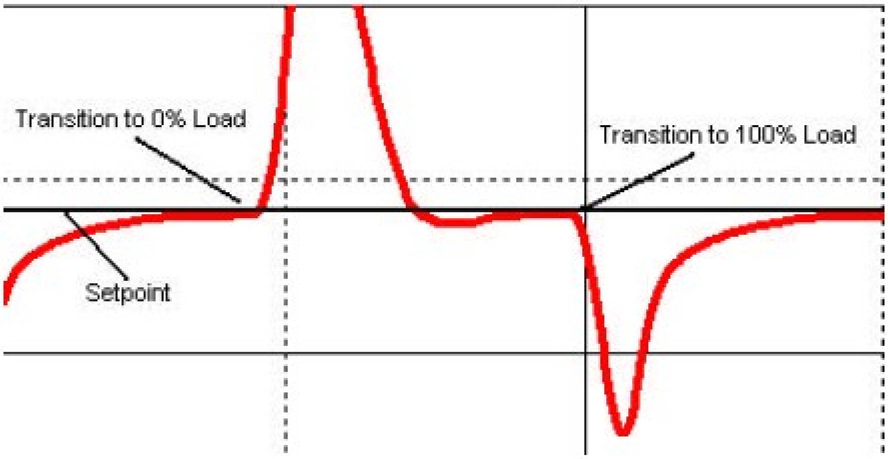
49) When removing the load, the PV may go out of range of the chart. The reason for that is that for 100% load, the control variable required is very high. When the load is removed, and is no longer removing heat from the system, the heat bands are still hot, and take some time to begin to cool.

50) Similarly, when heating up the system, the heat bands are cold, and require some amount of time to become hot enough to start adding heat to the system. For this example, the power level the PID controller is wanting to achieve is called the requested power. The amount of energy the heating bands are adding to the system is called the actual power. When there is a change in requested power, it takes some time for the actual power to acheive the requested power level.



51) Now Compare these 2 graphs. To the left is a graph of how the system reacted with proportional control only. Notice the steady state temperatures. The larger the load, the lower the temperature of the system. Even at no load, due to ambient losses, the steady state temperature was well below the setpoint.

52) Now that we have Ti added to our controller, you will see that we now achieve the SP.



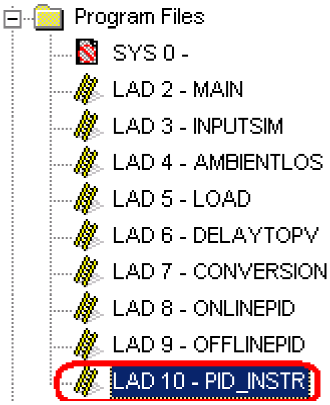
Derivative Worksheet

53) The proporional and integral control work well in maintaining a desired temperature. The last part of PID that we are going to discuss is the DERIVATIVE component. With derivative action, the controller output is proportional to the rate of change of the measurement or error. You do not always need derivative action, and on some processes, it should never be used. Since derivative is based on the rate of change of error, if you have a noisy PV, large amounts of gain can be generated causing the loop to become unstable. Derivative action on noisy loops can cause the valve to become 'jittery' and decrease the life of the valve. An example might be a flow process at a water treatment facility. The material flowing through the system might have various degrees of viscosity causing intermittent large changes in the PV. If the controller attempts to act on these changes, the loop could become unstable. Because of this effect, derivative action usually causes more harm than good for flow control processes and therefore should not be used.

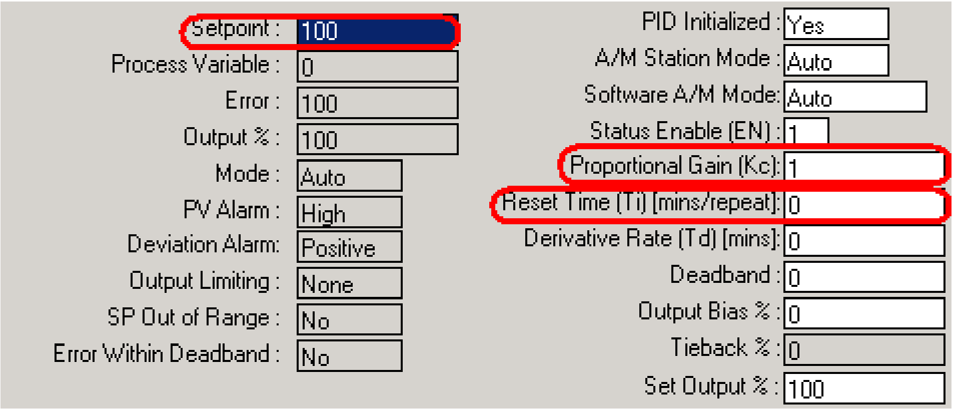
54) Imagine a heating process. Derivative action can be useful on a heating process by anticipating changes. If load is dramatically increased on a system, the PV is going to begin to decrease rapidly. The proportional and integral gain are not going to start adding large amounts of output to the CV until the error increases significantly. The derivative action will immediately detect that the PV is rapidly decreasing, and start to generate output immediately.

55) The Instruction Set Reference Manual states that a good rule of thumb to use when setting the derivativeTd, is one-eighth the value of Ti.

56) Because of the nature of the heating process, we have been using, Derivative action would not be significantly noticed. We are going to use our OFFLINE PID instruction in Ladder 10 to observe the effect of the derivative on a process.



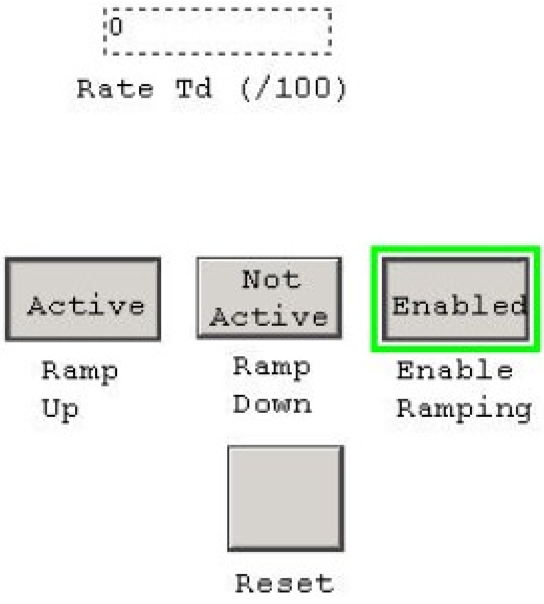
57) Let's go into the Setup Screen for our OFFLINE PID instruction, and configure the parameters as follows:



Setpoint : 100
Process Variable : 0
Error : 100
Output % : 100
Mode : Auto
PV Alarm : High
Deviation Alarm: Positive
Output Limiting : None
SP Out of Range : No
Error Within Deadband : No

PID Initialized : Yes
A/M Station Mode : Auto
Software A/M Mode: Auto
Status Enable (EN) : 1
Proportional Gain (Kc): 1
Reset Time (Ti) [mins/repeat]: 0
Derivative Rate (Td) [mins]: 0
Deadband : 0
Output Bias % : 0
Tieback % : 0
Set Output % : 100

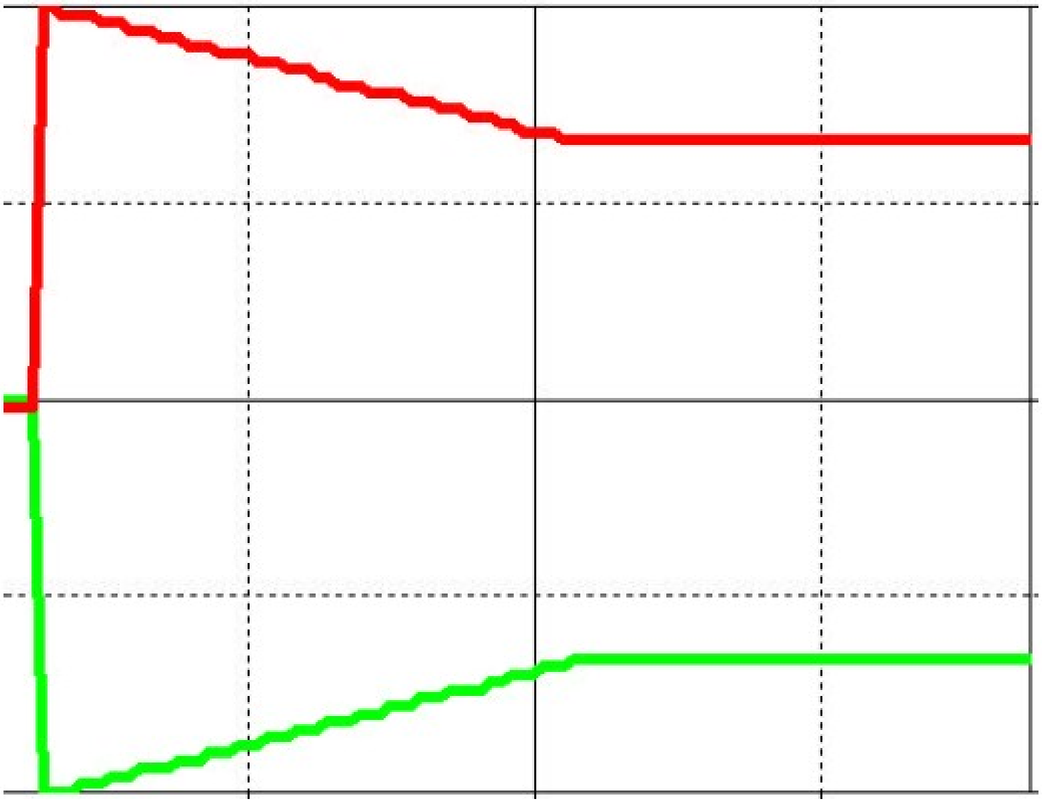
58) Since derivative gain is based on a rate of change, we are going to use a signal generator to feed a ramping PV into our system. Go back to your RSView project, and open the display called "noprocess". This screen will allow you to control the ramping signal generator, monitor the CV%, and change the Td of the controller.



Rate Td (/100): 0

59) Set your Rate Td at 0. Enable your ramping action by clicking the enable button, and then start your signal generator ramping up.

60) Watch your trending chart in RSView. You will see the PV (green line) at the bottom fo your chart. This PV might be representing a temperature. The red line represents your power output. Since our Kc gain is set to 1, the CV is an exact mirror of the PV.

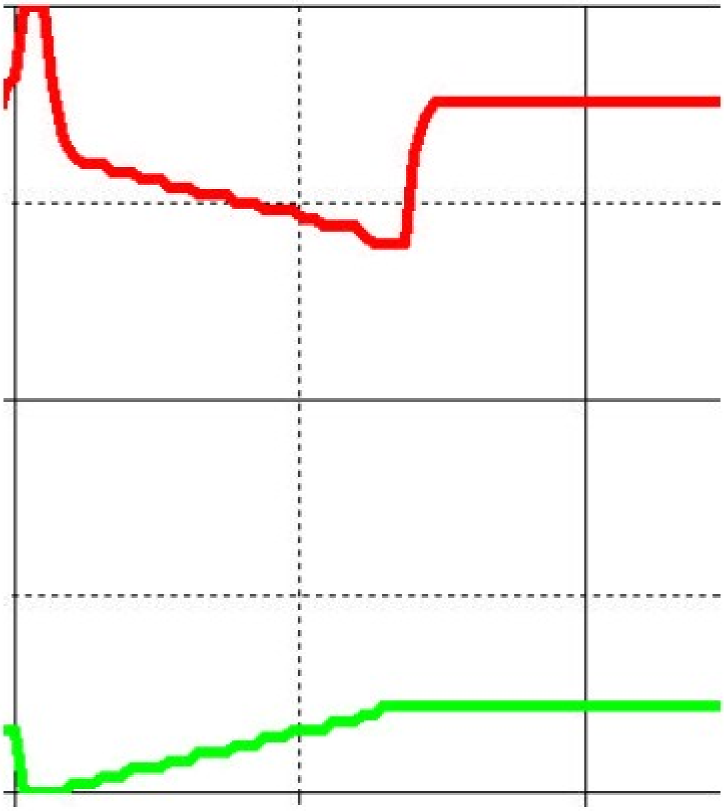


61) The chart below is what you observed with no derivative action. Notice that the output power decreases proportionally as the PV increases. If the PV stops increasing and remains the same, the Output power stops decreasing and remains the same as well.

62) Next we are going to put some derivative into the process, so you can see its effect on the output power. Before we do this, lets try to anticipate how the derivative action is going to change the output power. Remember the derivative action attempts to resist change. As long as our PV is changing, we are going to have some derivative action. If our PV stops changing, then we are no longer going to have the derivative action. The derivative sees that our temperature is increasing rapidly. Therefore, it will attempt to decrease the CV as long as this fast rate of change exists. Similarly, if the derivative sees the temperature falling rapidly, it will attempt to add output to the controller to compensate.

63) To keep you from having to switch back to RSLogix, you have a numeric entry in RSView that will allow you to change the Td value. We are going to enter the value of 30 into RSView, and this value will show up on the setup screen of the PID instruction as .3.

64) Press the RESET button in RSView, and then enable the ramping action. Start your PV ramping up. Notce the way the Control Variable reacts. Immediately it is decreased because of the rate of change of the PV. Now stop the ramping action. You will see the CV goes back to the proportional value, and derivative action no longer exists.

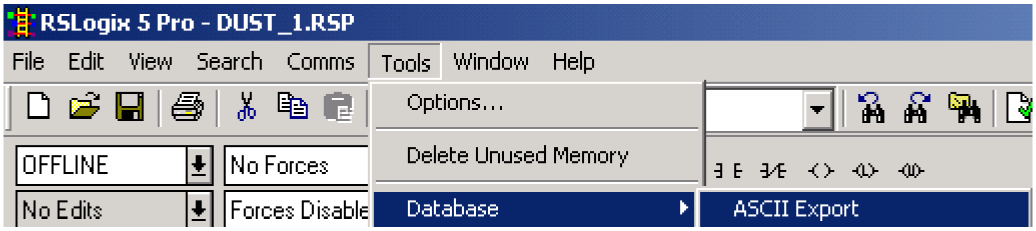


Importing and Exporting the Documentation Database

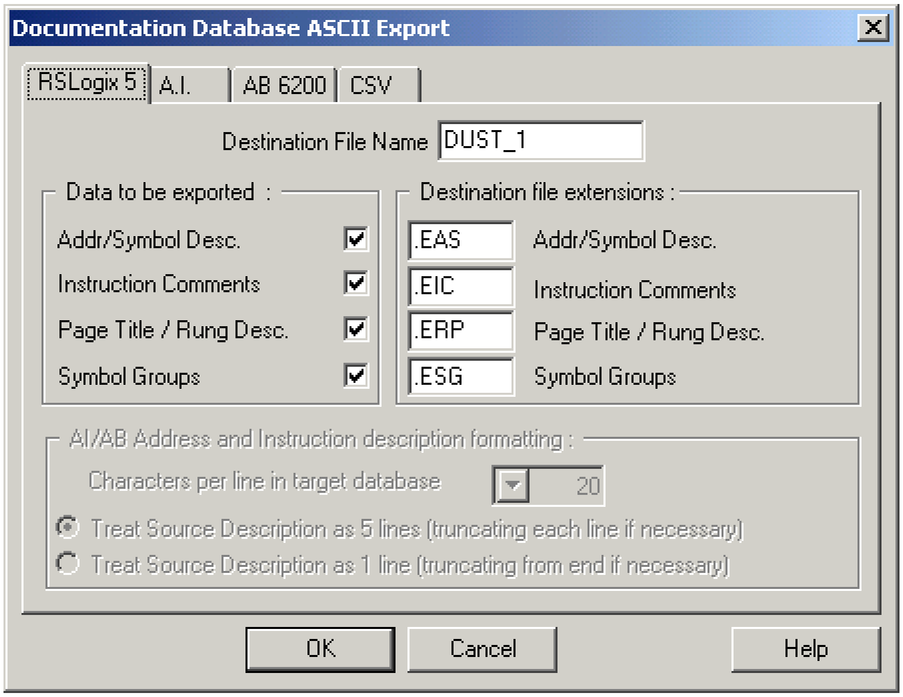
If documentation of a program gets lost, and you can still locate the original program file (from some time back), it is possible to export the documentation from the original program, and import documentation into the current running program. This would save you from having to re-document every address.

Exporting the Database:

To export the documentation database, open the original program containing documentation, then click Tools | Database | ASCII export from the menu bar.



Next choose the format and files in which you wish to export the documentation. If you are importing all documentation into another RSLogix 5 project, press OK.


Documentation Database ASCII Export

RSLogix 5 | A.I. | AB 6200 | CSV

Destination File Name: DUST_1

Data to be exported :
- Addr/Symbol Desc. ☑
- Instruction Comments ☑
- Page Title / Rung Desc. ☑
- Symbol Groups ☑

Destination file extensions :
- .EAS Addr/Symbol Desc.
- .EIC Instruction Comments
- .ERP Page Title / Rung Desc.
- .ESG Symbol Groups

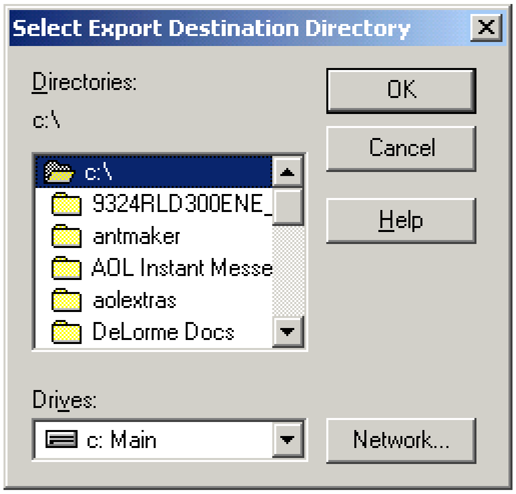
AI/AB Address and Instruction description formatting :

Characters per line in target database: 20

● Treat Source Description as 5 lines (truncating each line if necessary)

○ Treat Source Description as 1 line (truncating from end if necessary)

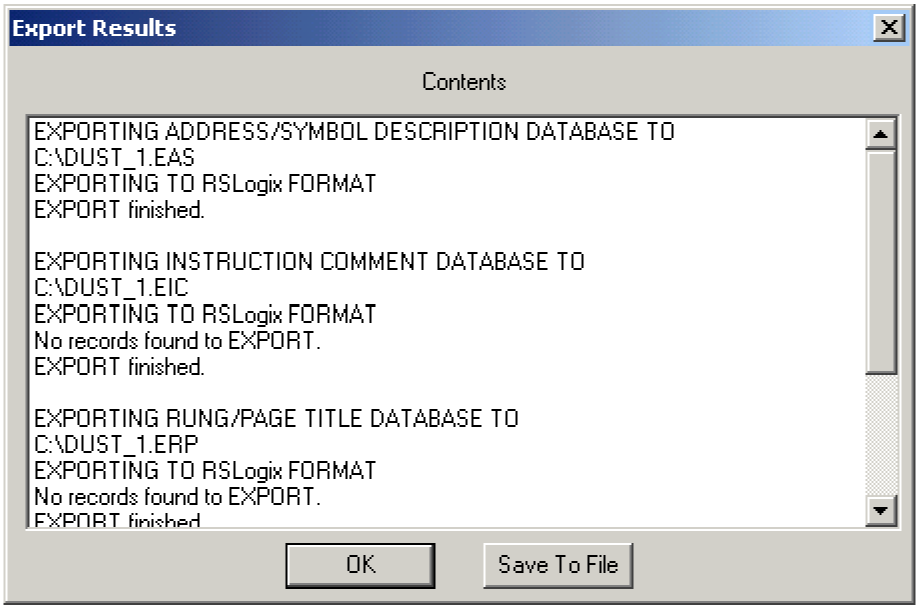
Next, you are asked for a destination directory. Choose the destination for all four files that are to be exported. For this example, we choose the C: drive. If you are saving the documentation to a floppy drive, choose A:.



Directories:
c:\

Drives: c: Main

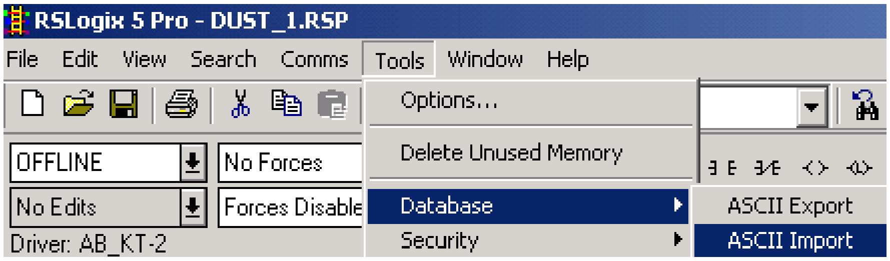
You will now see a report of the results of the export. You can choose to save this report to a file, or just click OK.



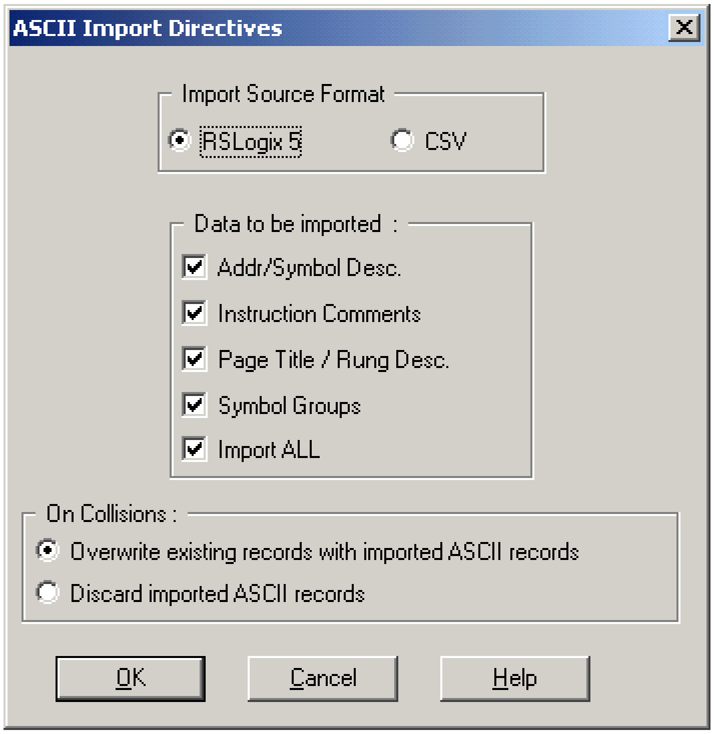
```
EXPORTING ADDRESS/SYMBOL DESCRIPTION DATABASE TO
C:\DUST_1.EAS
EXPORTING TO RSLogix FORMAT
EXPORT finished.

EXPORTING INSTRUCTION COMMENT DATABASE TO
C:\DUST_1.EIC
EXPORTING TO RSLogix FORMAT
No records found to EXPORT.
EXPORT finished.

EXPORTING RUNG/PAGE TITLE DATABASE TO
C:\DUST_1.ERP
EXPORTING TO RSLogix FORMAT
No records found to EXPORT.
EXPORT finished.
```

Importing the Database

To import a documentation database, be sure you have the undocumented project open, the click Tools | Database | ASCII Import from the menu bar.



Next you are asked the format of the documentation you are importing. If all documentation has been exported by RSLogix 5 for RSLogix 5, then press OK if you want to overwrite existing records with imported records. If you already had some documentation in your program that you want to keep (and not import new documentation for those addresses that are already documented), Then choose to discard imported records where collisions occur.


ASCII Import Directives

Import Source Format

● RSLogix 5 ○ CSV

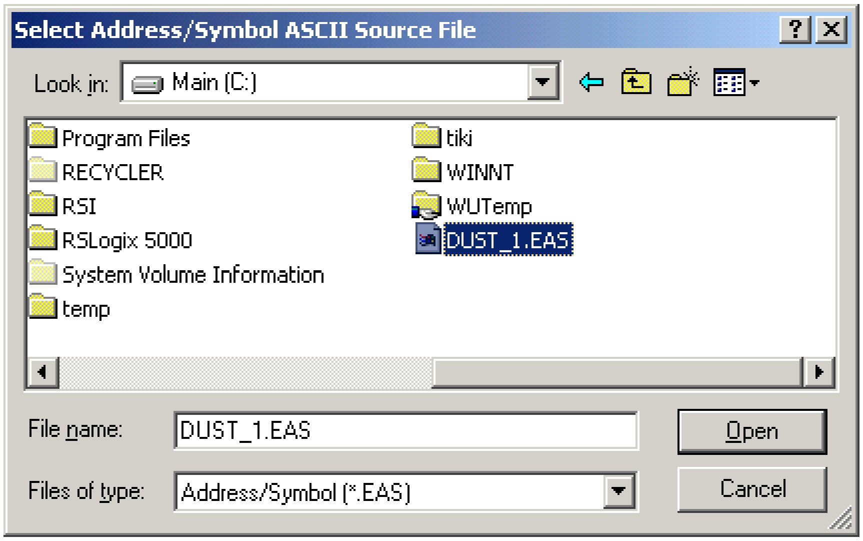
Data to be imported :

☑ Addr/Symbol Desc.
☑ Instruction Comments
☑ Page Title / Rung Desc.
☑ Symbol Groups
☑ Import ALL

On Collisions :

● Overwrite existing records with imported ASCII records
○ Discard imported ASCII records

In the pull down menu of the Look in box. Choose the location of the documentation files that we exported earlier. For this example, our documentation was on the C: Drive. Click on the documentation file (EAS extension), as shown below, and the press *OPEN*.



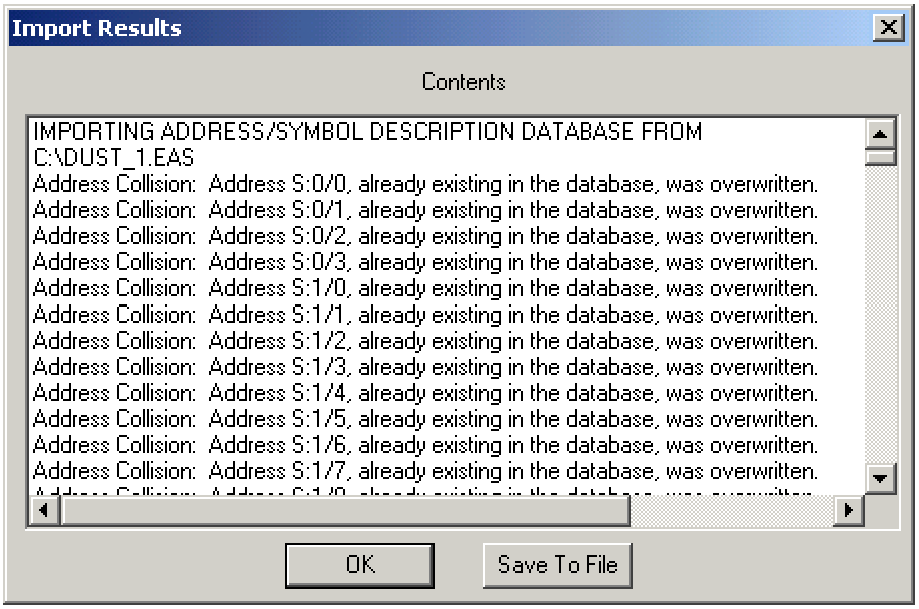
Look in: Main (C:)

File name: DUST_1.EAS

Files of type: Address/Symbol (*.EAS)

Repeat this procedure for the other 3 documentation files... EIC, ERP, and ESG.

You will then be told the results of the import operation. You can choose to save the results to a file, or hit OK to finish. You should not have documentation.

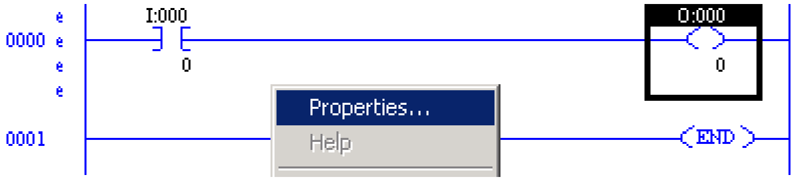


```
IMPORTING ADDRESS/SYMBOL DESCRIPTION DATABASE FROM
C:\DUST_1.EAS
Address Collision:  Address S:0/0, already existing in the database, was overwritten.
Address Collision:  Address S:0/1, already existing in the database, was overwritten.
Address Collision:  Address S:0/2, already existing in the database, was overwritten.
Address Collision:  Address S:0/3, already existing in the database, was overwritten.
Address Collision:  Address S:1/0, already existing in the database, was overwritten.
Address Collision:  Address S:1/1, already existing in the database, was overwritten.
Address Collision:  Address S:1/2, already existing in the database, was overwritten.
Address Collision:  Address S:1/3, already existing in the database, was overwritten.
Address Collision:  Address S:1/4, already existing in the database, was overwritten.
Address Collision:  Address S:1/5, already existing in the database, was overwritten.
Address Collision:  Address S:1/6, already existing in the database, was overwritten.
Address Collision:  Address S:1/7, already existing in the database, was overwritten.
```

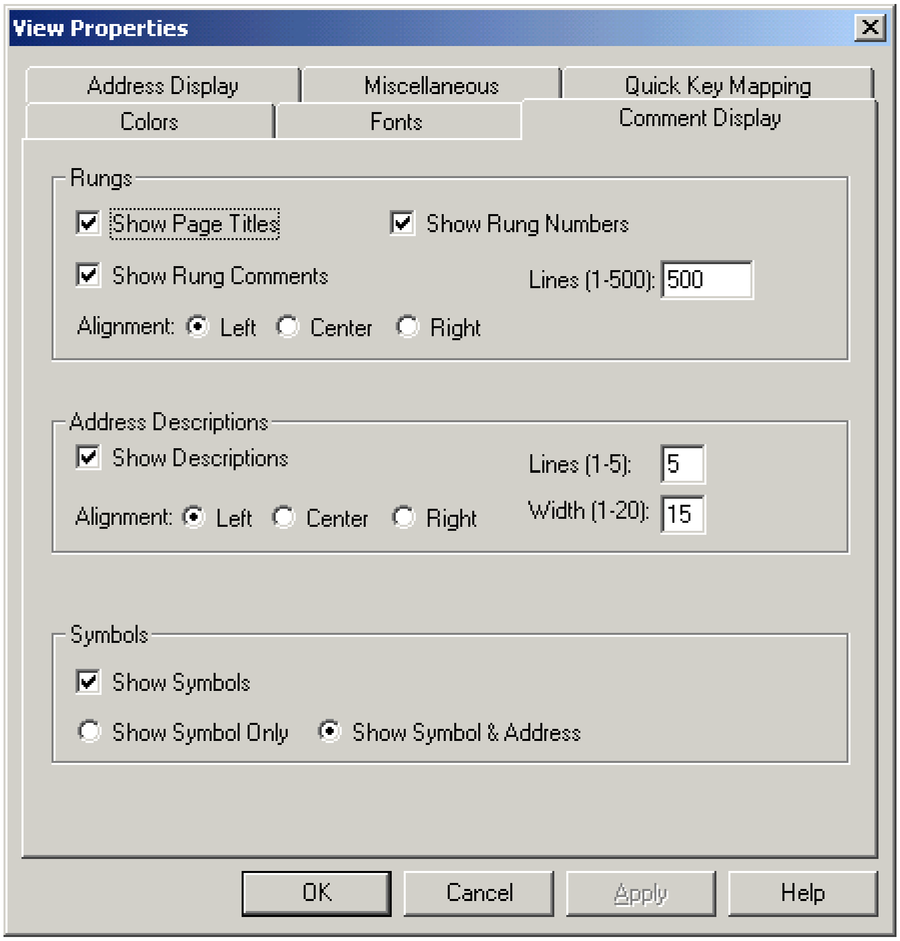
Changing the View Settings

View settings are available for the user to set his own environment under RSLogix. Each user of a professional windows operating system (based on the windows login) can have his own view settings.

1) To access the view settings, right click the background of the ladder view and choose properties.



2) The Comment display tab allows the user to show or suppress certain pieces of documentation.



Rungs

☑ Show Page Titles ☑ Show Rung Numbers

☑ Show Rung Comments Lines (1-500): 500

Alignment: ● Left ○ Center ○ Right

Address Descriptions

☑ Show Descriptions Lines (1-5): 5

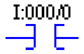
Alignment: ● Left ○ Center ○ Right Width (1-20): 15

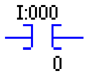
Symbols

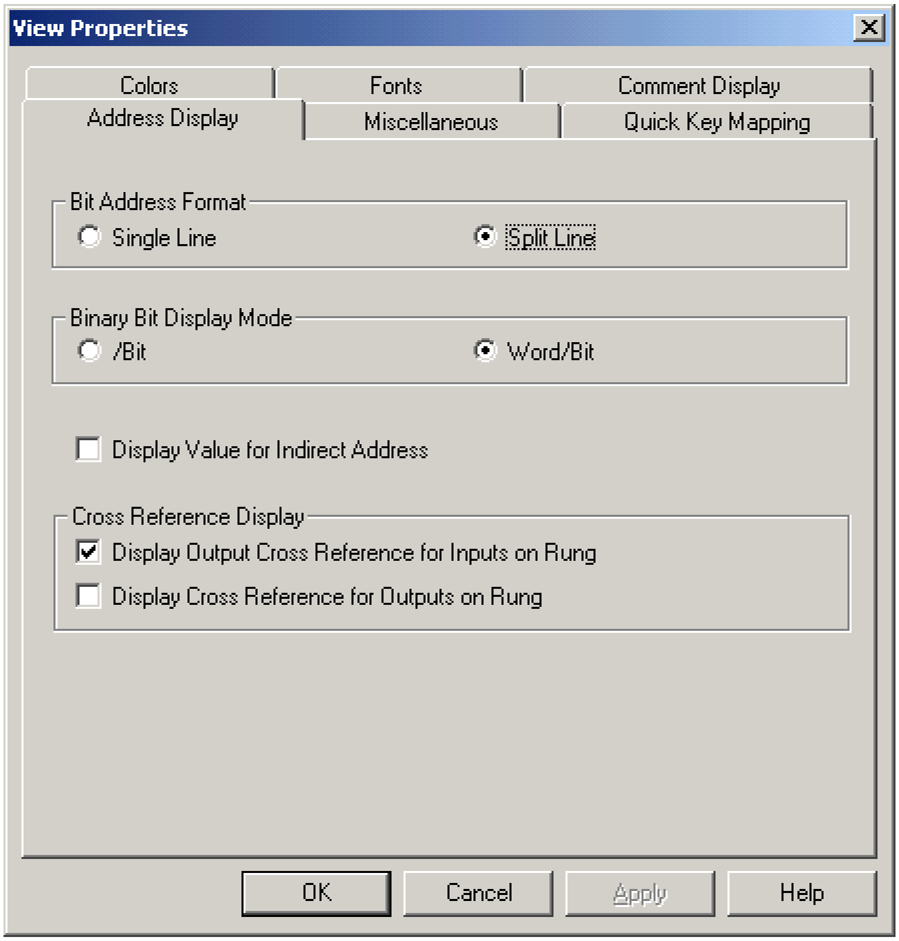
☑ Show Symbols

○ Show Symbol Only ● Show Symbol & Address

3) The address display tab shows information about the way addresses appear in the ladder logic. For the Bit Address format, if 'Single Line' is chosen, the entire address will appear above the instructions. In split line mode, the bit address is displayed underneath an instruction. Below is an example of Single Line (to the left) and Split Line mode (to the right).







Bit Address Format
○ Single Line ● Split Line

Binary Bit Display Mode
○ /Bit ● Word/Bit

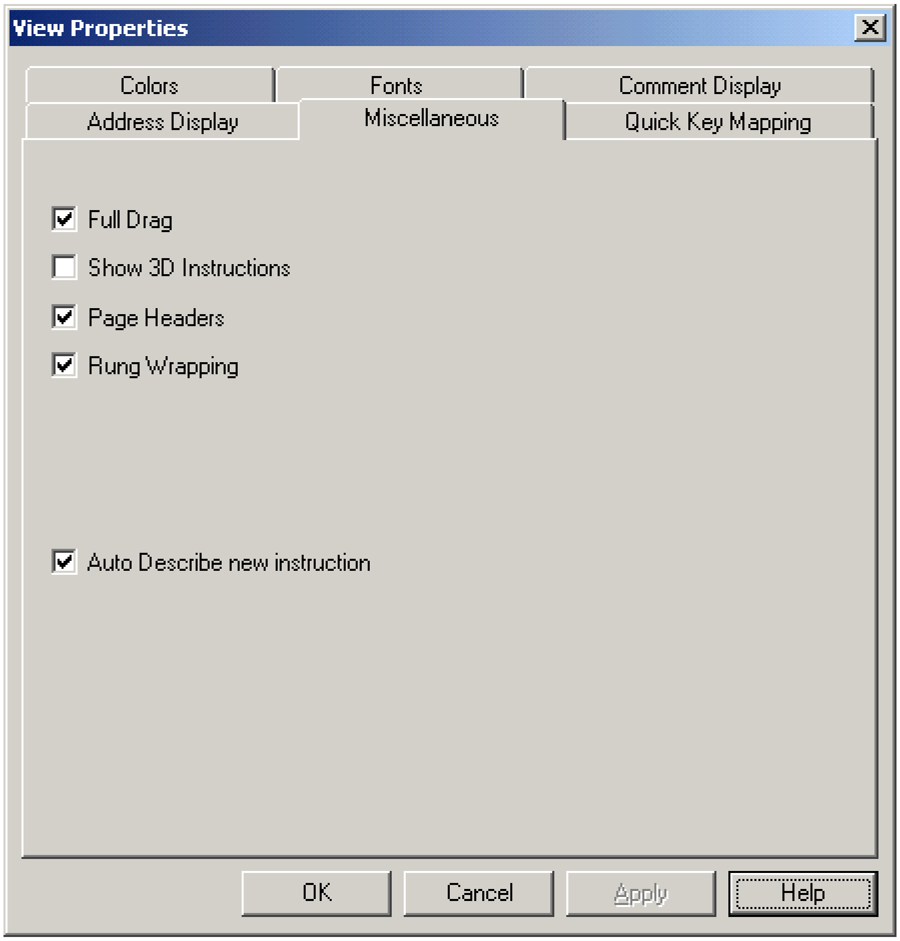
□ Display Value for Indirect Address

Cross Reference Display
☑ Display Output Cross Reference for Inputs on Rung
□ Display Cross Reference for Outputs on Rung

For the Binary Bit Di play Mode: in the /bit format, all bits in the file are numbered consecutively. In the Word/Bit format, the Word level is added to the address. B3/16 and B3:1/0 are exactly the same address. Because a word contains sixteen bits, word 0 contains B3/0 to B3/15. The very next bit is B3/16 which is in the next word of memory.

Other examples: B3/96 is bit 0 in word 6, so therefore the long address would be B3:6/0. B3/159 is bit 15 in word 9, so therefore the long address would be B3:9/15. If you multiply word 9 times 16 bits, then add the bit number (15), you will find the short address to be B3/159.

4) On the Miscellaneous tab, **Full Drag** mode lets you see the address or symbol when dragging an instruction in the ladder view. **3D instructions** put shadows behind instructions to give your logic a 3 dimensional look. **Page headers** display information about the ladder under the title bar in the ladder view. **Rung Wrapping** will wrap rungs to the next line if they are too large to fit on the current page. If rung wrapping is shut off, you must scroll to the right if a rung is too large to fit in the ladder view. **Auto Describe new instruction** will bring up a dialog screen for you to enter a description and symbol when new addresses are used in the project.



☑ Full Drag

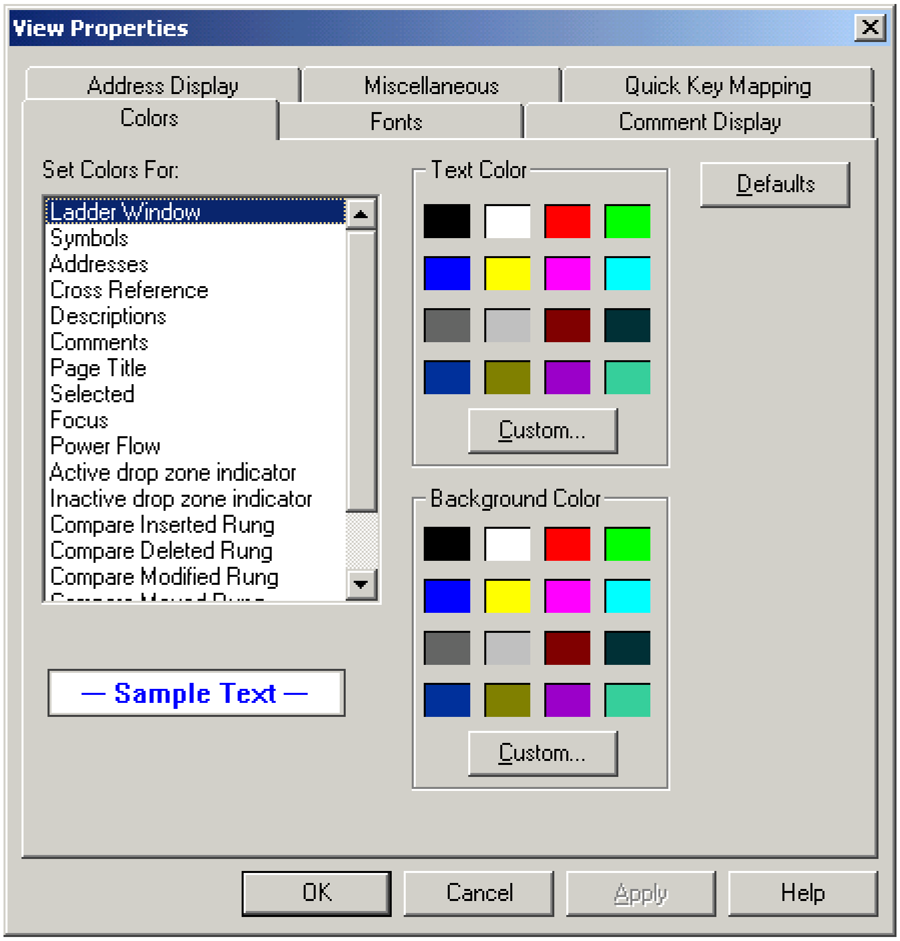
□ Show 3D Instructions

☑ Page Headers

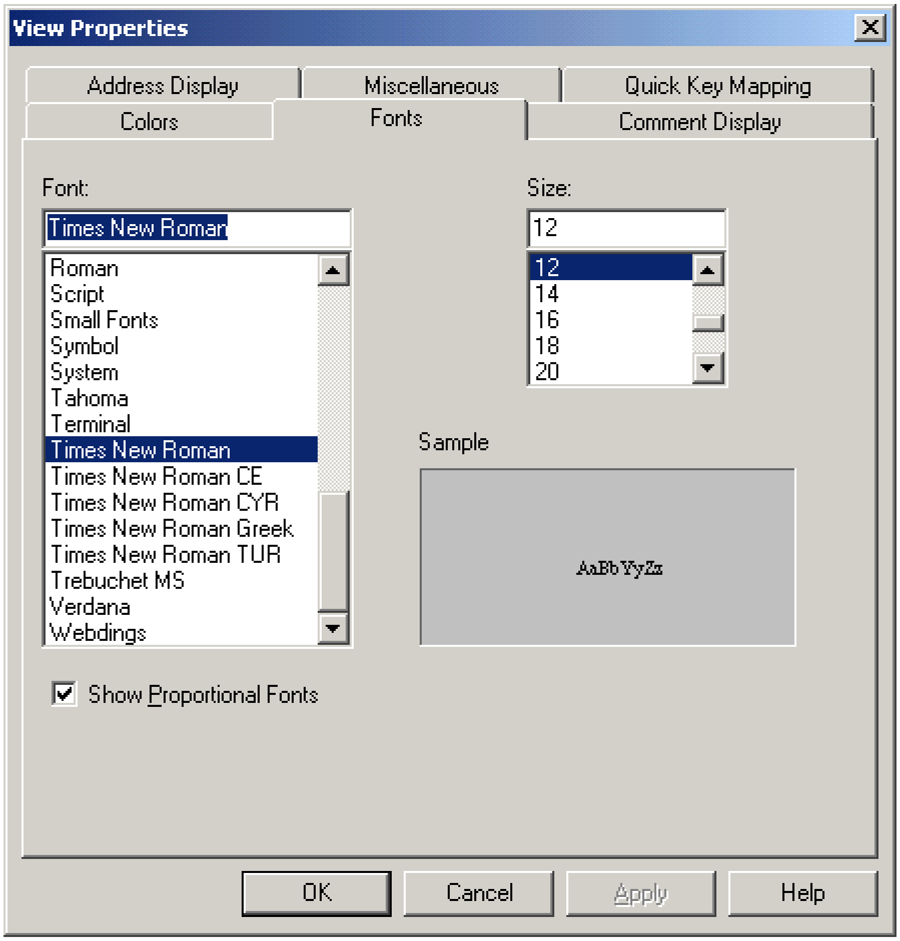
☑ Rung Wrapping

☑ Auto Describe new instruction

5) The Colors tab lets you change the colors of items in the ladder view. Click the item on the left you wish to set colors for, and then choose a text color and background color to the right. You will notice the Sample Text will change so you can see how the colors will look in the ladder. You also have a button for default colors if you would like to revert to the original color settings for RSLogix.



6) Below is an example of the Fonts tab.   Usually the default font is fine, but occasionally, on some systems, you may want to change the font type and increase or decrease the font size.



View Properties

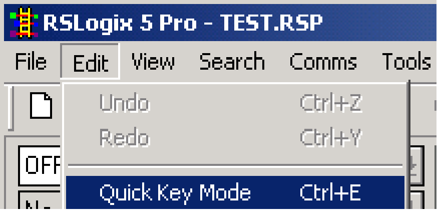
Font: Times New Roman

Size: 12

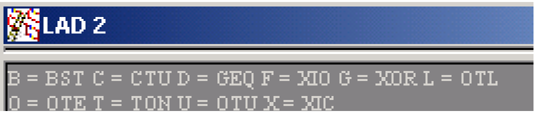
☑ Show Proportional Fonts

7) Under the Edit Menu, quick key mode can be turned on or off. Quick key mode allows a user to add instructions to ladder logic with the stroke of one key. When quick key mode is on, a header appears below the title bar indicating how the quick keys are mapped. In this example, if quick key mode was on, the user could press the letter O when editing ladder logic to insert the OTE (output to energize) instruction.

From the Edit Menu, Quick Key mode can be activated:

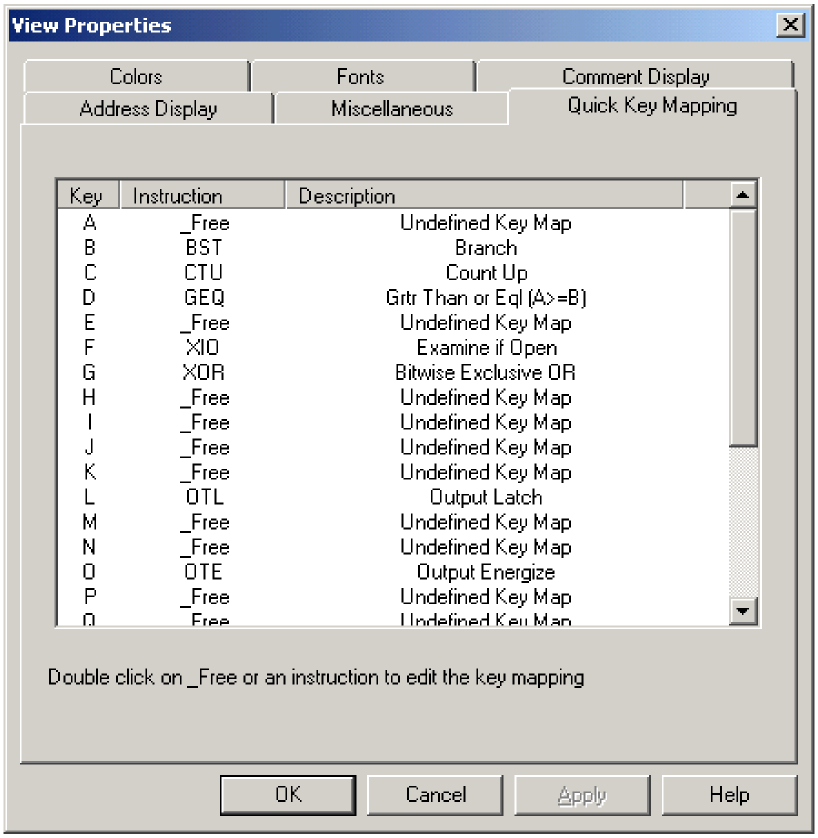


While in quick key mode, a header below the title bar indicates mapping:



B = BST C = CTU D = GEQ F = XIO G = XOR L = OTL
O = OTE T = TON U = OTU X = XIC

Under View|Properties, Other keys can be mapped:



Key	Instruction	Description
A	_Free	Undefined Key Map
B	BST	Branch
C	CTU	Count Up
D	GEQ	Grtr Than or Eql (A>=B)
E	_Free	Undefined Key Map
F	XIO	Examine if Open
G	XOR	Bitwise Exclusive OR
H	_Free	Undefined Key Map
I	_Free	Undefined Key Map
J	_Free	Undefined Key Map
K	_Free	Undefined Key Map
L	OTL	Output Latch
M	_Free	Undefined Key Map
N	_Free	Undefined Key Map
O	OTE	Output Energize
P	_Free	Undefined Key Map
Q	_Free	Undefined Key Map

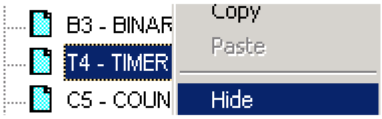
Double click on _Free or an instruction to edit the key mapping

Hide and Unhide Program and Data Files

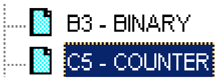
Program and Data files that are seldom accessed can be hidden from view. They can also become inadvertently hidden from view.

Hiding a Data or Program file:

To hide a file from view, Right click the file, and choose *Hide:*

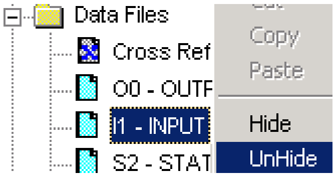


When a file is hidden it becomes invisible from view of the user:

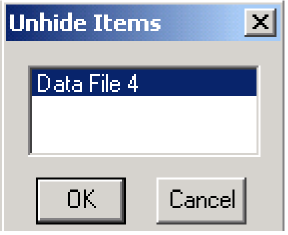


Unhiding a Data or Program File:

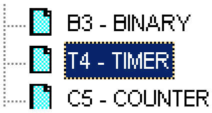
To unhide a data or program file, right click on any data or program file (or data/program file folder), and choose *Unhide.*



A dialog box will appear containing a list of the hidden data and program files. Choose the files you wish to unhide, then press OK.



You will then see that your program or data files are again shown in the project tree.



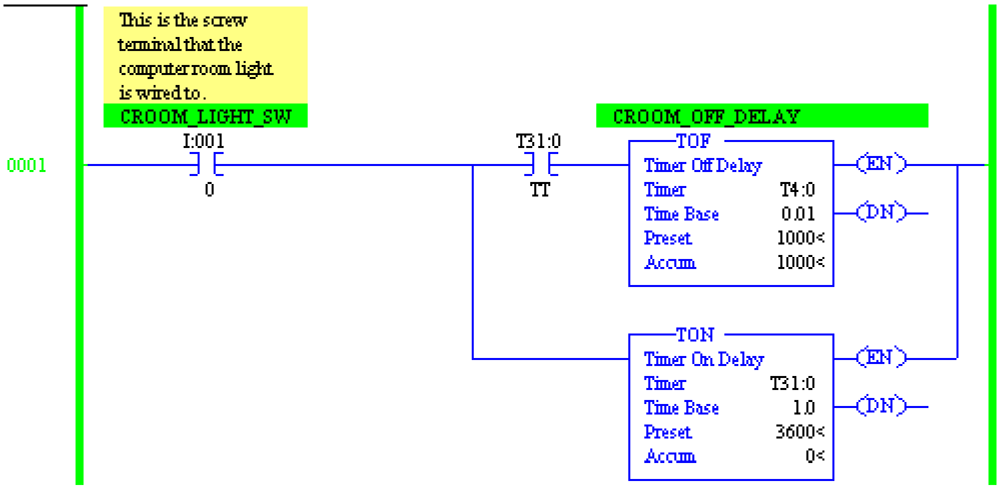
On line Editing

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

- You cannot expand or create a data table on line in the SLC, and you must be in program mode (or off line) to expand a data table on line in the PLC-5.
- You cannot make an on line edit if the key switch is in Run Mode.
- In the PLC-5, if backplane switch 8 is on, you cannot make an on line edit.
- 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 I:001/1. If you click on bit 0 and attempt to make a change, nothing happens.

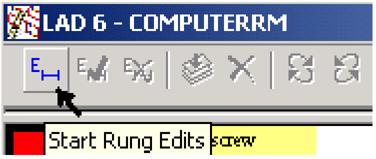


This is the screw terminal that the computer room light is wired to.

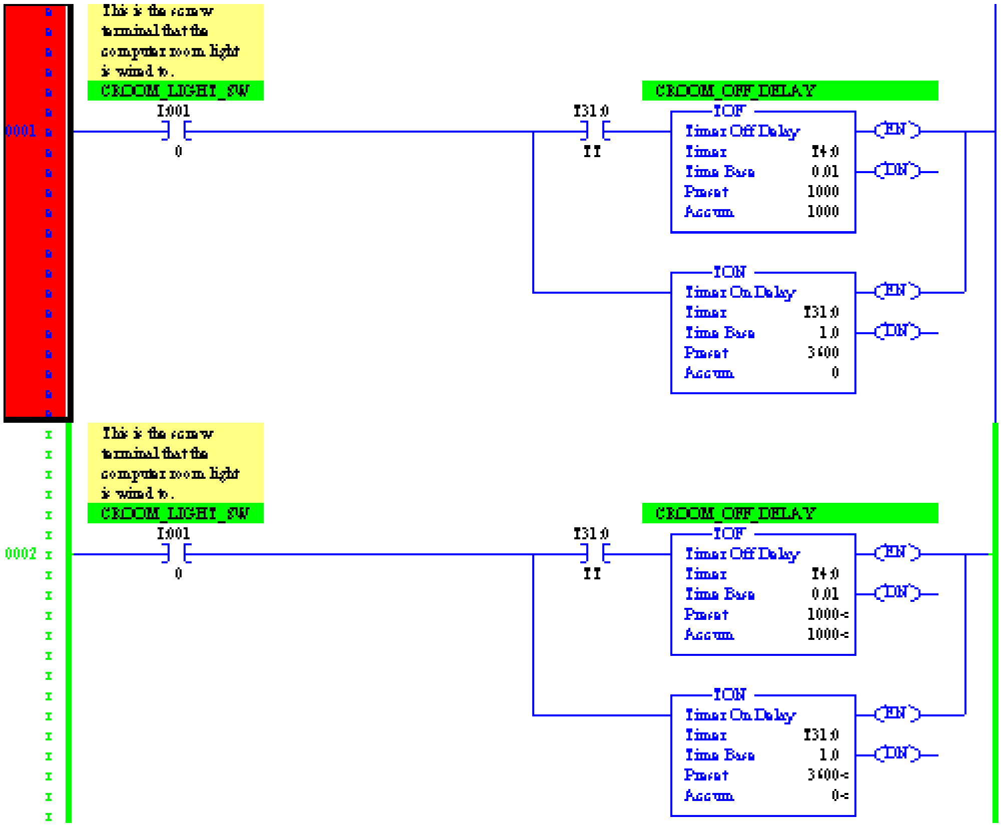
Step 1) Start Rung Edits

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 'Edit' on the menu bar, click rung edits, then start 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) 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.



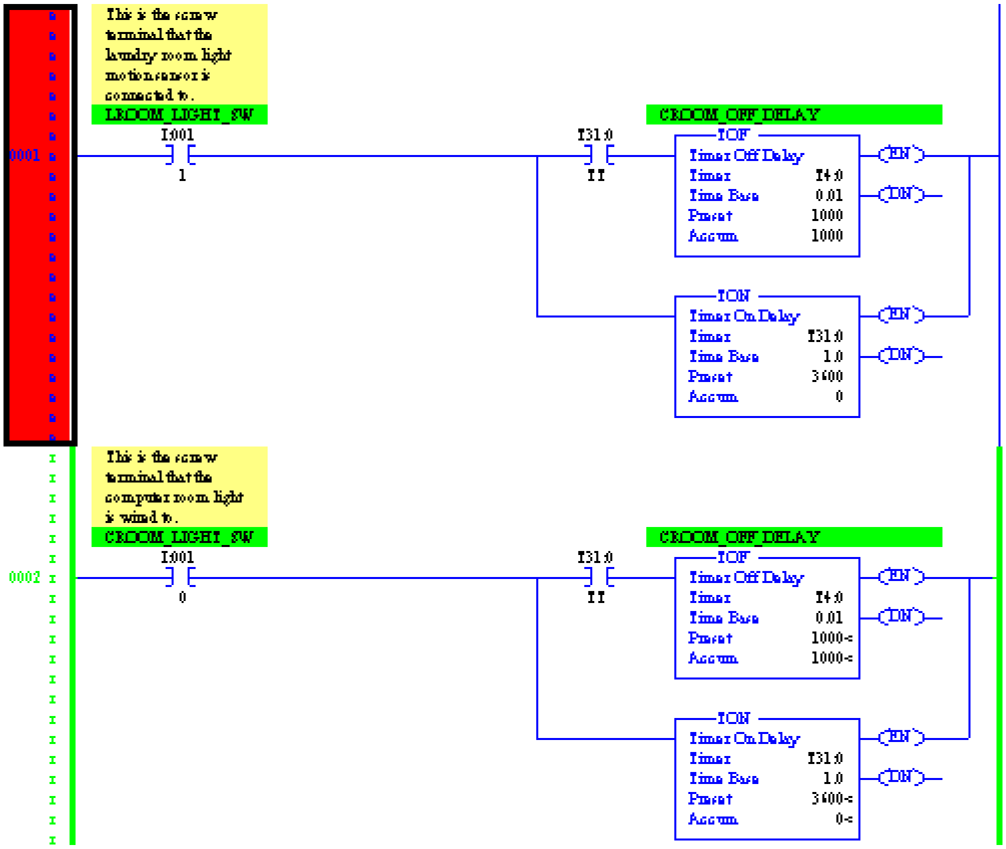
Step 2) Make Changes

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

If you were adding a new rung, you can now add the new logic to the rung.

Be careful not to add any logic that will fault the processor or cause damage to personnel or equipment. In the next step, the changes we make will be sent to the processor.

In this example, bit 0 is being changed to bit 1 on the input. If you are using an SLC, the addressing will be slightly different.



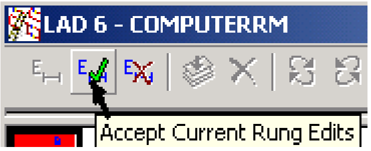
This is the screw terminal that the laundry room light motion sensor is connected to.

This is the screw terminal that the computer room light is wired to.

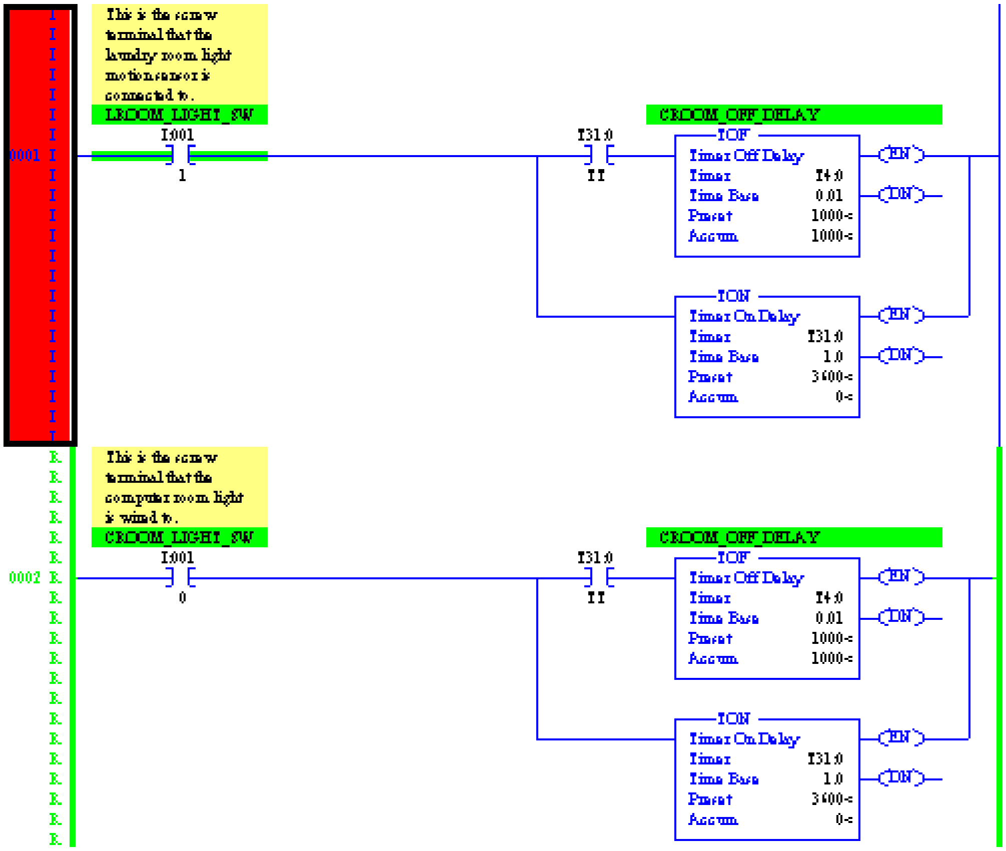
Step 3) Accept Edits

Now that your rung is set up as you need it, it's time to send the edits to the processor. There are several ways to perform the next three steps.

- Right click the rung number, and accept edits
- Click Edit | Rung Edits | Accept rung from the menu bar
- Click the Accept Edits icon in the on line editing tool bar as shown below



If you need to accept multiple rung edits, you can hold down the CTRL key and click on the rungs you need to accept, or if you have many edits all together, you can click the first rung number, hold shift, then click the last rung number of the range you wish to accept. Notice in the margin rung 1 is marked for insertion, and rung 2 is marked for removal. The I's and R's are capitol 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.


This is the screw terminal that the laundry room light motion sensor is connected to.

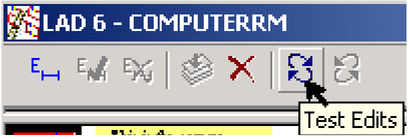
This is the screw terminal that the computer room light is wired to.

Step 4) Test Edits

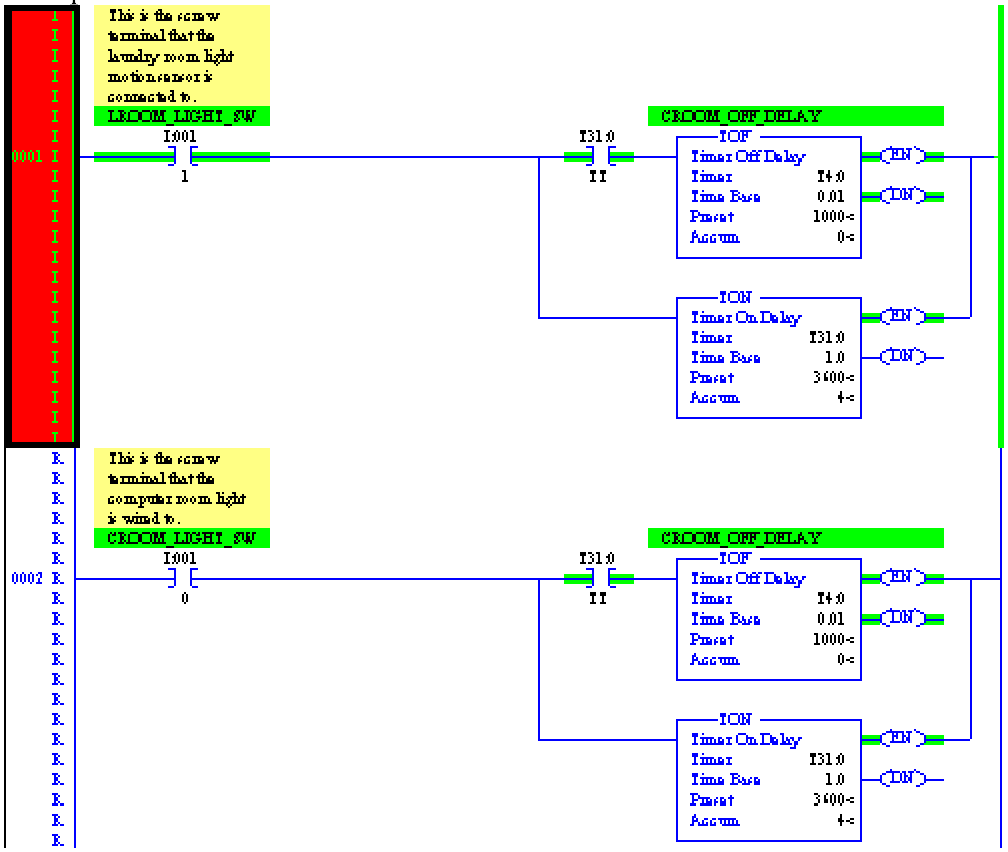
When you test edits, the new or modified rungs will become active. The old rungs will be left in the processor until we are sure our new rungs are working properly. Be aware that if you change an output address, there might no longer be logic writing to that address. This means that you could abandon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

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 untest to revert to the old rung while you make other changes to the new rung.



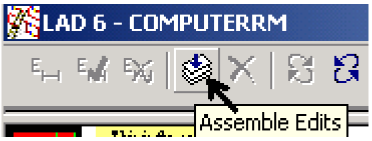
Notice the power rails:


This is the screw terminal that the laundry room light motion sensor is connected to.

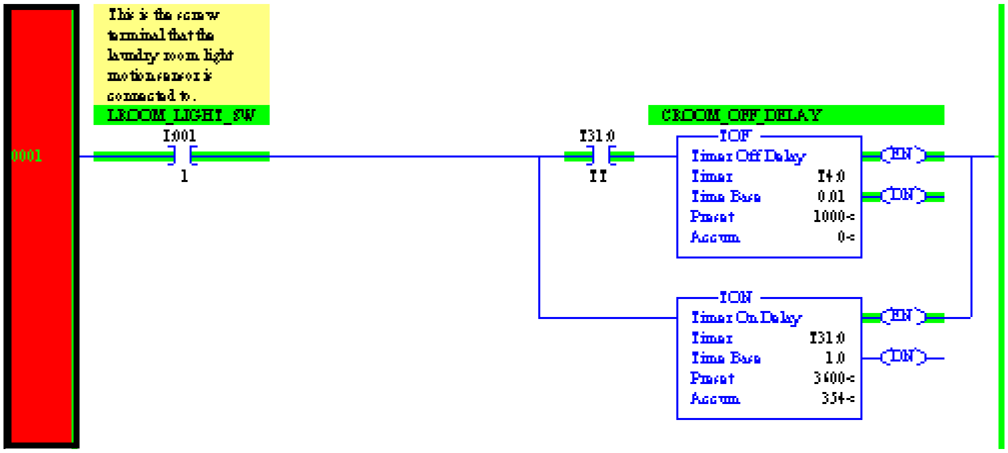
This is the screw terminal that the computer room light is wired to.

Step 5) Assemble Edits

If you 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.



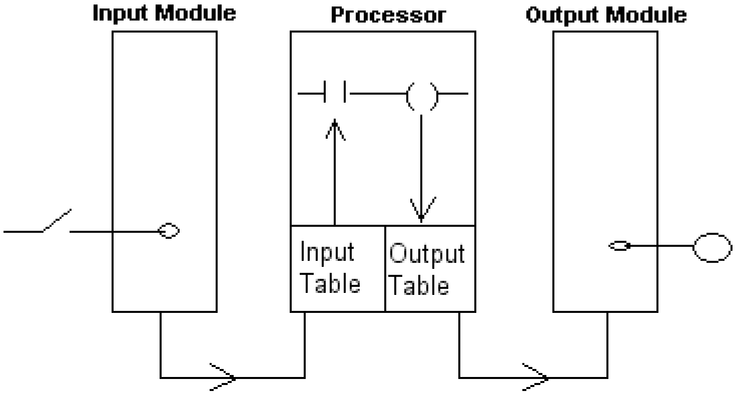
Notice the Logic now appears to be normal:


This is the screw terminal that the laundry room light motion sensor is connected to.

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 data table
3. The XIC instruction goes true
4. The OTE is enabled
5. A 1 is written to the output data table
6. The light will energize on the output module

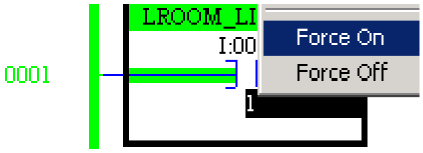
Forcing the input:
If you place a jumper across the switch, you would have the same effect as the switch always being shut. A 1 would always be in the data table, the logic would be true, and the light would energize. The same effect apply to forcing. Forcing the input on would result in a 1 in the input data table for the switch, and all logic would be executed as if the switch was shut. The opposite applys 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.

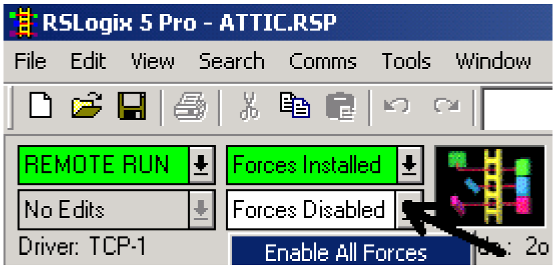
Note: Even though forcing an output does not directly effect the data table, The field device itself could feed an input back into the processor causing other things to happen in logic. Know your system before using the force feature.

There are several ways to force an output. Forcing can be applied from ladder logic, from the force table itself, or in the I/O configuration on a PLC-5.

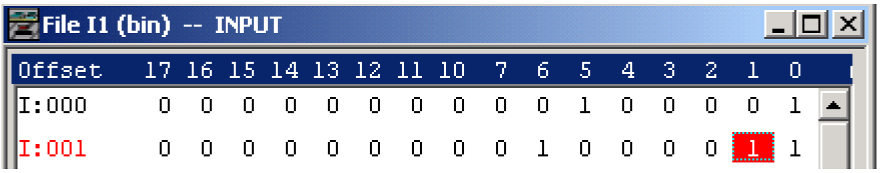
In this example, we will force an input directly from ladder logic. Right click on the input address, and choose 'Force On'.



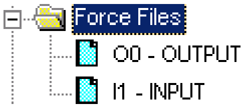
Notice the force light on the processor begins to flash indicating that forces are installed, but not enabled. The force can be enabled from the on line tool bar as shown below:



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.


File I1 (bin) -- INPUT

Offset	17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0
I:000	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
I:001	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1

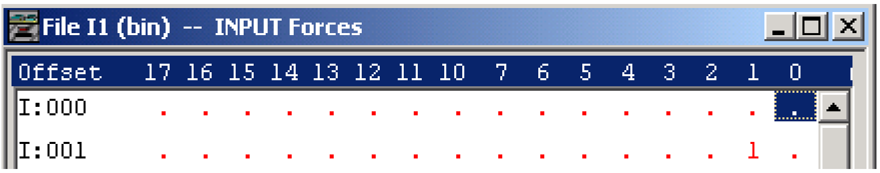


Force Files

O0 - OUTPUT

I1 - INPUT

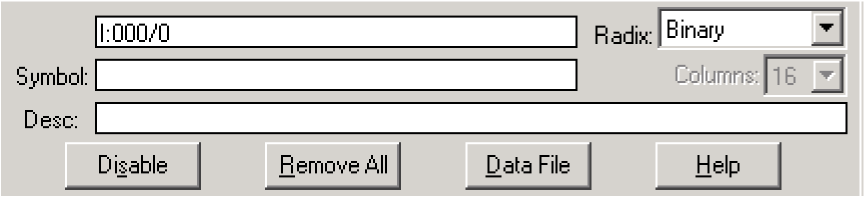
Forcing can also be performed directly from the force table. Scroll down toward the bottom of the project tree, and you will see the input and output force files. If we open the input force file, we can see what bit has been forced.



File I1 (bin) -- INPUT Forces

Offset	17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0
I:000	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
I:001	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.

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. Forcing can be done directly from this table. If you right click on the table, forces can be enabled without using the on line tool bar. You can also use the Enable/Disable buttons at the bottom of the force table.



I:000/0

Radix: Binary

Symbol:

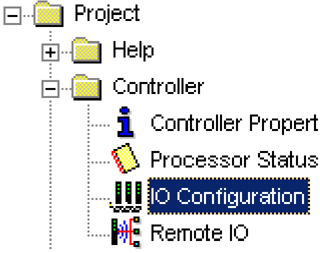
Columns: 16

Desc:

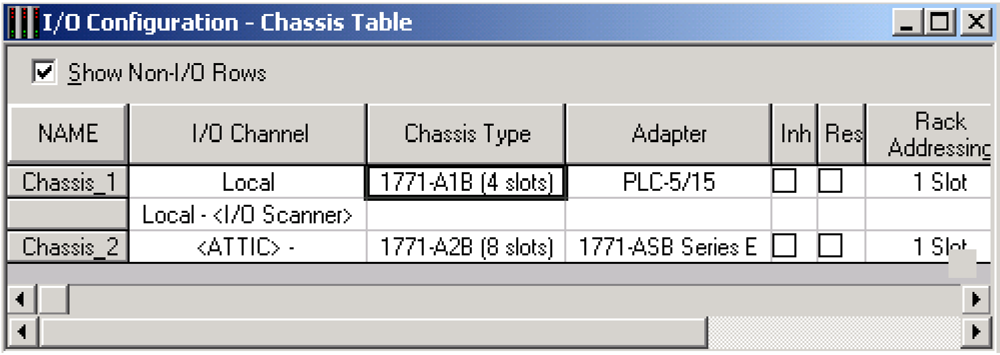
I/O Configuration in the PLC-5

Another option to force I/O is from the I/O configuration table (PLC-5 only). This feature only works if the I/O Configuration has been set up.

1) Open I/O Configuration from the top of the project tree.



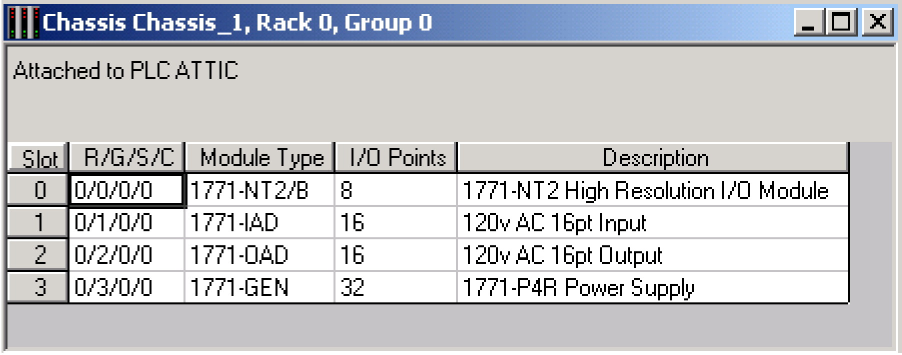
2) Double click the chassis where your I/O resides to reveal the modules in the chassis. In this case, I want to force bit 1 of an input card in the local (4 slot) chassis.



☑ Show Non-I/O Rows

NAME	I/O Channel	Chassis Type	Adapter	Inh	Res	Rack Addressing
Chassis_1	Local	1771-A1B (4 slots)	PLC-5/15	□	□	1 Slot
	Local - <I/O Scanner>					
Chassis_2	<ATTIC> -	1771-A2B (8 slots)	1771-ASB Series E	□	□	1 Slot

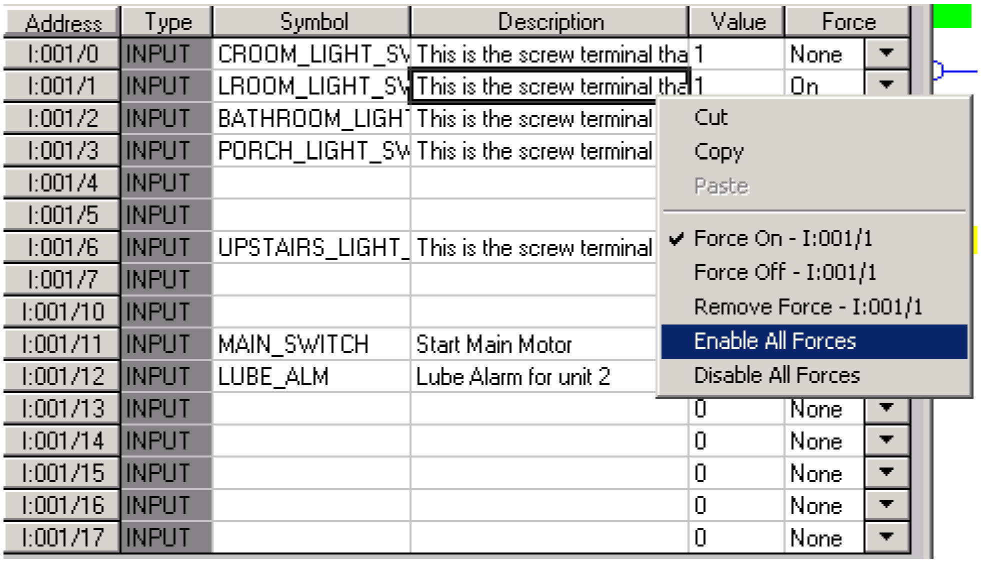
3) Now you are looking at all the modules in the local chassis. Double click the input card itself to reveal all the terminals on the module. (1771-IAD)



Attached to PLC ATTIC

Slot	R/G/S/C	Module Type	I/O Points	Description
0	0/0/0/0	1771-NT2/B	8	1771-NT2 High Resolution I/O Module
1	0/1/0/0	1771-IAD	16	120v AC 16pt Input
2	0/2/0/0	1771-OAD	16	120v AC 16pt Output
3	0/3/0/0	1771-GEN	32	1771-P4R Power Supply

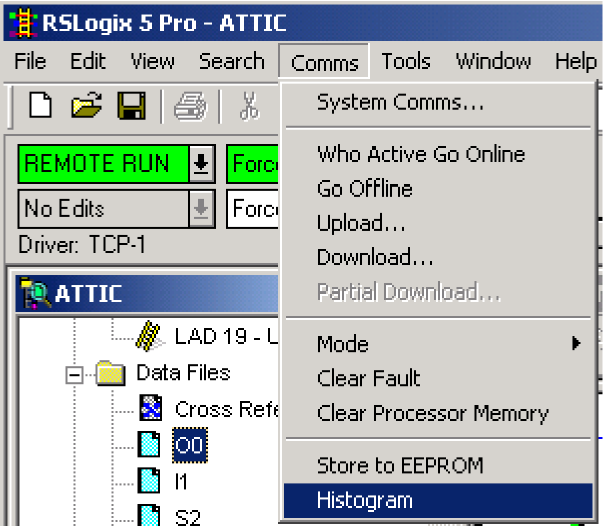
4) Notice all the documentation was brought in from the documentation database. From this screen, documentation can be modified, values can be changed (Usually a bad practice), and forces can be installed. To install a force, just hit the pull down tab in the force column to install an on or off force. Then forces can be enabled with a right-click on the table.



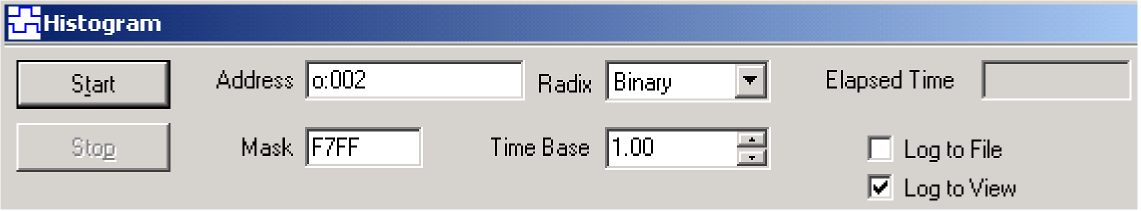
Address	Type	Symbol	Description	Value	Force
I:001/0	INPUT	CROOM_LIGHT_SV	This is the screw terminal tha	1	None
I:001/1	INPUT	LROOM_LIGHT_SV	This is the screw terminal tha	1	On
I:001/2	INPUT	BATHROOM_LIGHT	This is the screw terminal		
I:001/3	INPUT	PORCH_LIGHT_SV	This is the screw terminal		
I:001/4	INPUT				
I:001/5	INPUT				
I:001/6	INPUT	UPSTAIRS_LIGHT_	This is the screw terminal		
I:001/7	INPUT				
I:001/10	INPUT				
I:001/11	INPUT	MAIN_SWITCH	Start Main Motor		
I:001/12	INPUT	LUBE_ALM	Lube Alarm for unit 2		
I:001/13	INPUT			0	None
I:001/14	INPUT			0	None
I:001/15	INPUT			0	None
I:001/16	INPUT			0	None
I:001/17	INPUT			0	None

Using the Histogram

1) To use the Histogram, you must be on line with the processor, then click Comms | Histogram.



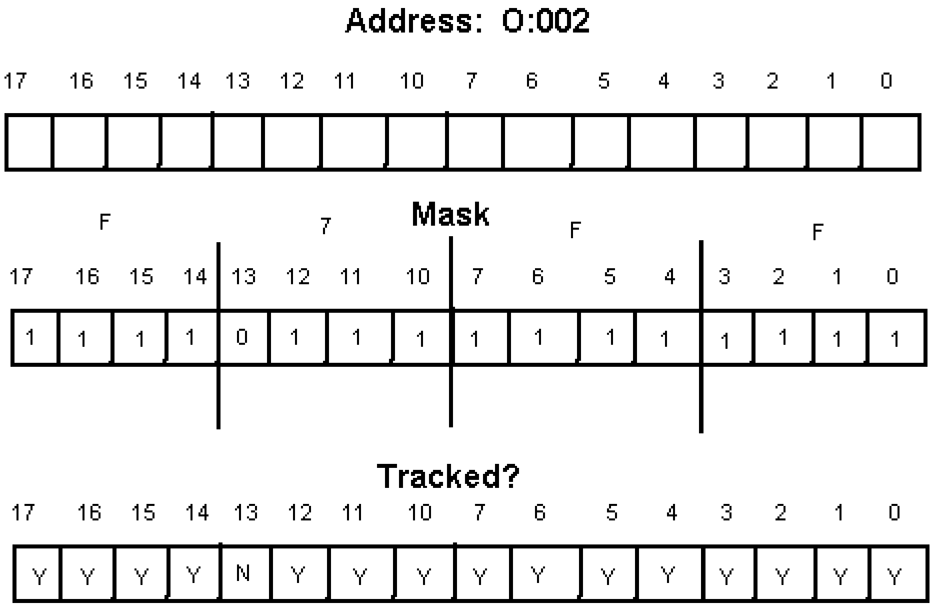
2) Next, you will be setting up the histogram.
   1. Enter the address of the word you wish to run the histogram on. A histogram can be run on most any 16 bit word. If you enter the address at the bit level, the mask will change so the histogram is only run on the bit you selected. This will be explained later.
   2. The Radix indicates the numbering scheme in which you wish to view the data. Options are: Binary, Octal, Hex, or Decimal.
   3. The mask is a hexadecimal number indicating which bits will be monitored in the address you entered. If you use a scientific calculator, specify a Hex radix, then enter a mask value. Convert to binary, and you will see that F7FF is equivalent to four 1's, a zero, and eleven more 1's. This means that all bits will be monitored except bit 13. (Look at the example on the next page).
   4. The time base is the amount of time between grid lines on the timing chart.
   5. If you choose 'Log to File', the histogram data will be stored in a file on your hard drive which can be viewed using a standard text viewer such as Microsoft Notepad. The location of this log file can be set in the histogram properties (right click on the histogram window).
   6. Log to view allows you to view the histogram after you start the chart.



Address: o:002
Radix: Binary
Elapsed Time:
Mask: F7FF
Time Base: 1.00
☐ Log to File
☑ Log to View

Understanding the Mask:

This example shows what bits of the address you entered will be monitored with a mask value of F7FF:



Address: O:002

17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0

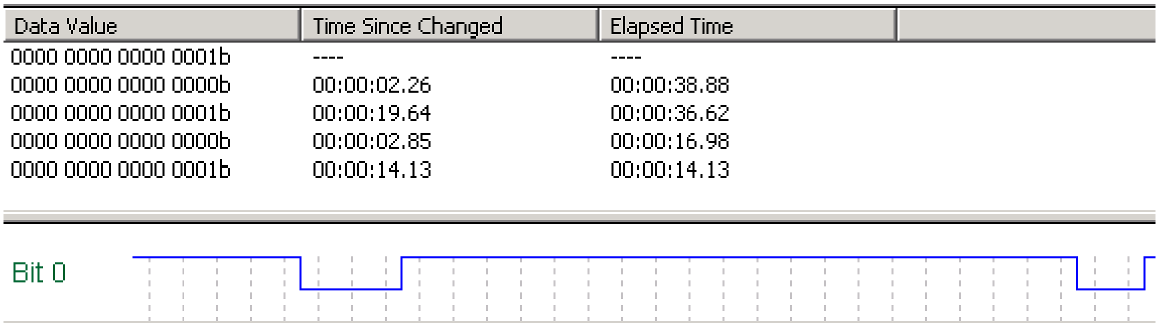
Mask

F				7				F				F			
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

Tracked?

17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0
Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

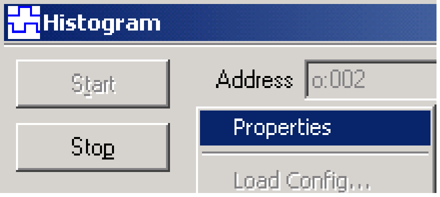
3) Press the start button to begin tracking. In this example you see four transitions: Two ON transitions, and two OFF transitions.



Data Value	Time Since Changed	Elapsed Time	
0000 0000 0000 0001b	----	----	
0000 0000 0000 0000b	00:00:02.26	00:00:38.88	
0000 0000 0000 0001b	00:00:19.64	00:00:36.62	
0000 0000 0000 0000b	00:00:02.85	00:00:16.98	
0000 0000 0000 0001b	00:00:14.13	00:00:14.13	

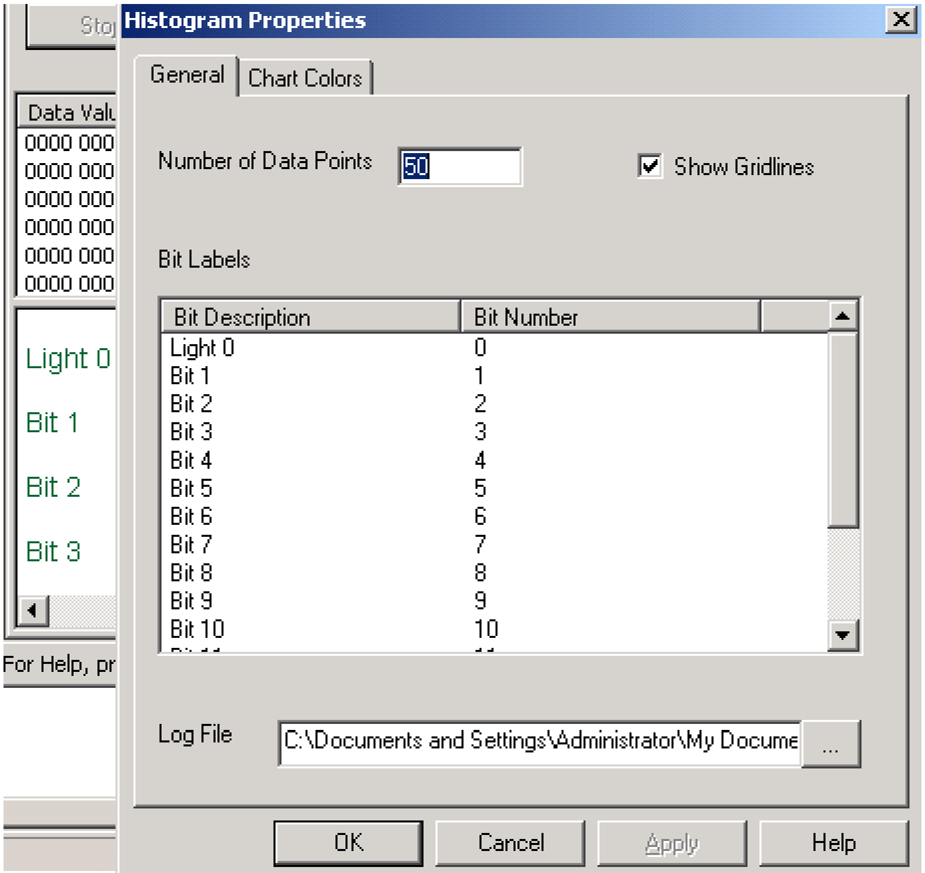
The histogram (top window) will catch most all transitions, however, the timing chart (lower window) works from a time base, and may not catch all transitions.

4) Special configuration for the histogram window can be accessed through the histogram properties. To access the properties, right click within the histogram window, and choose 'properties' from the window that appears.



Address o:002

5) Look at the properties screen below:



Number of Data Points 50

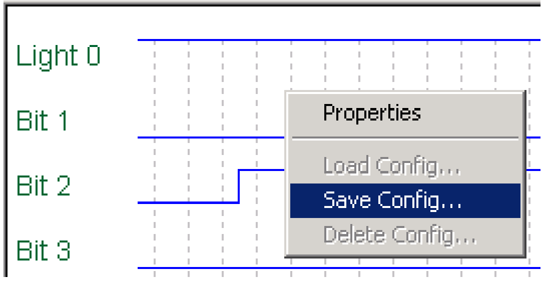
☑ Show Gridlines

Bit Description	Bit Number
Light 0	0
Bit 1	1
Bit 2	2
Bit 3	3
Bit 4	4
Bit 5	5
Bit 6	6
Bit 7	7
Bit 8	8
Bit 9	9
Bit 10	10

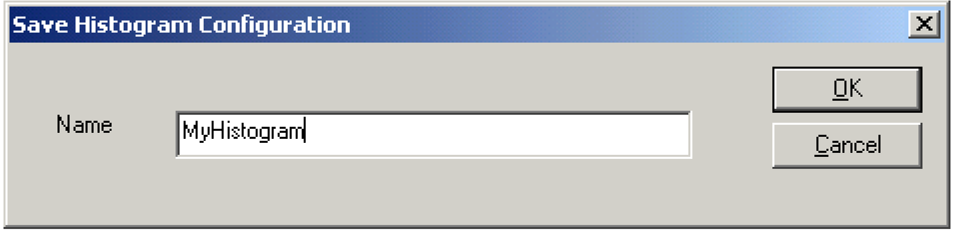
Log File C:\Documents and Settings\Administrator\My Docume ...

1. Select the number of data points you wish to log. The default value is 50. By choosing more data points, you will be able to stop the histogram, and scroll back through the timing chart.
2. The grid lines can be enabled or disabled on the timing chart.
3. To change a bit description, click on one of the bits such as bit0. Change the text, then press enter, and apply your changes. You will notice the text changes on the timing chart.
4. If you selected 'log to file' on the main histogram window, you see or change where this file is located on the system.
5. Click the chart colors tab to change the look of the main histogram window.

6) After you are finished with the histogram, you can save the configuration. Right click on the chart, and choose 'save config'.

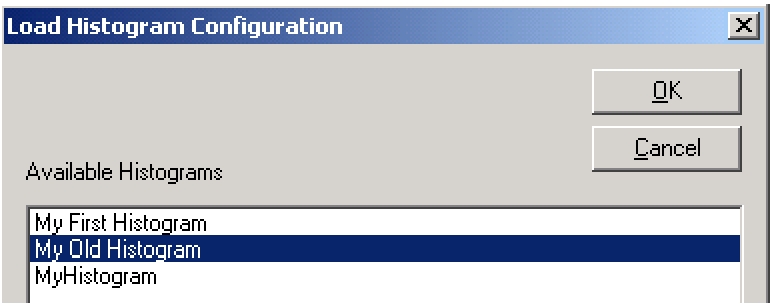


7) You will then be prompted for a name:



Name: MyHistogram

8) You can also revert to other saved histograms if you right click the chart, and choose 'Load Config'. A window will then appear showing histogram configurations that have previously been saved.



Available Histograms

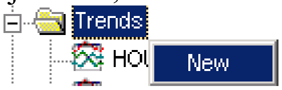
My First Histogram
My Old Histogram
MyHistogram

Setting up a Trend Chart

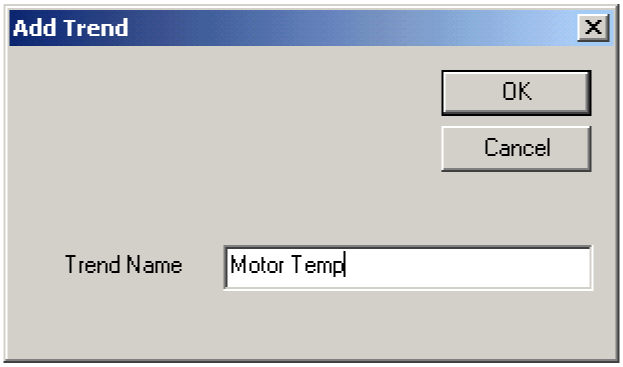
The trend chart allows you to track data over time, as a software 'chart recorder'. The trending chart is good for analog signals, but will work with bits as well (although not as good as the histogram for tracking bit transitions).

In this example, we are going to track and address, N7:7 over time.

1) Right click on Trends in the project tree, and select 'New'.

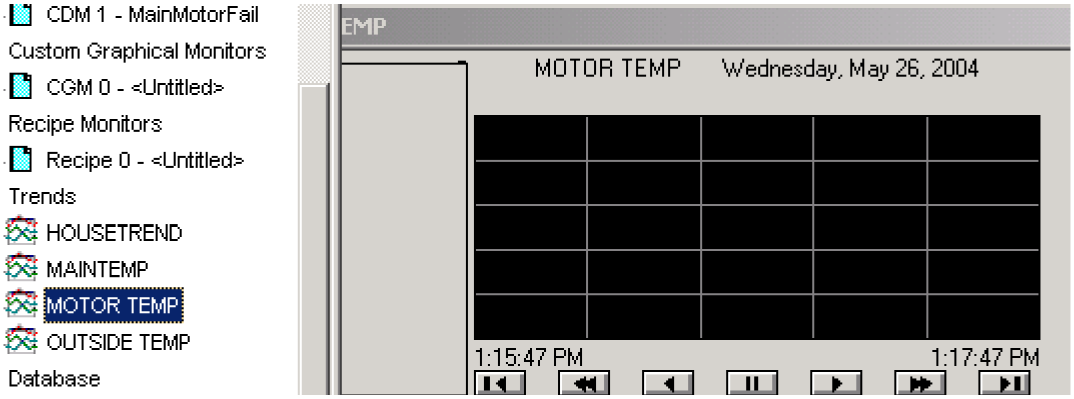


2) Next, you will name your trend. For this example, we will call it motor temp, then press OK.

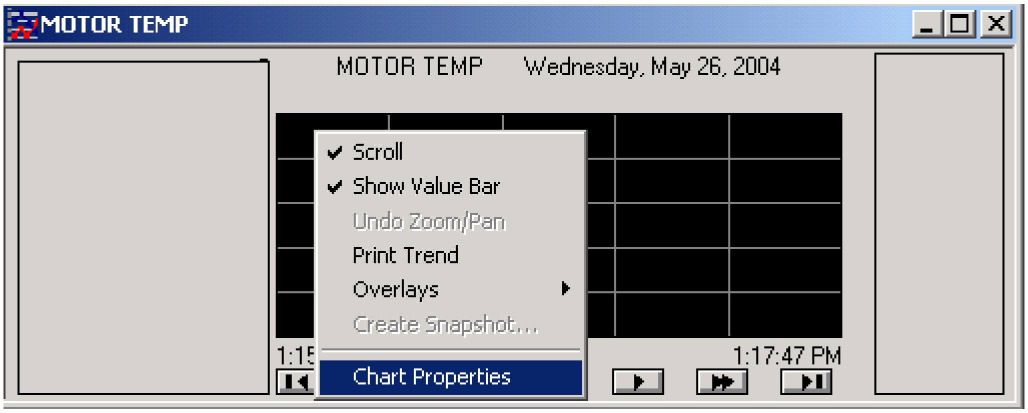


Trend Name: Motor Temp

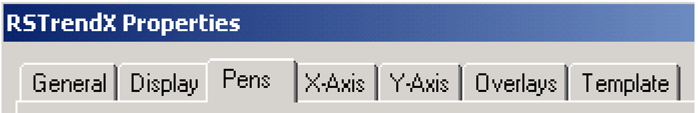
3) Under the trends folder, double click to open the Motor Temp trend chart.



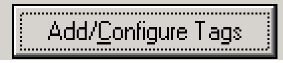
4) Next, a pen must be added to the chart. Right click on the chart to access the chart properties.



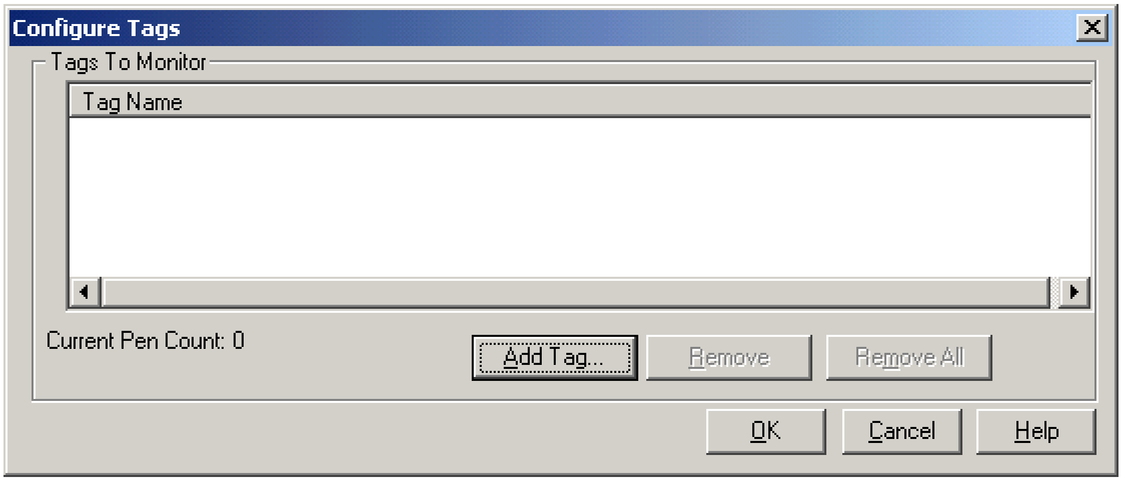
5) Click on the 'Pens' tab in the properties window.



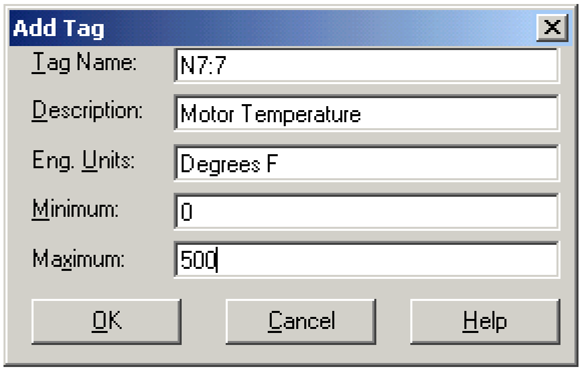
6) Click the Add/Configure Tags button (in the middle of the properties window).



7) Next, the 'Configure Tags' Dialog screen appears. Click 'Add Tag'.



8) The 'Add Tag' Window will be set up as follows for this example. The Tag Name is the actual address you need to track. Also set up the Minimum and Maximum values you would expect at this address. Then we will press OK when finished, then press OK on the Configure Tags window.



Tag Name: N7:7

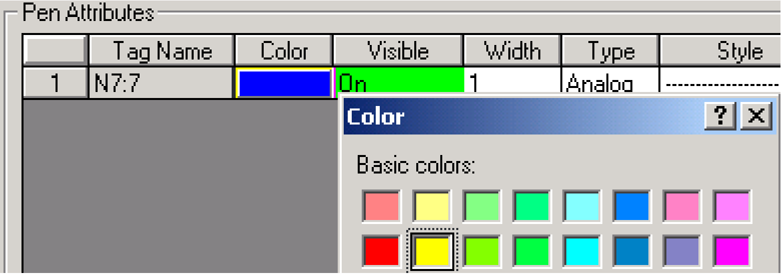
Description: Motor Temperature

Eng. Units: Degrees F

Minimum: 0

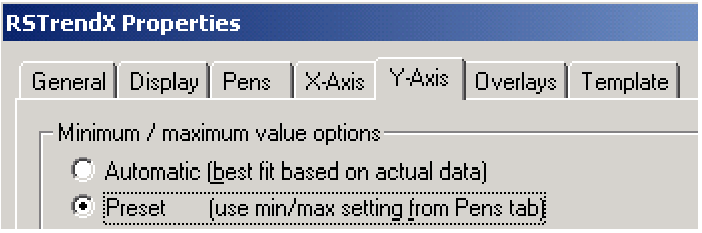
Maximum: 500

9) Next, set the color for your pen. The chart has a black background by default, so we can choose a bright color. To set the color, double click the color box in the pen attributes frame. A pallet will appear. Select a new color, then press OK on the pallet. Next, press Apply then OK at the bottom of the chart properties window.



	Tag Name	Color	Visible	Width	Type	Style
1	N7:7	🟦	On	1	Analog	------------------

10)The chart is now tracking, but the first thing we notice is what appears to be wild fluctuations in temperature. If we look at the scale, however, the temperature is only fluctuating a few degrees. By default, the chart is in automatic mode, so the scale will automatically adjust to the lowest and highest value it sees. To change this scale to the preset values we entered when setting up the tag, right click the chart, and choose chart properties. On the Y Axis tab, and choose 'Preset'. Apply your changes, then press OK.

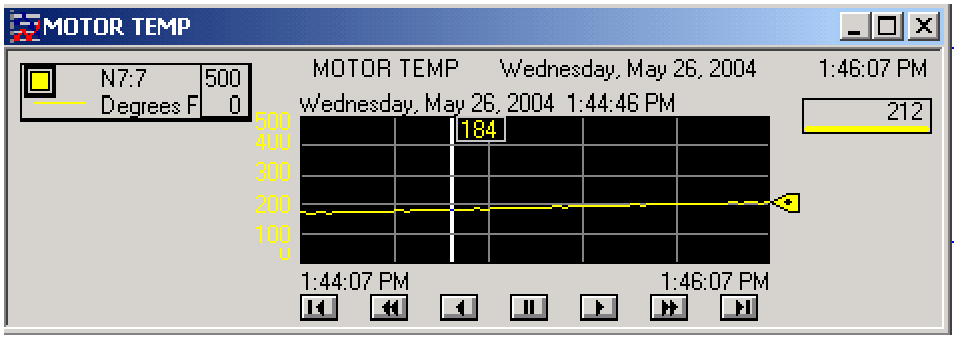


Minimum / maximum value options

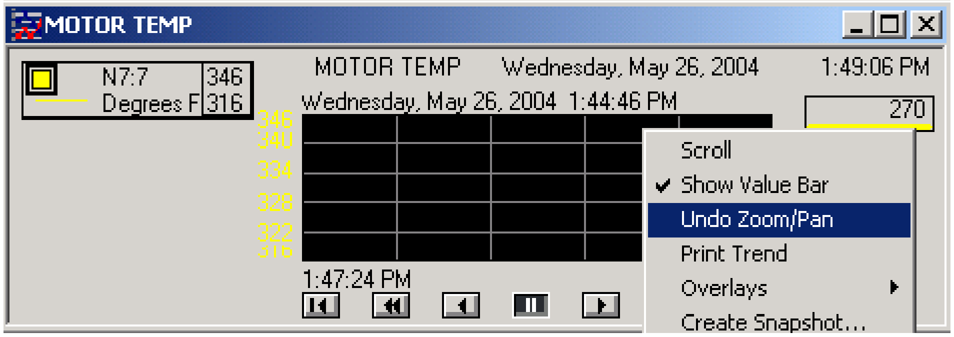
○ Automatic (best fit based on actual data)

● Preset (use min/max setting from Pens tab)

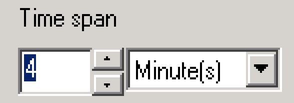
11)You can now see the pen is tracking your data. The motor temperature is slowly increasing over time. To see what the motor temperature was at any point along the graph, click one time on the chart, and a value bar will appear indicating what the value of the pen was at that moment.



12)Be careful. If you accidentally move your mouse while clicking, you will draw a box around a small portion of the chart. When you let up on your mouse button, the chart will zoom in on the area where you clicked. You will notice this by a magnifying glass on the mouse pointer. To get out of this mode, right click on the chart, and Undo Zoom/Pan. Then press the || (pause/scroll) button at the bottom of the chart to begin tracking again.



13)By default the amount of time shown from left to right is 2 minutes. If you wish to change this value, right click on the chart, and choose chart properties. Then click on the X axis tab. The time span can then be modified. When finished, apply your changes, then press OK.

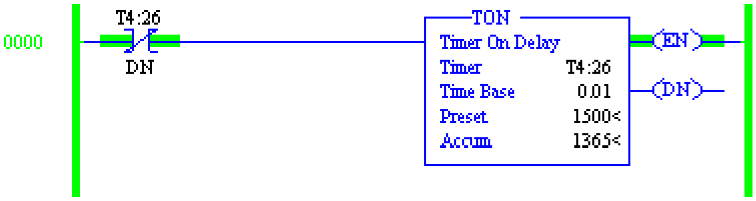


Time span

4 Minute(s)

On Your Own!

Create a trend chart based on any analog signal in your program. If you don't have an analog signal to track, create a self running timer as in the example shown below:



The Tag Name for this example would be T4:26.ACC (The accumulated value of timer 26).

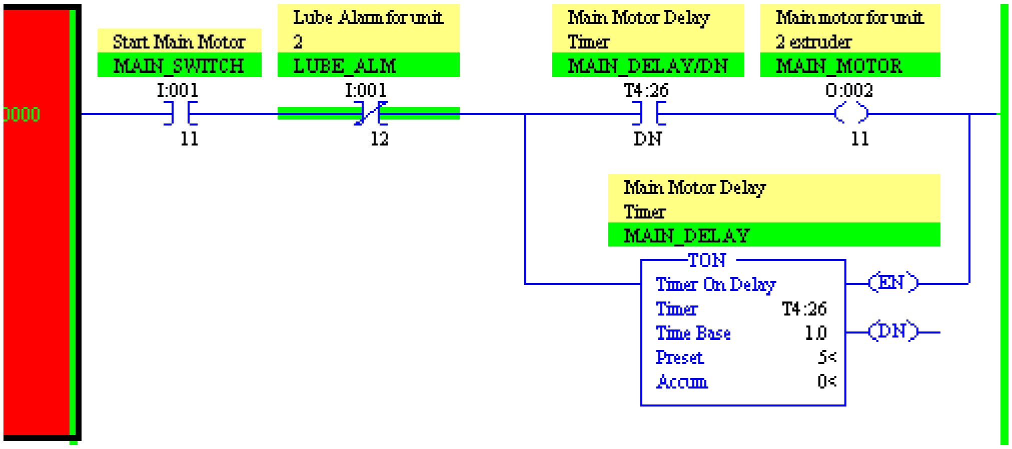
Explore the other tabs in the trending chart. There are many other features that can be customized. Once you get your pen set up, try to change the color of the background under chart properties.

Setting up a Custom Data Monitor

The Custom Data Monitor can significantly reduce downtime by allowing the user of RSLogix to create their own troubleshooting tool. If a pump quits running, a user could bring up the custom data monitor for that particular pump to see what input has not been made, or to see what alarm is locking the pump out just by looking down a list.

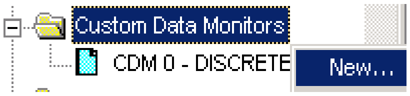
In this example, we will build a custom data monitor which will walk a user through troubleshooting a motor failure.

Look at the simple example below:

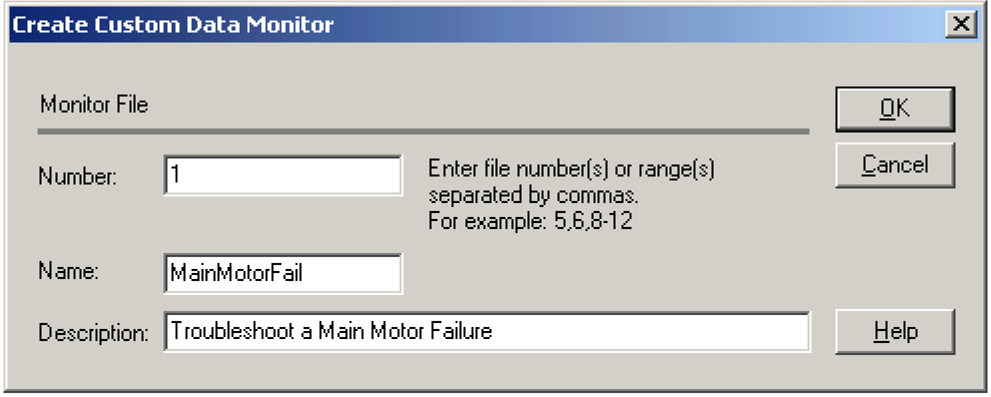


The main motor will only start if the main switch is on, the lube alarm is not on, and then we must wait 10 seconds. Let's build a CDM so we don't have to troubleshoot using ladder logic.

1) In the project tree, right click on the custom data monitor folder, and select 'New'.



2) Let's name and describe the data monitor as shown below, then press OK.



Create Custom Data Monitor

Monitor File

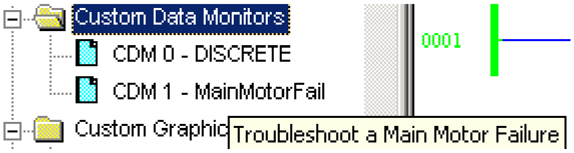
Number: 1

Enter file number(s) or range(s) separated by commas.
For example: 5,6,8-12

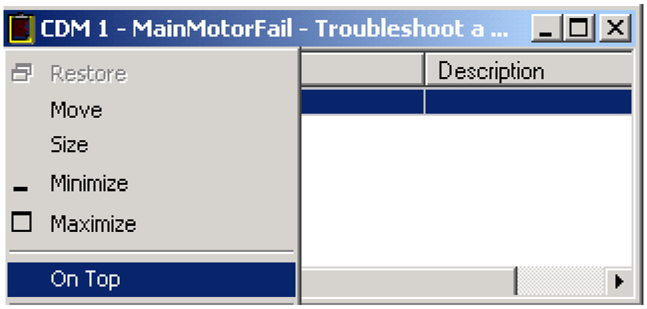
Name: MainMotorFail

Description: Troubleshoot a Main Motor Failure

By using descriptors, you are helping the troubleshooter determine the purpose of the data monitor when he hovers his mouse over the CDM in the project tree. You can create up to 255 data monitors, so descriptions can become very useful.

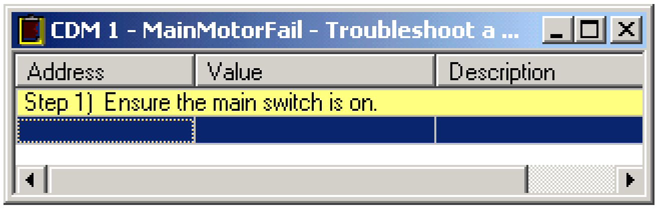


4) Next, double click your new custom data monitor. This will open the CDM, but if a user clicks on the logic window, part of the logic window will cover up the CDM. Usually, we want the CDM to be on top of other windows. To put your data monitor on top of all other windows, click the clipboard in the upper left corner of the CDM, and choose 'On Top'.



We can now set up the CDM!

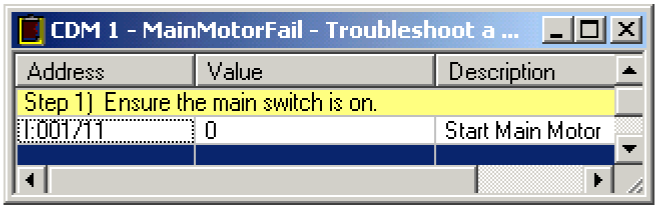
5) First, we will document the first step in troubleshooting. Highlight the first line of the CDM (so it's blue), and then begin to type your text as shown below:



CDM 1 - MainMotorFail - Troubleshoot a ...

Address	Value	Description
Step 1) Ensure the main switch is on.		

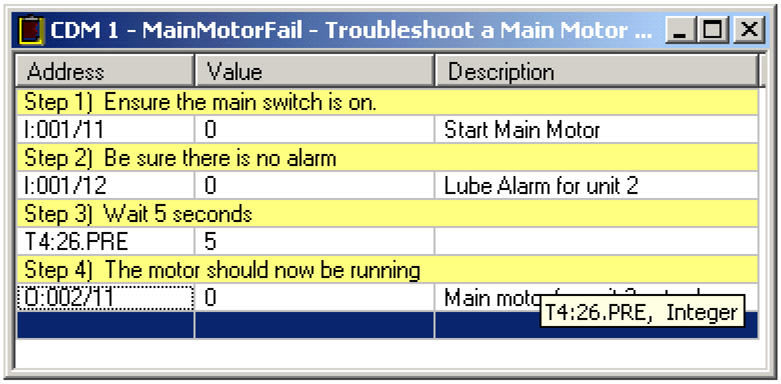
6) Next, we will give the user an address so the value can be monitored. This address can be typed manually, or you can drag this address from the ladder. If you prefer the drag and drop method, be sure not to drop the address on top of your text. The address can be dropped anywhere in the bottom of the CDM (below all other entries), and it will assume the last available row.



CDM 1 - MainMotorFail - Troubleshoot a ...

Address	Value	Description
Step 1) Ensure the main switch is on.		
I:001/11	0	Start Main Motor

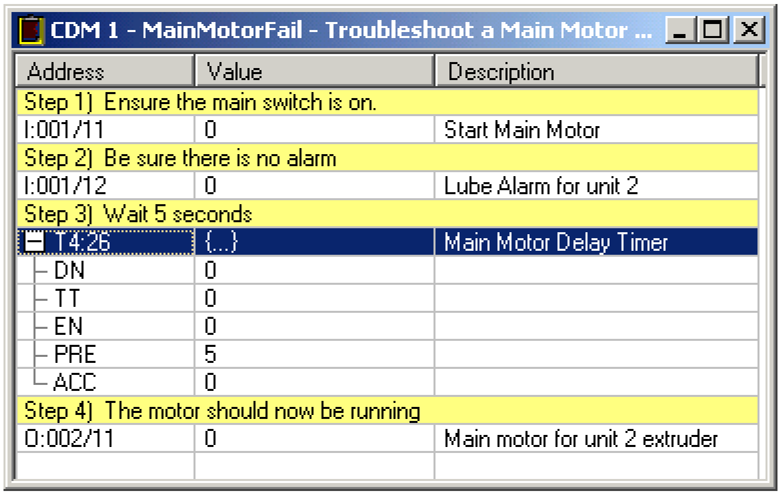
7) Now the switch can be monitored in real time (as long as you are on line). Go ahead and set up the rest of the CDM as shown:



CDM 1 - MainMotorFail - Troubleshoot a Main Motor ...

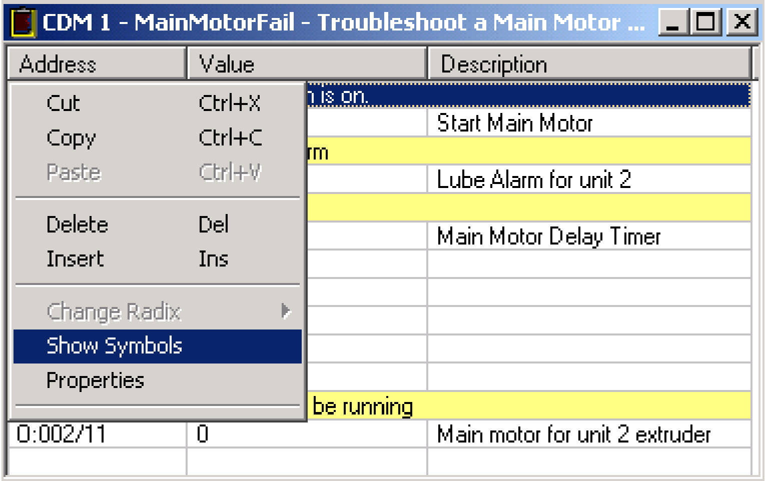
Address	Value	Description
Step 1) Ensure the main switch is on.		
I:001/11	0	Start Main Motor
Step 2) Be sure there is no alarm		
I:001/12	0	Lube Alarm for unit 2
Step 3) Wait 5 seconds		
T4:26.PRE	5	
Step 4) The motor should now be running		
O:002/11	0	Main moto

8) Notice the descriptions were brought in from the database. A user can also change descriptors within the CDM file, and those changes will be entered into the documentation database. Values can also be modified from the custom data monitor such as the timer's preset value. In the above example, just the preset is shown. If you were to drag the entire timer into the CDM, you would see a + next to the timer indicating the timer can be expanded to see the rest of the components. Look below:



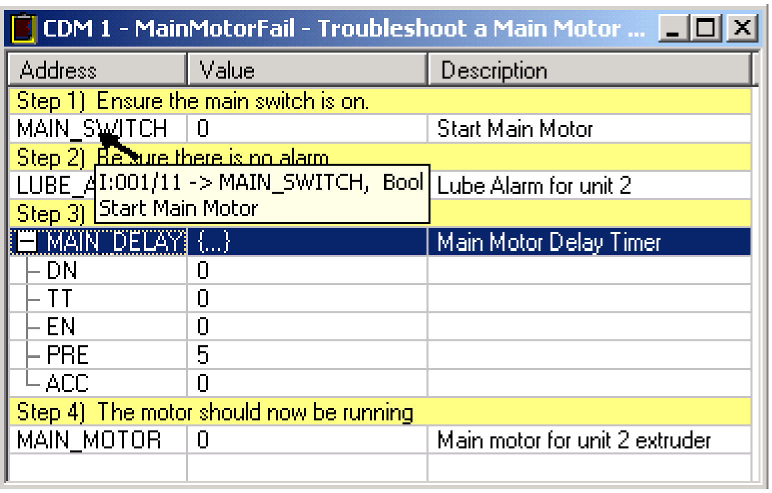
Address	Value	Description
Step 1) Ensure the main switch is on.		
I:001/11	0	Start Main Motor
Step 2) Be sure there is no alarm		
I:001/12	0	Lube Alarm for unit 2
Step 3) Wait 5 seconds		
⊟ T4:26	{...}	Main Motor Delay Timer
├ DN	0	
├ TT	0	
├ EN	0	
├ PRE	5	
└ ACC	0	
Step 4) The motor should now be running		
O:002/11	0	Main motor for unit 2 extruder

9) Addresses such as I:001/11 O:002/11 really don't mean a lot to us. The CDM can be set up to show symbols instead of addresses. To change this setting, right click on the CDM, and select 'Show Symbols'.



Address	Value	Description
	n is on.	
		Start Main Motor
	rm	
		Lube Alarm for unit 2
		Main Motor Delay Timer
	be running	
O:002/11	0	Main motor for unit 2 extruder

10)Now that you are viewing the symbols, you can move your mouse over any of the symbols, and a tool tip will display the address for the symbol. To go back to address mode, right click on the CDM, and choose 'Show Addresses'.



CDM 1 - MainMotorFail - Troubleshoot a Main Motor ...

Address	Value	Description
Step 1] Ensure the main switch is on.		
MAIN_SWITCH	0	Start Main Motor
Step 2] Be sure there is no alarm		
LUBE_A		Lube Alarm for unit 2
Step 3]		
MAIN_DELAY	{...}	Main Motor Delay Timer
DN	0	
TT	0	
EN	0	
PRE	5	
ACC	0	
Step 4] The motor should now be running		
MAIN_MOTOR	0	Main motor for unit 2 extruder

I:001/11 -> MAIN_SWITCH, Bool
Start Main Motor

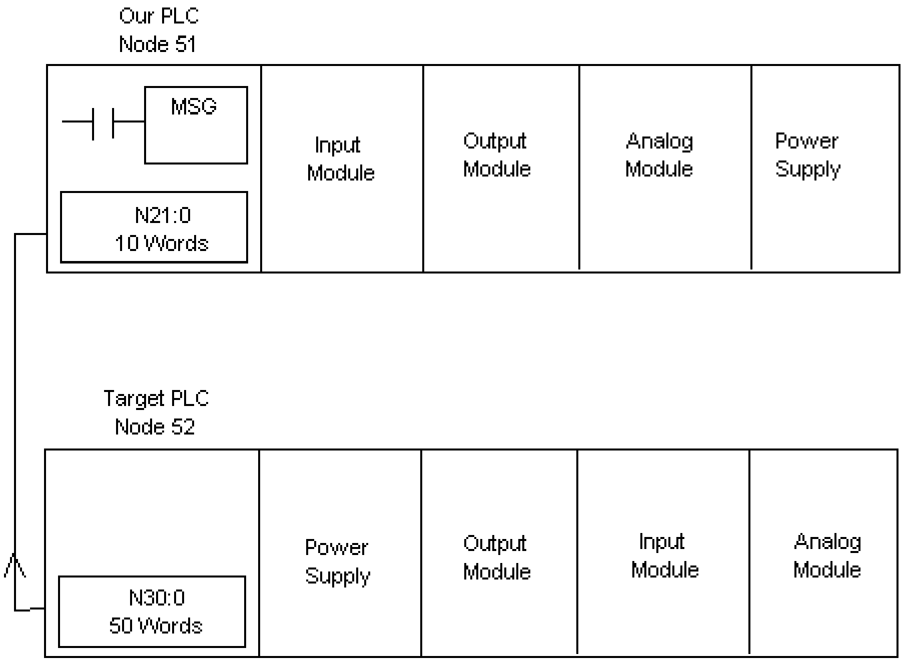
On your own!

In RSLogix, write logic similar to the example shown above. Be sure to use descriptions and symbols to document your project. Create your own CDM for a user to quickly find a problem when the output is not on.

Messaging in the PLC-5

Many times, one PLC needs to communicate with another PLC to share information such as the speed of a drive, or the position of a switch. The message instruction can be used to send or receive this information over several protocols such as DH+, ControlNet, or Ethernet.

In this example, we will set up a message instruction to receive data from another processor on DH+. Our PLC will be instructed to read data from another PLC on the network. Look at the diagram below:



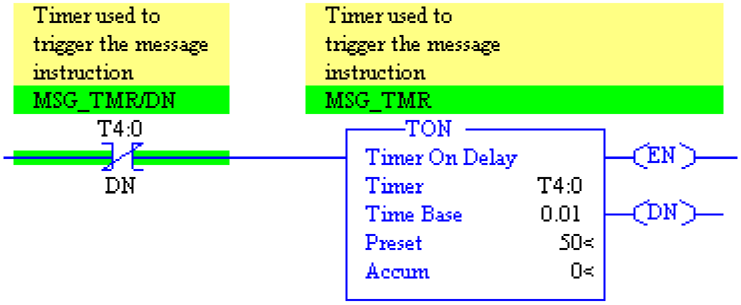
You can see from the diagram what most of the requirements are for the message instruction to operate.

1) We need a memory location (or a range of words) in the **target PLC** that we are reading from.
2) We also must have a place to put this data in **our processor** once we read it from the target. You will notice the file numbers and size of the data tables do not have to match, however since we have only 10 words available in our PLC to store the data, we cannot read more than 10 words from the remote processor in this example.
3) You will also need to know the **node number** of the target PLC. Up to 64 processors can run on this DH+ network, so we must specify which of the processors we want to read data from.

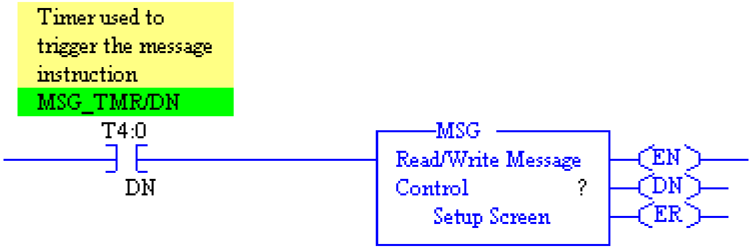
4) We need a way to trigger the message instruction if it's not being run in continuous mode. (This is what the XIC is for before the MSG instruction). For this example, we'll just use a timer to trigger the instruction.
5) We will need a **workspace** for the message instruction to be able to operate, just like we need a workspace for a timer or counter to operate. This workspace will contain status bits such as DN (done), EN(enable), EW(enabled and waiting), etc... The data file we create will have the message (MG) data type (We'll do this later).

Writing the message instruction:

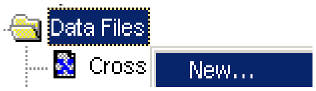
1) First we will add a timer to logic. The purpose of this timer will be to later trigger the message instruction at periodic intervals. This timer will reset itself every time the DN bit goes high. Each time the DN bit goes high, we will trigger the message instruction. Therefore, the lower the preset of this timer, the more frequently the message instruction will be triggered.


Timer used to trigger the message instruction
MSG_TMR/DN
T4:0
DN
Timer used to trigger the message instruction
MSG_TMR
TON
Timer On Delay
Timer T4:0
Time Base 0.01
Preset 50<
Accum 0<
EN
DN

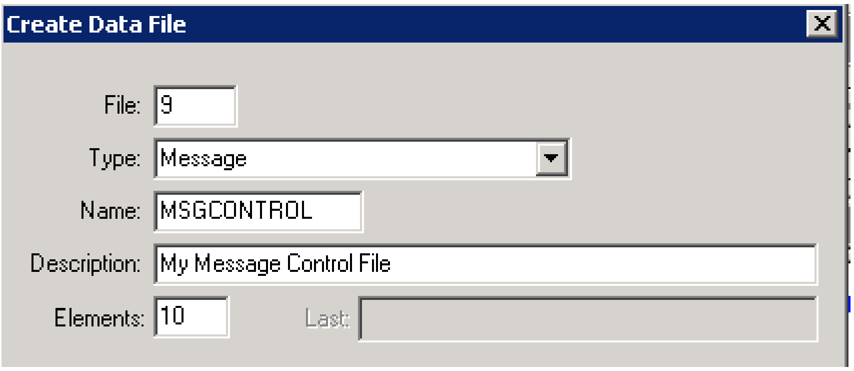
2) Next, add the following rung for the message instruction. In the next few steps, we will discuss how to configure this message instruction.



3) You can see the MSG instruction is asking for a control element. This is the workspace we discussed earlier. This workspace has not been created yet. To create a control file for message instructions, right click on the data files folder in the project tree, and select 'new'.

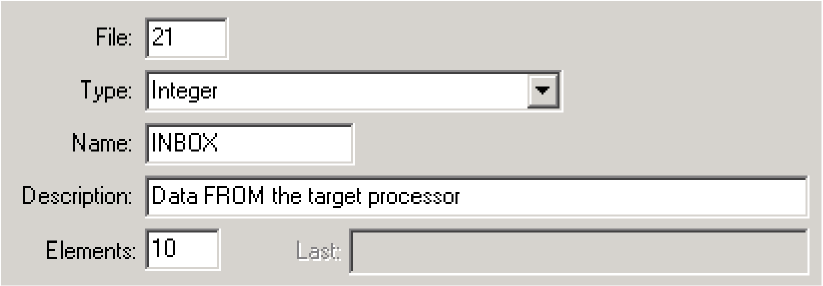


4) File 9 is the next available data file, since 0 through 8 are already in use. You can see this in the project tree. We will use data file 9 to store control elements for message instructions. Since we only have one message instruction, we only need one element in this file, however if you would like to create extra elements for later use, you can do so at this time. We will set up file 9, name it MSGCONTROL, and will have the Message data type. Create 10 elements as shown to allow for more MSG instructions in our project. Press OK when finished.

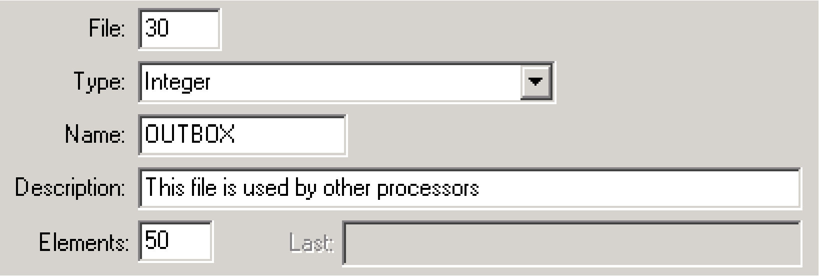

Create Data File

File: 9
Type: Message
Name: MSGCONTROL
Description: My Message Control File
Elements: 10
Last:

5) This would be a good time to create the memory location we are storing data to once we retrieve it from the target processor. As shown earlier, we will make this data file 21. It will be an i(N)teger data type, and it will be 10 elements in length. Right click the data file folder again to create the new data file.

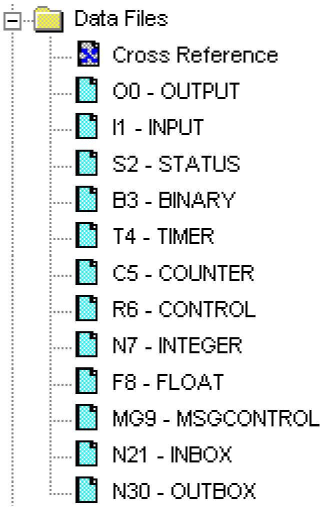

File: 21
Type: Integer
Name: INBOX
Description: Data FROM the target processor
Elements: 10
Last:

6) Let's also create a data file that someone else in the classroom can retrieve data from. The only thing we need to do with this data file is populate it with a value. Another processor will be reading this value. Create the N30 file as shown:



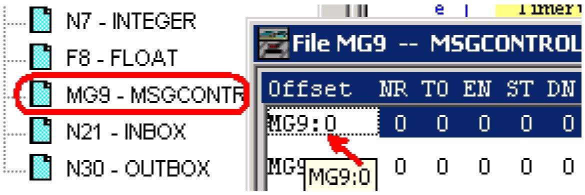
File:	30
Type:	Integer
Name:	OUTBOX
Description:	This file is used by other processors
Elements:	50
Last:	

7) When finished, your data file list will look similar to this:



Data Files
- Cross Reference
- O0 - OUTPUT
- I1 - INPUT
- S2 - STATUS
- B3 - BINARY
- T4 - TIMER
- C5 - COUNTER
- R6 - CONTROL
- N7 - INTEGER
- F8 - FLOAT
- MG9 - MSGCONTROL
- N21 - INBOX
- N30 - OUTBOX

8) Open the MG9 Data File. You will see 10 elements in the MG9 file. Each of these 10 elements can be used to control a message instruction. We will use the first element which is MG9:0.

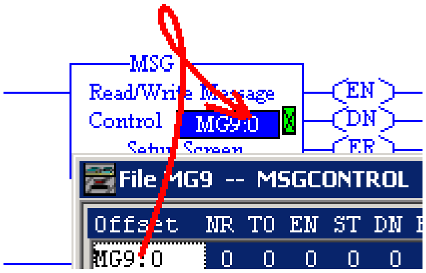


- N7 - INTEGER
- F8 - FLOAT
- MG9 - MSGCONTR
- N21 - INBOX
- N30 - OUTBOX

File MG9 -- MSGCONTROL

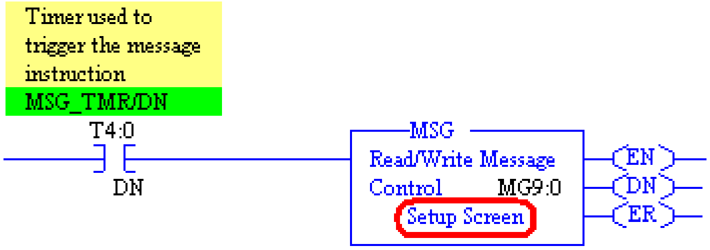
Offset	NR	TO	EN	ST	DN
MG9:0	0	0	0	0	0
MG9	0	0	0	0	0

9) You can drag MG9:0 onto the MSG instruction, or type it manually.

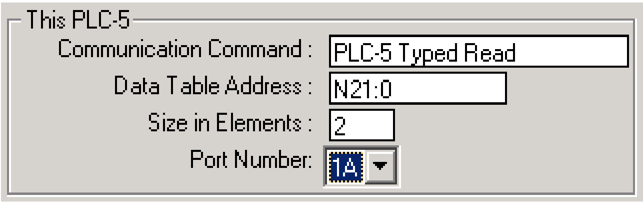


Offset	NR	TO	EN	ST	DN	E
MG9:0	0	0	0	0	0	

10)Next, go to the setup screen to configure the message instruction.



11)First, let's configure 'This PLC'. That is the PLC you are programming right now. Remember that we are wanting to read data from the N30 file in the target, and we created an N21 file in our PLC to store the data in. Let's just copy 2 elements for this example. Since we are doing this over Data Highway Plus, channel 1A will be used. Other channels can be used if data is to be retrieved over other networks such as Ethernet or ControlNet.



This PLC-5

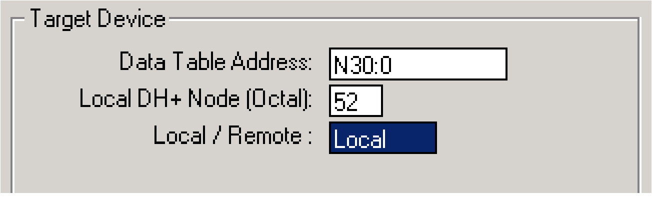
Communication Command : PLC-5 Typed Read

Data Table Address : N21:0

Size in Elements : 2

Port Number: 1A

12)The target processor is the remote processor. Remember from our example that you will be reading from N30:0 in the target processor (the first two elements). The node number will be 52 for this example, and we are accessing this node over the local network, versus going through a bridge to access a processor on a remote network.



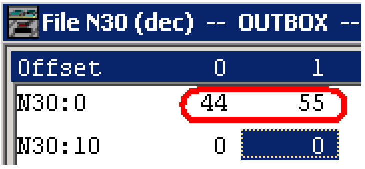
Target Device

Data Table Address: N30:0

Local DH+ Node (Octal): 52

Local / Remote : Local

13)Next, let's populate N30:0 and N30:1 with data so the person reading your station has something to look at. The N30 file has nothing to do with the way your MSG instruction works in your processor. You are only populating these memory locations for another station to read.



File N30 (dec) -- OUTBOX --

Offset	0	1
N30:0	44	55
N30:10	0	0

14)You are ready to download and test your work.

Custom Help Screen

For each application, one customized help screen can be created. This help screen can be in many formats including text, html, and word document format.

Since there is only one custom help screen available per project, this example will use the HTML format. Using HTML, we can create a local web page with links to other documents. This makes this feature very useful.

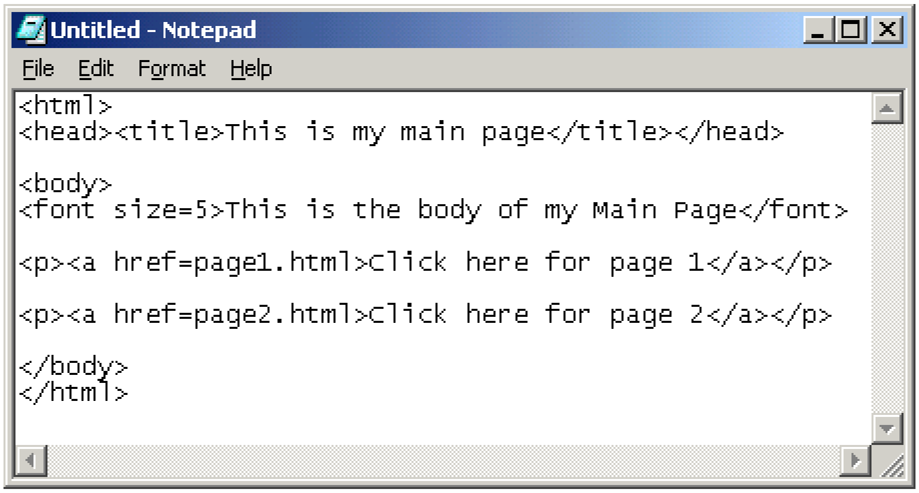
For this example, we will create 3 web pages. A Main page called main.html, and two other pages called page1.html, and page2.html. The user will call up the main page from RSLogix, and from the main page, the user can bring up page 1 or page 2.

1) The first step is to bring up a standard text editor such as windows notepad. This is under Start | Programs | Accessories.

2) Now, you can copy and paste the following code into windows notepad:

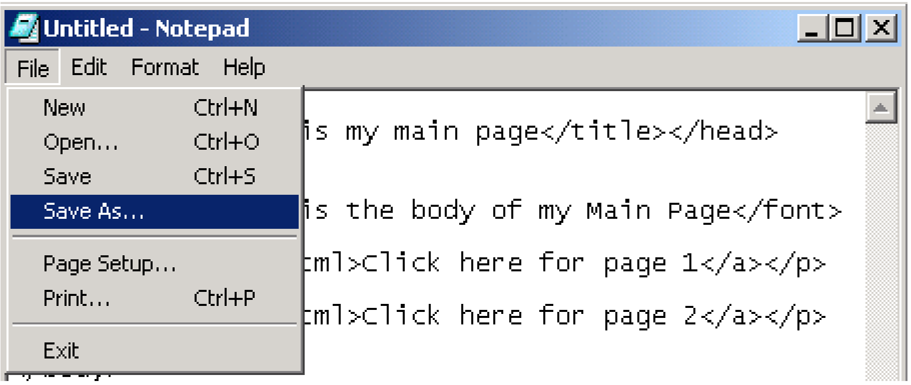
```
<html>
<head><title>This is my main page</title></head>
<body>
<font size=5>This is the body of my Main Page</font>
<p><a href=page1.html>Click here for page 1</a></p>
<p><a href=page2.html>Click here for page 2</a></p>
</body>
</html>
```

Your page will look like this:



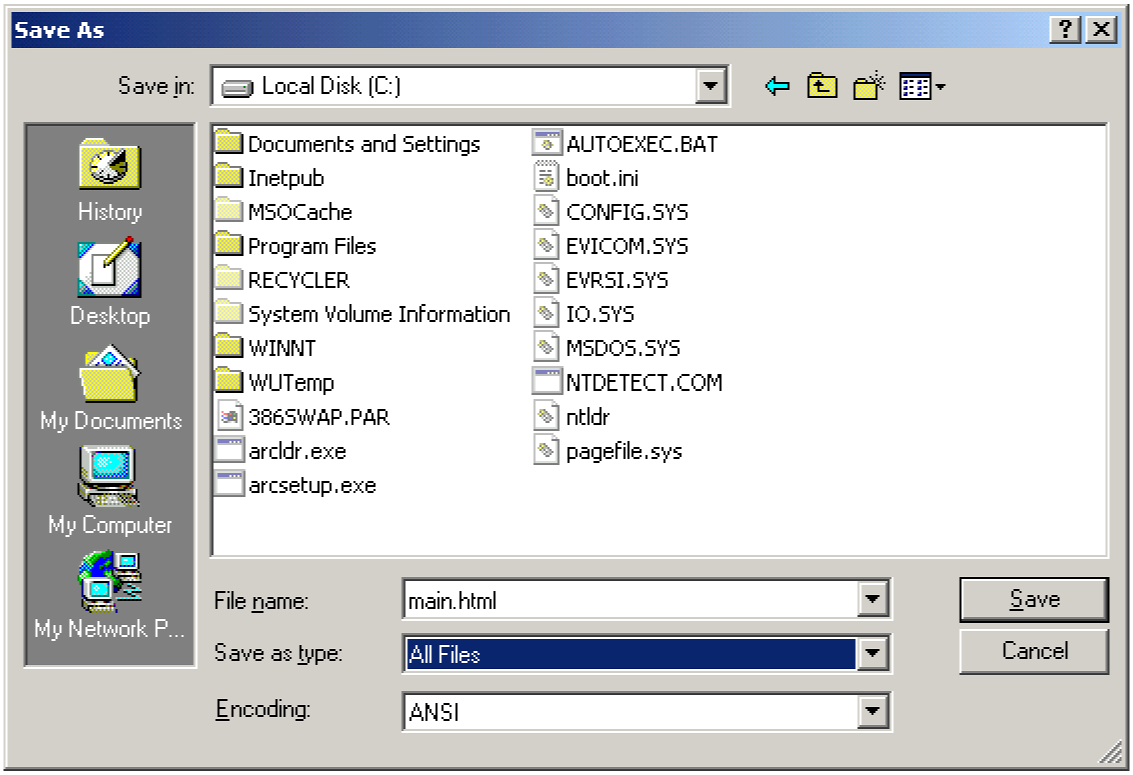
```
<html>
<head><title>This is my main page</title></head>

<body>
<font size=5>This is the body of my Main Page</font>

<p><a href=page1.html>Click here for page 1</a></p>

<p><a href=page2.html>Click here for page 2</a></p>

</body>
</html>
```

3) Next click File | Save As from the menu bar in notepad.



```
is my main page</title></head>

is the body of my Main Page</font>
ml>Click here for page 1</a></p>
ml>Click here for page 2</a></p>
```

4) The file will be called main.html on the main C: drive as shown. Be sure to set the file type as All Files, then save your work.



Save in: Local Disk (C:)

File name: main.html

Save as type: All Files

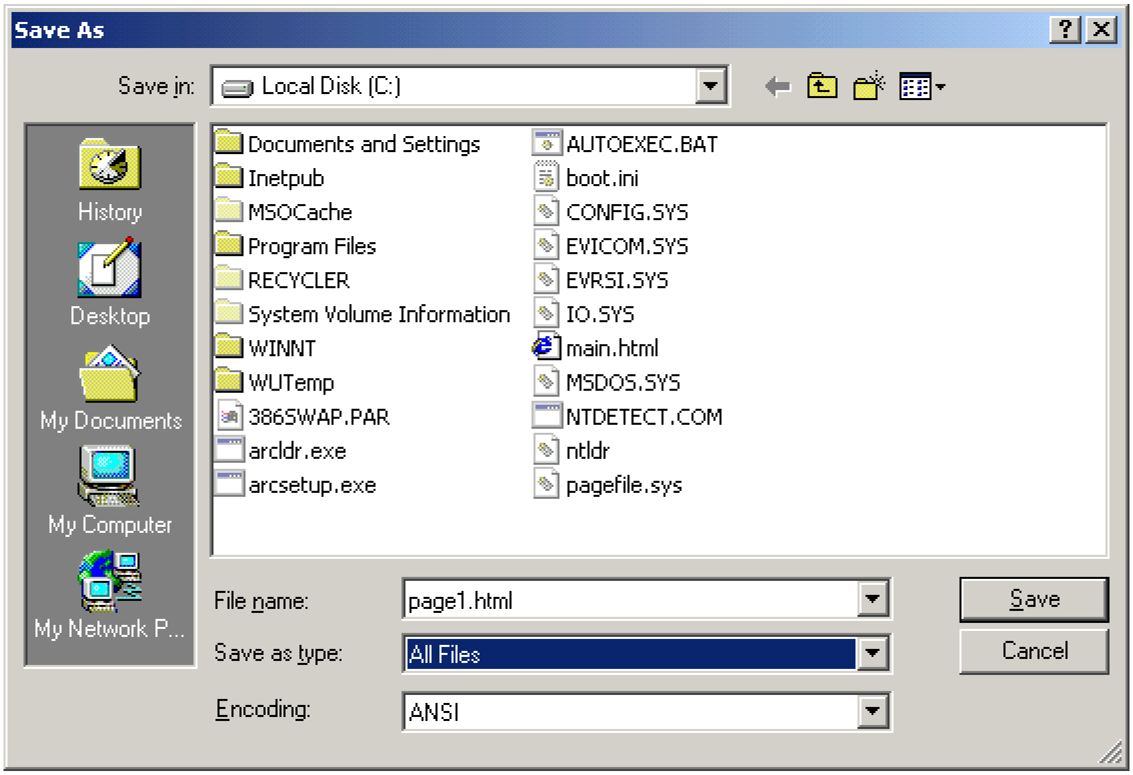
Encoding: ANSI

5) A similar procedure will be followed for Page 1 and Page 2. Click File | New in notepad to create a new document.

6) Paste the following code into notepad:

```
<html>
<head><title>This is Page 1</title></head>
<body>
<font size=5>This is the body of  Page 1</font>
<p><a href=main.html>Click here to go back to main</a></p>
<p><a href=page2.html>Click here for page 2</a></p>
</body>
</html>
```

7) Click File | Save As, to save page1.html to the C: drive. Again be sure to select files off All Types as shown below, and then click the save button.



Save in: Local Disk (C:)

File name: page1.html

Save as type: All Files

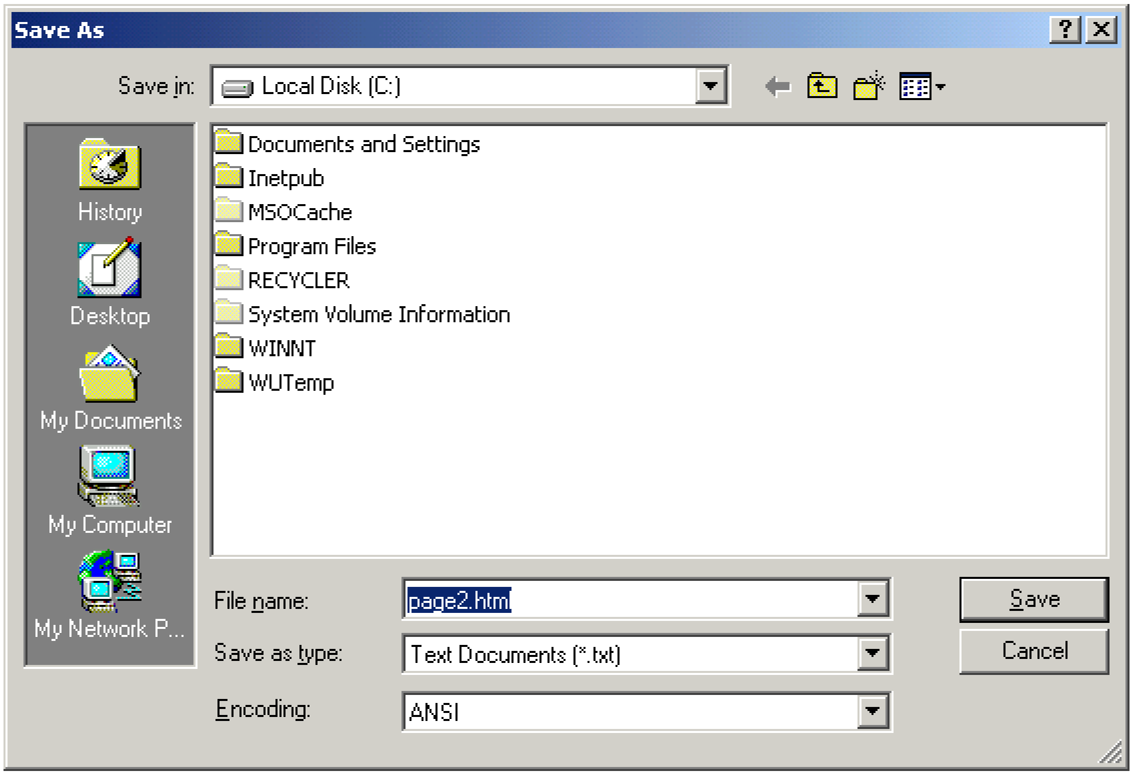
Encoding: ANSI

8) Again, click File | New in notepad

9) Paste the following code into notepad:

```
<html>
<head><title>This is Page 2</title></head>
<body>
<font size=5>This is the body of  Page 2</font>
<p><a href=main.html>Click here to go back to main</a></p>
<p><a href=page1.html>Click here for page 1</a></p>
</body>
</html>
```

10)Click File | Save As, to save page2.html to the C: drive. Again be sure to select files off All Types as shown below, and then click the save button.



File name: page2.html

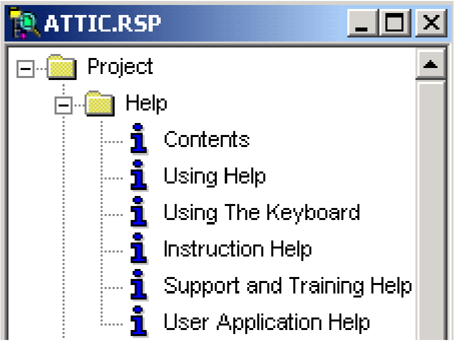
Save as type: Text Documents (*.txt)

Encoding: ANSI

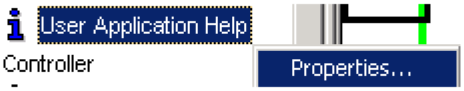
11)You can now go back and modify the content of the pages, or even add pages so they fit the need of your application.

12)Next, RSLogix needs to be configured so the user help feature knows what file to look for. Open RSLogix, and then open the project (.rsp or .rss file) that you wish to link this help information to.

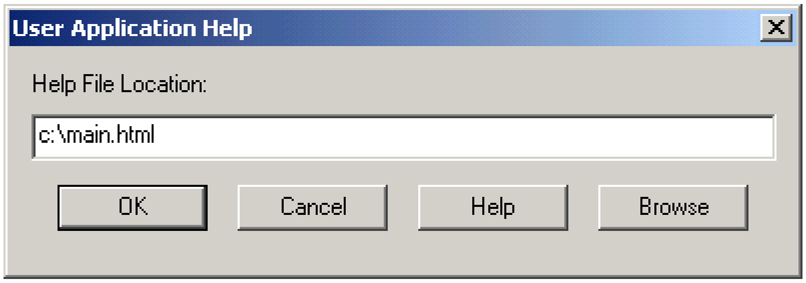
13)In the project tree, expand the help folder as shown:



14)Right click on User Application Help.

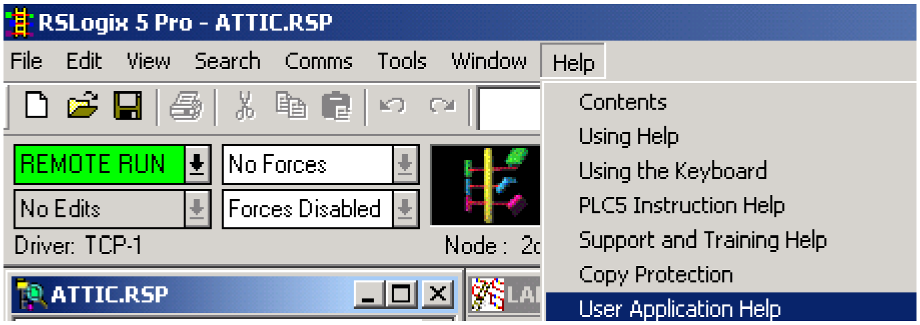


15)You are then asked for the location of the help file. This will be c:\main.html . You can also browse for the help file. Click OK when the help file location is correct.

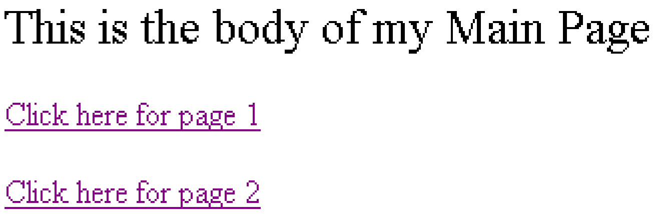


Help File Location: c:\main.html

16)Now you can select user application help either from the project tree, or from the help menu on the menu bar.



17)Your default browser is now launched, and navigation should work correctly.



This is the body of my Main Page

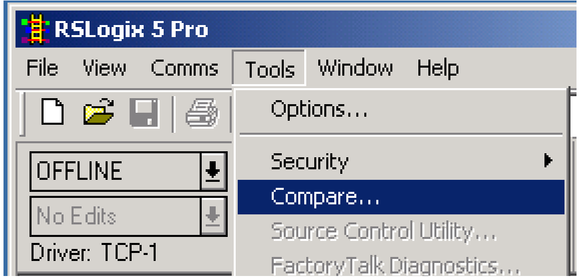
Click here for page 1

Click here for page 2

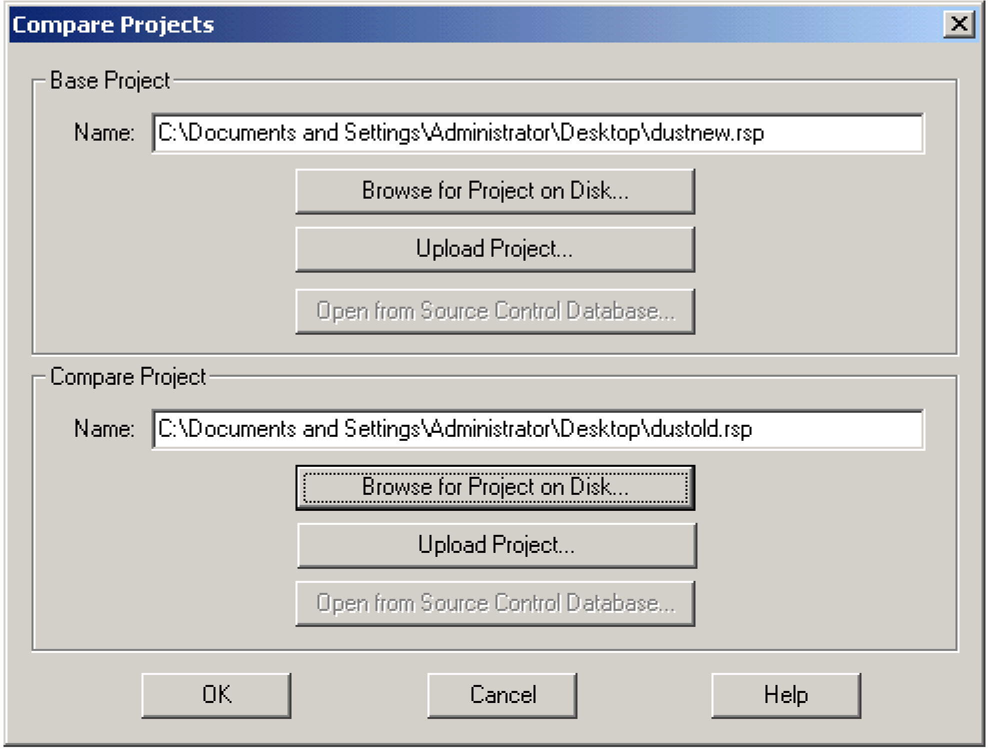
Using the Project Compare Utility

Have you ever wondered if the project on your computer is the same as the project currently running in the PLC? Or have you ever wondered what the differences are between two PLC programs? The Project Compare Utility will help you determine the differences between 1) Two projects on the hard drive, 2) A project on the hard drive against a project in the PLC, or 3) Two projects currently running in different processors.

1) Click Tools | Compare from the menu bar to access the project compare utility. You do not need to have a project open, and any open projects will be closed.



2) Next, you can browse for a project on the hard disk, or upload a project from the PLC for both the Base and Compare projects. For this example, we will compare two files on the hard drive against each other to find the differences. These two files are on the desktop, and are called dustnew.rsp, and dustold.srp (browse for projects)


Compare Projects

Base Project

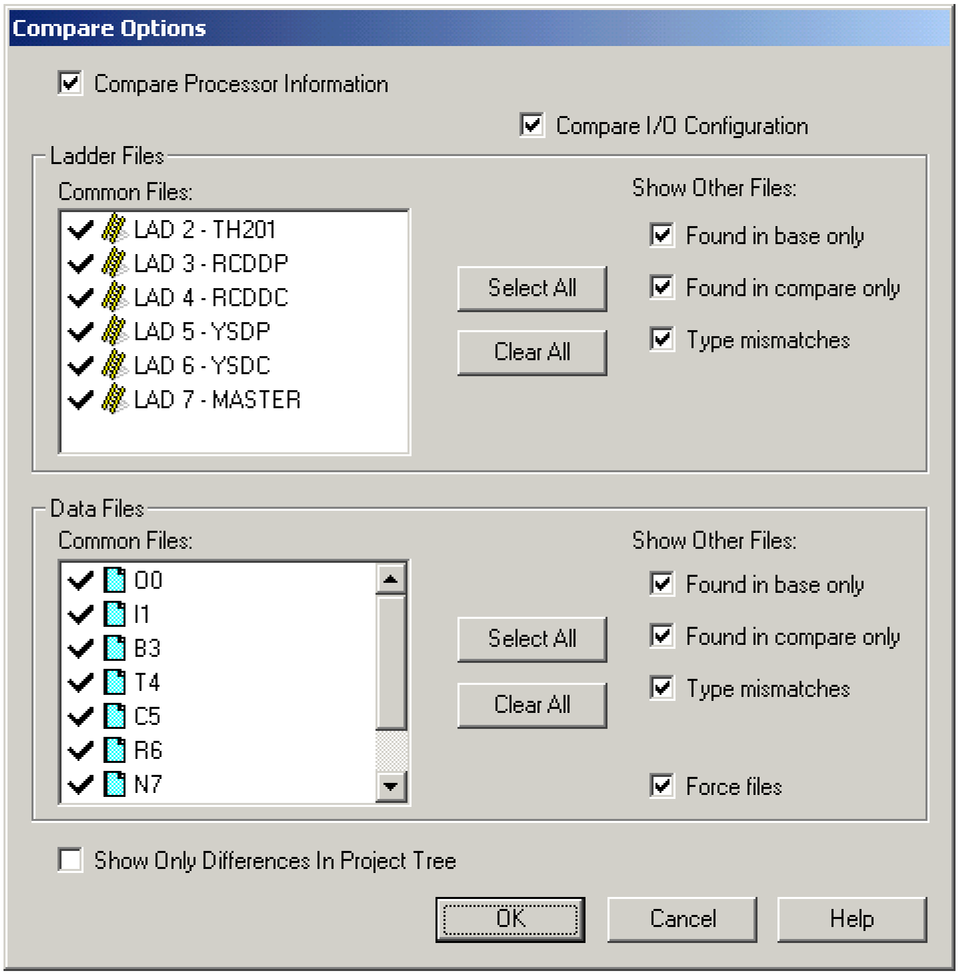
Name: C:\Documents and Settings\Administrator\Desktop\dustnew.rsp

Compare Project

Name: C:\Documents and Settings\Administrator\Desktop\dustold.rsp

Browse for Project on Disk...
Upload Project...
Open from Source Control Database...
Browse for Project on Disk...
Upload Project...
Open from Source Control Database...
OK
Cancel
Help

3) Next, the compare utility needs to know what you want to compare. For this example, we will leave this at default to compare everything. You can also check a box at the bottom of the Compare Options screen so only the differences are shown in the results. It is normal for data files to differ between projects, because the state of the I/O, Timers and Counters could be different depending on what part of the machine cycle the program was uploaded. On this screen just press OK.



Compare Options

☑ Compare Processor Information

☑ Compare I/O Configuration

Ladder Files

Common Files:

- ✓ LAD 2 - TH201
- ✓ LAD 3 - RCDDP
- ✓ LAD 4 - RCDDC
- ✓ LAD 5 - YSDP
- ✓ LAD 6 - YSDC
- ✓ LAD 7 - MASTER

Show Other Files:

☑ Found in base only

☑ Found in compare only

☑ Type mismatches

Data Files

Common Files:

- ✓ O0
- ✓ I1
- ✓ B3
- ✓ T4
- ✓ C5
- ✓ R6
- ✓ N7

Show Other Files:

☑ Found in base only

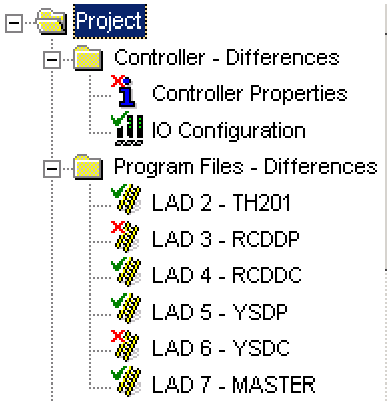
☑ Found in compare only

☑ Type mismatches

☑ Force files

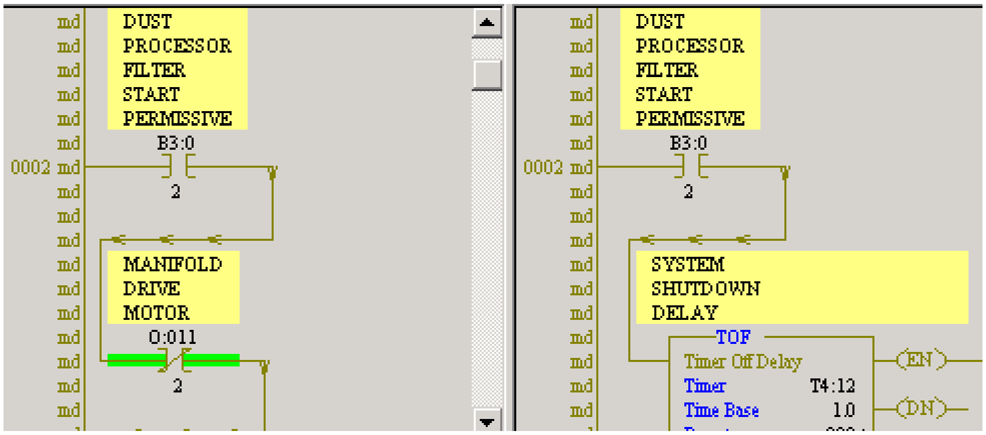
□ Show Only Differences In Project Tree

5) In the image below, you can clearly see what ladders have discrepancies between the two projects. Ladder 3 and ladder 6. There are also some differences in the controller properties. Let's double click ladder 3.



6) The ladders are then placed side by side. Click the blue down arrow to navigate to the first discrepancy.

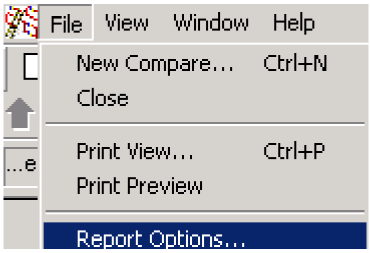
7) Look in the margin, and you will see the letters md. A legend at the top of the window tells us that md has been modified. Can you find the difference between the two rungs? An extra bit has been added to the new project.



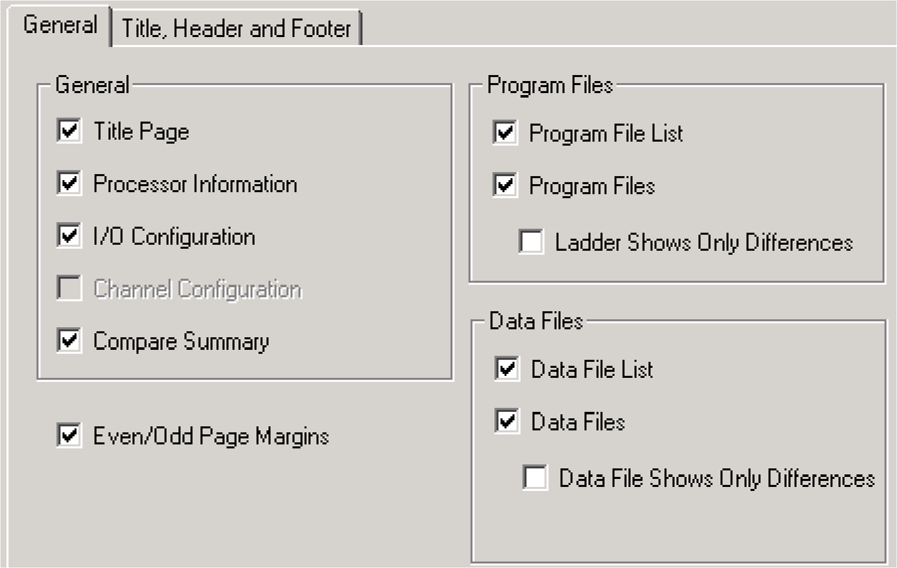
8) To navigate up or down through discrepancies between the two ladders, continue to click the blue up or down arrows. To look at another ladder, close out of the window that is open within the compare utility. You will be taken back to the project tree.

Generating Compare Reports

9) Compare reports can be generated in a similar way that project reports are generated. To access the report options, click File | Report options from the menu bar.



10)On the General tab, select the reports you would like to print.



General

☑ Title Page

☑ Processor Information

☑ I/O Configuration

☐ Channel Configuration

☑ Compare Summary

☑ Even/Odd Page Margins

Program Files

☑ Program File List

☑ Program Files

☐ Ladder Shows Only Differences

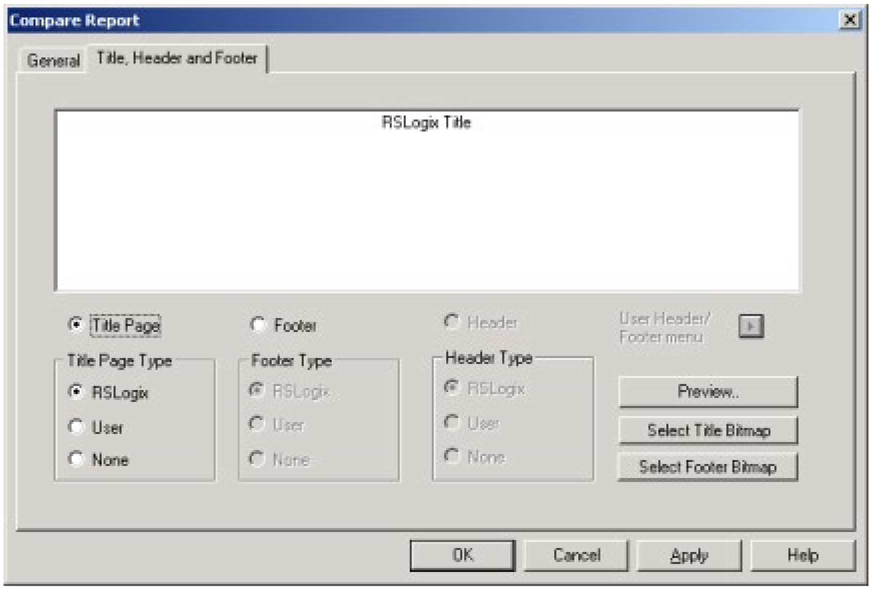
Data Files

☑ Data File List

☑ Data Files

☐ Data File Shows Only Differences

11)Click on the Title, Header, and Footer tab. On this screen you can configure your text and bitmaps for the report. Remember, if you need to customize one of these options, be sure to select User mode.



Compare Report

General | Title, Header and Footer

RSLogix Title

● Title Page ○ Footer ○ Header

Title Page Type: ● RSLogix ○ User ○ None

Footer Type: ● RSLogix ○ User ○ None

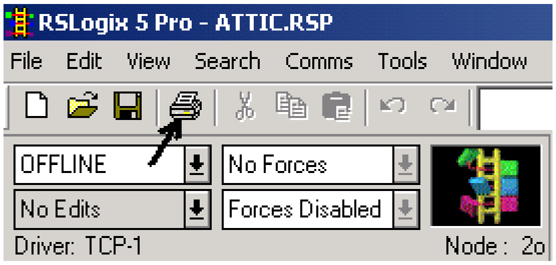
Header Type: ● RSLogix ○ User ○ None

12)Click the preview to see what your final report will look like.

Printing Reports

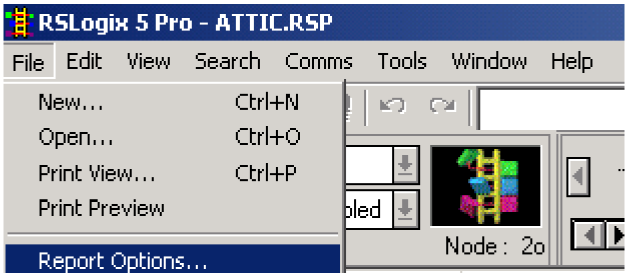
Reports can be a useful troubleshooting tool for those who do not have access to RSLogix, or those who prefer to troubleshoot from a hard copy of the program. Before reports can be printed, the user must first tell RSLogix what reports are to be printed (such as ladder logic or data files). After the reports are selected, each report can be given special options (such as what ladders or rungs to print in a program file). Be aware that even the smallest project can use a lot of paper if all reports are selected.

Note: If you are just wanting to print the current ladder, or a current data file, a report does not have to be configured. Just use the print view feature (File | Print View), or click the print ladder icon on the standard tool bar.

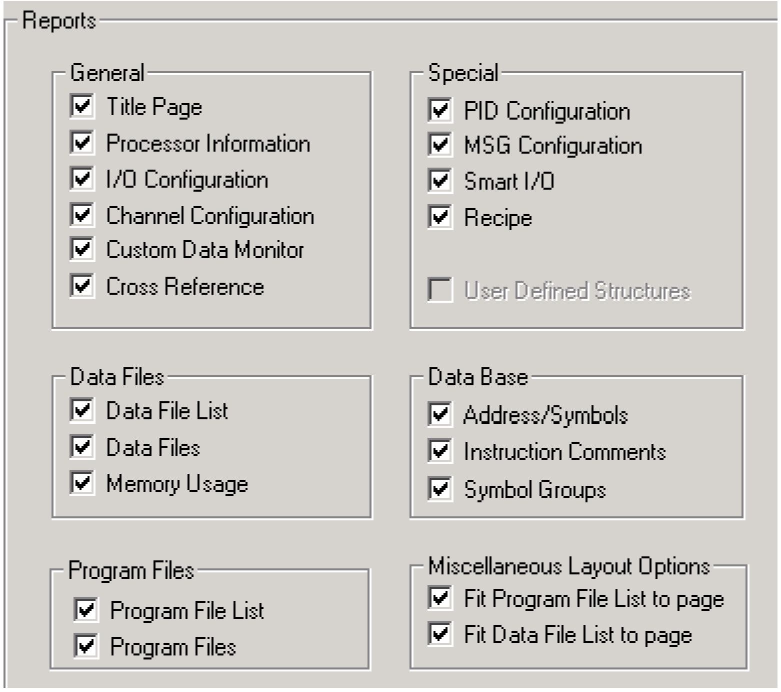

RSLogix 5 Pro - ATTIC.RSP
File Edit View Search Comms Tools Window
OFFLINE
No Forces
No Edits
Forces Disabled
Driver: TCP-1
Node : 2o

Note: Reports may be configured while on line, but in order to print or preview a report, you must be off line so data table values are not changing.

1) Click File | Report options from the menu bar.



2) Next, select what reports you would like to print. For this example, select all reports. (We will just preview the reports instead of actually printing them.)



Reports

General
☑ Title Page
☑ Processor Information
☑ I/O Configuration
☑ Channel Configuration
☑ Custom Data Monitor
☑ Cross Reference

Special
☑ PID Configuration
☑ MSG Configuration
☑ Smart I/O
☑ Recipe
□ User Defined Structures

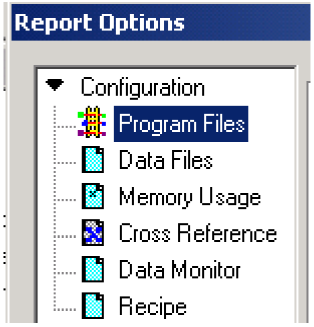
Data Files
☑ Data File List
☑ Data Files
☑ Memory Usage

Data Base
☑ Address/Symbols
☑ Instruction Comments
☑ Symbol Groups

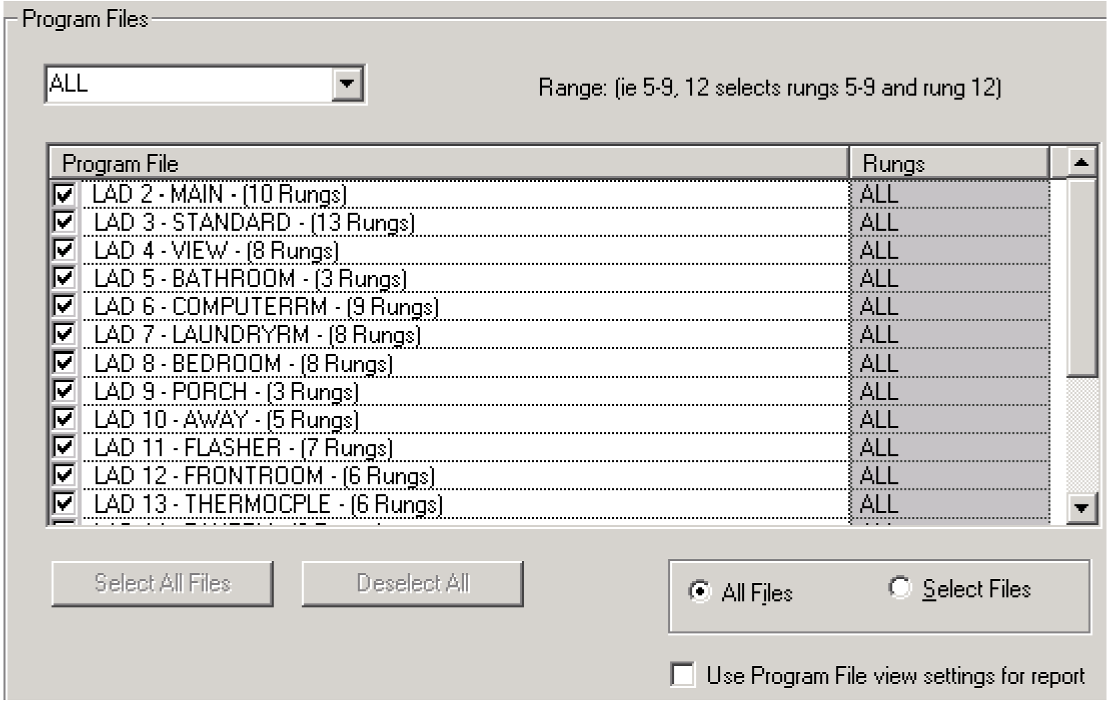
Program Files
☑ Program File List
☑ Program Files

Miscellaneous Layout Options
☑ Fit Program File List to page
☑ Fit Data File List to page

3) Now that all reports have been selected, let's look at the options that are available for each individual report. On the left side of report options, you will find a configuration tree. Click on Program files to see what options are available.



4) By default, all rungs on all ladders will be printed. If you wish to select only certain programs,



Program Files

ALL ▾

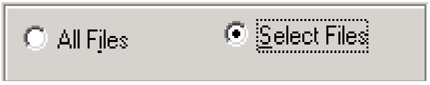
Range: (ie 5-9, 12 selects rungs 5-9 and rung 12)

Program File	Rungs
☑ LAD 2 - MAIN - (10 Rungs)	ALL
☑ LAD 3 - STANDARD - (13 Rungs)	ALL
☑ LAD 4 - VIEW - (8 Rungs)	ALL
☑ LAD 5 - BATHROOM - (3 Rungs)	ALL
☑ LAD 6 - COMPUTERRM - (9 Rungs)	ALL
☑ LAD 7 - LAUNDRYRM - (8 Rungs)	ALL
☑ LAD 8 - BEDROOM - (8 Rungs)	ALL
☑ LAD 9 - PORCH - (3 Rungs)	ALL
☑ LAD 10 - AWAY - (5 Rungs)	ALL
☑ LAD 11 - FLASHER - (7 Rungs)	ALL
☑ LAD 12 - FRONTROOM - (6 Rungs)	ALL
☑ LAD 13 - THERMOCPLE - (6 Rungs)	ALL

● All Files ○ Select Files

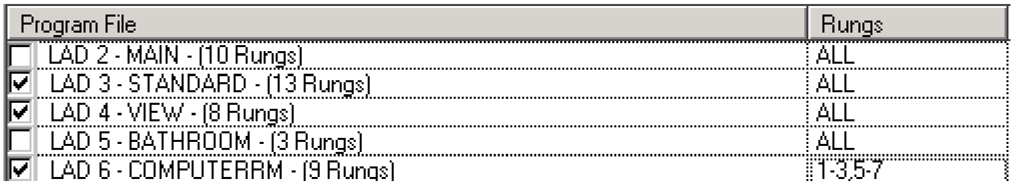
□ Use Program File view settings for report

If you wish to print only certain ladders, you must choose the 'Select Files' button.



○ All Files ● Select Files

You can then select which ladders you wish to print. In the example below, Ladders 3 and 4 will be printed, and rungs 1,2,3,5,6, and 7 will be printed of Ladder 6.



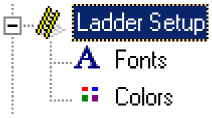
Program File	Rungs
□ LAD 2 - MAIN - (10 Rungs)	ALL
☑ LAD 3 - STANDARD - (13 Rungs)	ALL
☑ LAD 4 - VIEW - (8 Rungs)	ALL
□ LAD 5 - BATHROOM - (3 Rungs)	ALL
☑ LAD 6 - COMPUTERRM - (9 Rungs)	1-3,5-7

You also have the ability to use the program's view settings for the report (found under View | properties). This feature allows you to change the way the ladder is displayed on your screen such as comments, colors, fonts, and bit display mode. Check this box if you want the printout to reflect your view settings.

□ Use Program File view settings for report

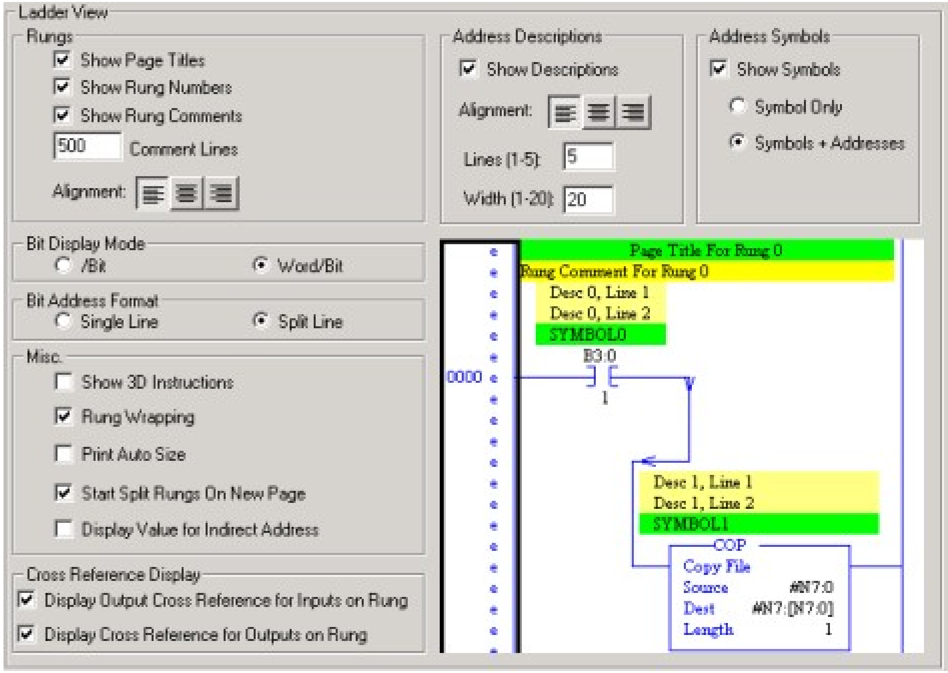
If this box is not checked, the report settings can be different than the view settings. If the colors in your view settings will not print well on a black and white printer, it might be a good idea to configure the print settings differently.

To change the ladder print settings, click 'Ladder Setup' in the configuration tree.



Ladder Setup
- Fonts
- Colors

Now, configure how you want the ladder to print. The colors and fonts can be changed as well. Remember, this screen is only used if you are not using the view settings for the report.



Ladder View

Rungs
- ☑ Show Page Titles
- ☑ Show Rung Numbers
- ☑ Show Rung Comments
- 500 Comment Lines
- Alignment:

Address Descriptions
- ☑ Show Descriptions
- Alignment:
- Lines (1-5): 5
- Width (1-20): 20

Address Symbols
- ☑ Show Symbols
- ○ Symbol Only
- ● Symbols + Addresses

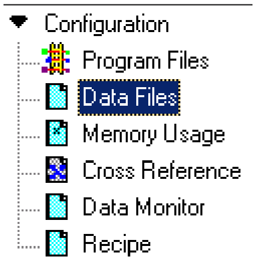
Bit Display Mode
- ○ /Bit
- ● Word/Bit

Bit Address Format
- ○ Single Line
- ● Split Line

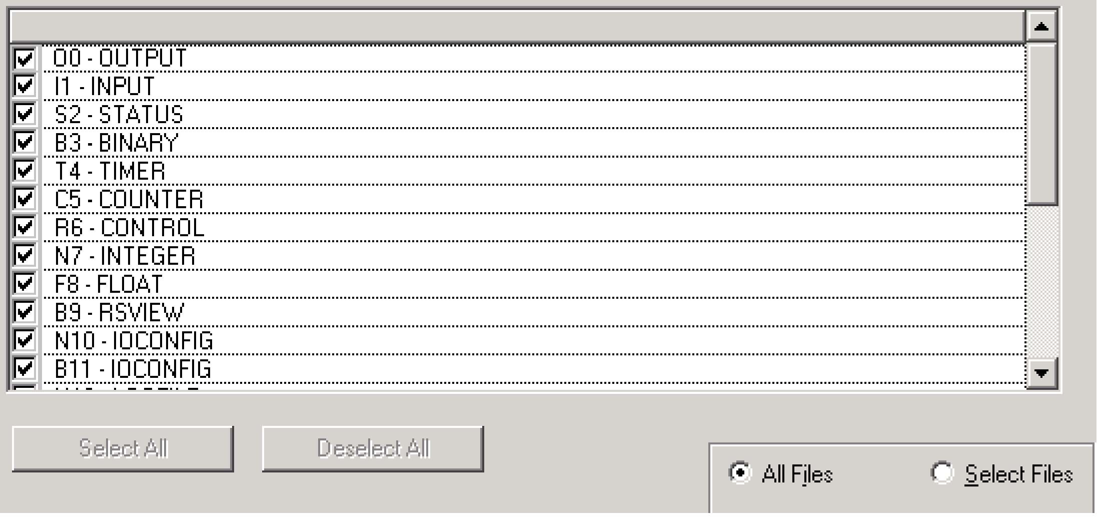
Misc.
- □ Show 3D Instructions
- ☑ Rung Wrapping
- □ Print Auto Size
- ☑ Start Split Rungs On New Page
- □ Display Value for Indirect Address

Cross Reference Display
- ☑ Display Output Cross Reference for Inputs on Rung
- ☑ Display Cross Reference for Outputs on Rung

5) Next, Click 'Data Files' under Configuration.



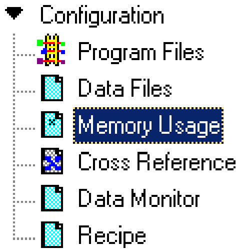
6) Configuration of the data files report is very similar to the program files. By default, all data files are printed. If you want only certain files to print, choose 'select files', and put a check mark next to the files you wish to print.



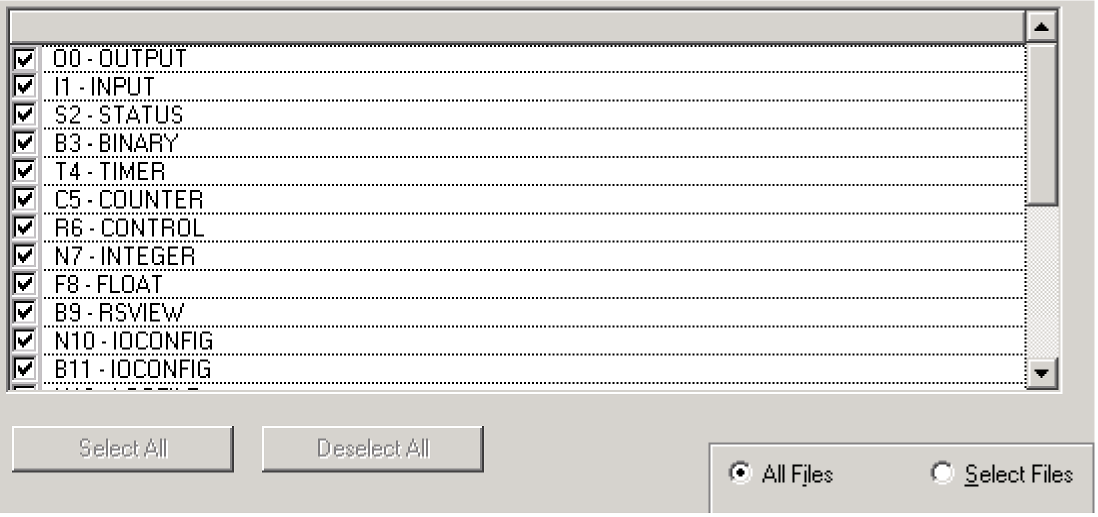
☑ O0 - OUTPUT
☑ I1 - INPUT
☑ S2 - STATUS
☑ B3 - BINARY
☑ T4 - TIMER
☑ C5 - COUNTER
☑ R6 - CONTROL
☑ N7 - INTEGER
☑ F8 - FLOAT
☑ B9 - RSVIEW
☑ N10 - IOCONFIG
☑ B11 - IOCONFIG

● All Files ○ Select Files

7) Next is the memory usage report. Occasionally, an RSLogix user will inadvertently specify a timer such as T4:999, when he actually meant to specify T4:99. RSLogix will automatically expand the T4 file to include a thousand timers. This is a waste of memory. The memory usage report will you locate files which are unnecessarily using a lot of memory.



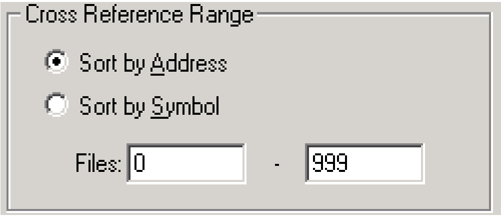
Configuration of the memory usage report is very similar to that of the data file report. By default, all files are selected. To select only certain files, choose 'select files', and put a pockmarked next to the files you want included in the report.



☑ O0 - OUTPUT
☑ I1 - INPUT
☑ S2 - STATUS
☑ B3 - BINARY
☑ T4 - TIMER
☑ C5 - COUNTER
☑ R6 - CONTROL
☑ N7 - INTEGER
☑ F8 - FLOAT
☑ B9 - RSVIEW
☑ N10 - IOCONFIG
☑ B11 - IOCONFIG

● All Files ○ Select Files

8) For the cross-reference report, you can sort by symbols, or sort by addresses. If you aren't using many symbols in your project, you may want to leave this setting at default (sort by addresses). The cross reference report will help the user navigate through logic by showing every location in logic each address is used.

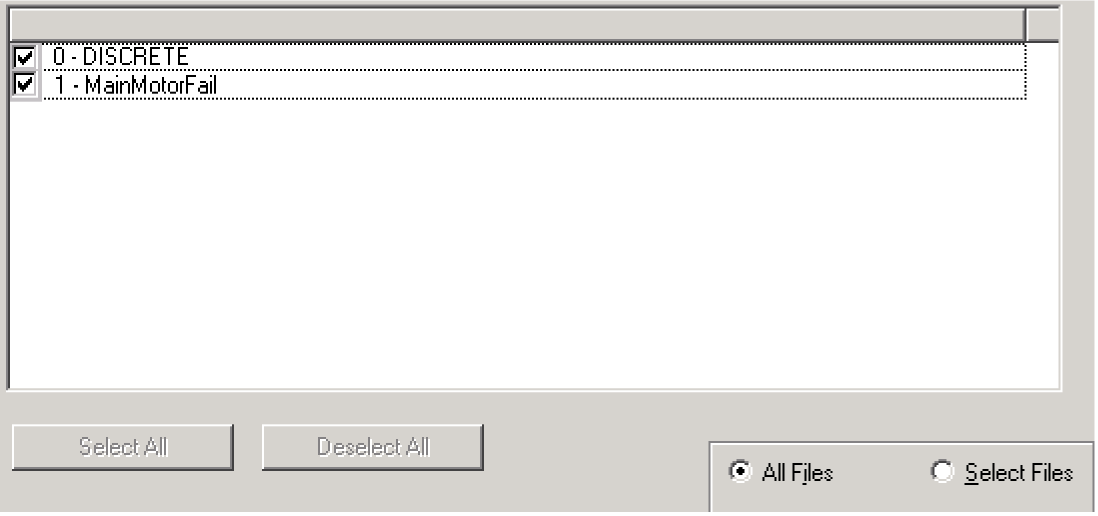


Cross Reference Range

● Sort by Address
○ Sort by Symbol

Files: 0 - 999

9) Custom Data Monitors can be used as troubleshooting tools when on line with the processor. Although a printout of a custom data monitor would have limited use, you do have the option to include CDM files in your report.



☑ 0 - DISCRETE
☑ 1 - MainMotorFail

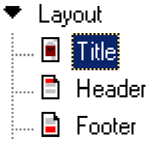
● All Files ○ Select Files

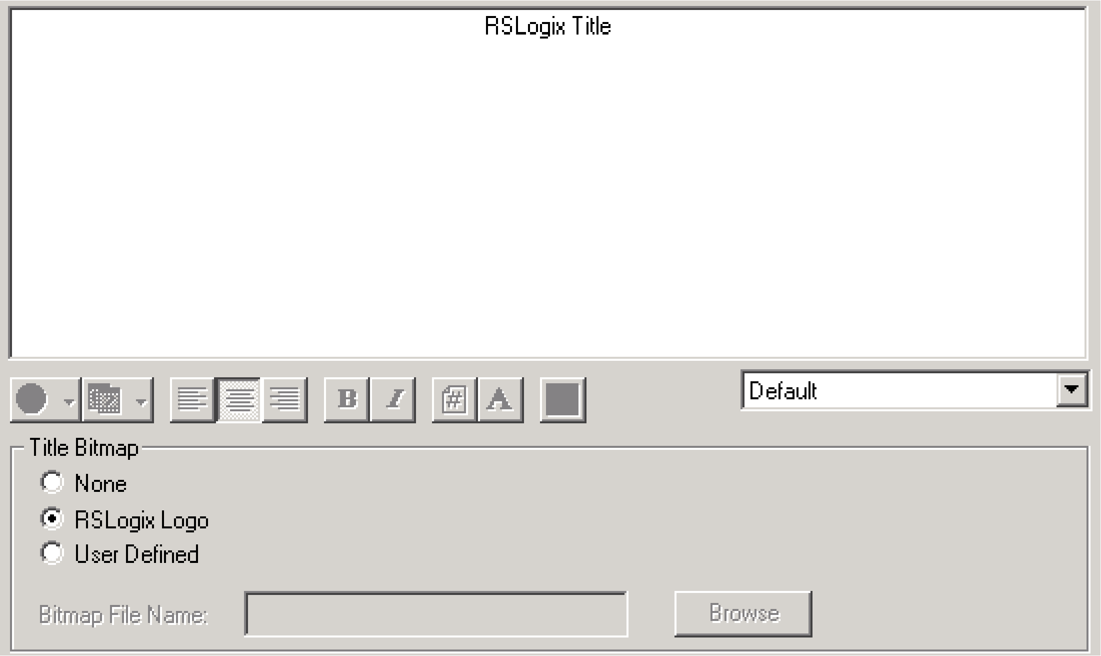
Customizing the report

If you plan to keep the report, you want it to look professional. You can configure your own title page, and a header and footer for every page. Your report can contain your own text, and you own bitmap.

Since the configuration of all three options (title, header and footer) are the same, we will just take a look at the report's title page.

Double click 'Title' under Layout on the left side of your page.





RSLogix Title

Default

Title Bitmap

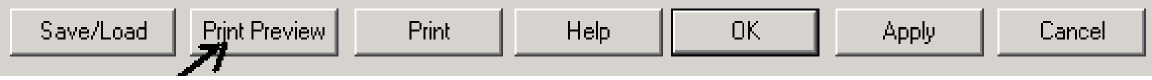
○ None
● RSLogix Logo
○ User Defined

Bitmap File Name:

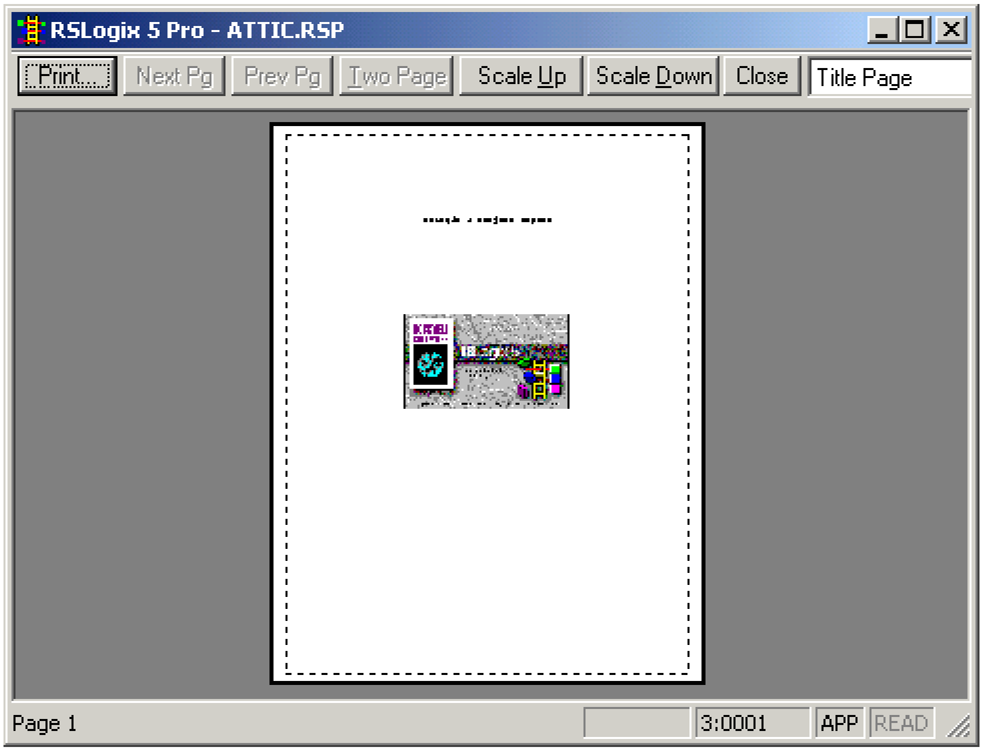
In the above example, you will notice that you don't have the ability to change the text from RSLogix Title to your own title. If you want to customize the text, you must hit the pull down tab, and change 'default' to 'user'. You can also define your own bitmap, and browse to the file on your system.

Previewing your report

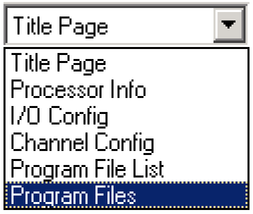
Since a report this large can use a lot of paper, let's do a preview to make sure the report is going to print how we would expect. Click the Print Preview button.



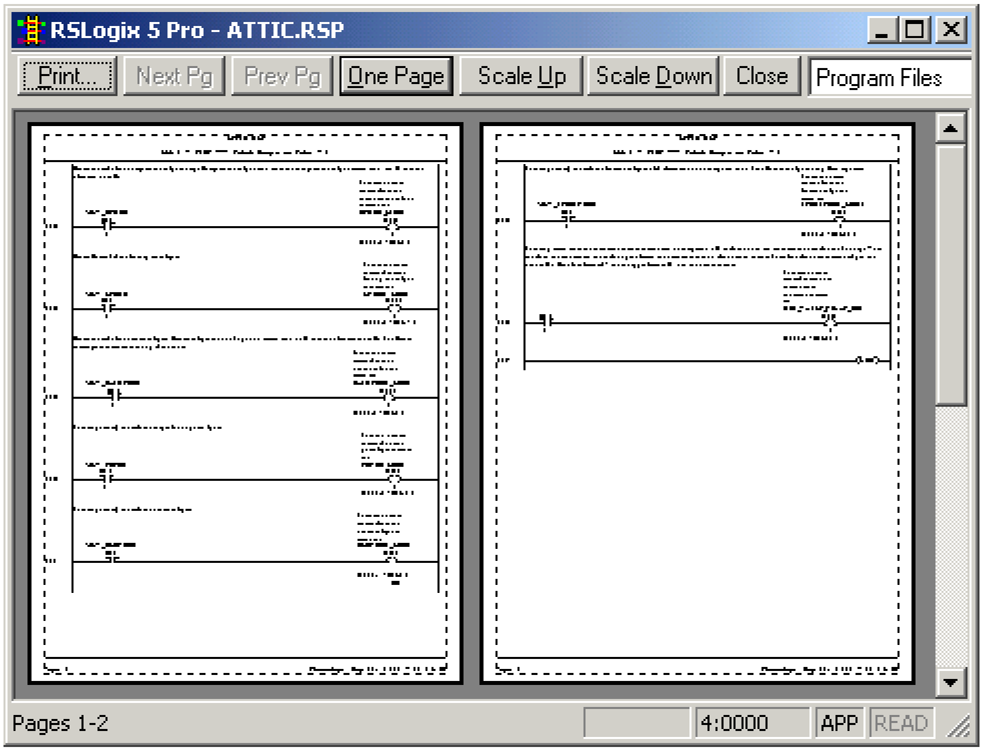
The report is generated. Notice that when you move your mouse over the preview, your mouse pointer becomes a magnifying glass. Click on the report to zoom in. Click again... And again... Notice there are 3 different zoom levels. The last time you clicked the report went back to standard size.



We are currently looking at the title page. From the pull down menu, choose 'Program Files'.



You see the JSR's in ladder 2. Click the 'Next File' to get to ladder 4. Remember, we didn't include ladder 3 in the report. While within ladder 4, you can click next page and previous page to view the entire ladder, or you can press the 'Two Page' button to view both pages at the same time. You must be in the lowest zoom level for this to be an option.



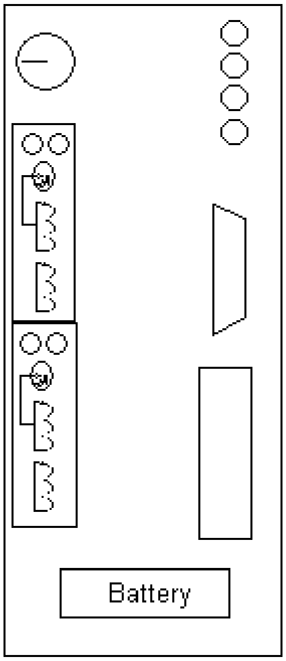
The scale up and scale down buttons make the content of each page larger or smaller (not the page itself). Take a look at some of the other pages you generated. See if the preview is what you expected when the report was being set up.

On Your Own!

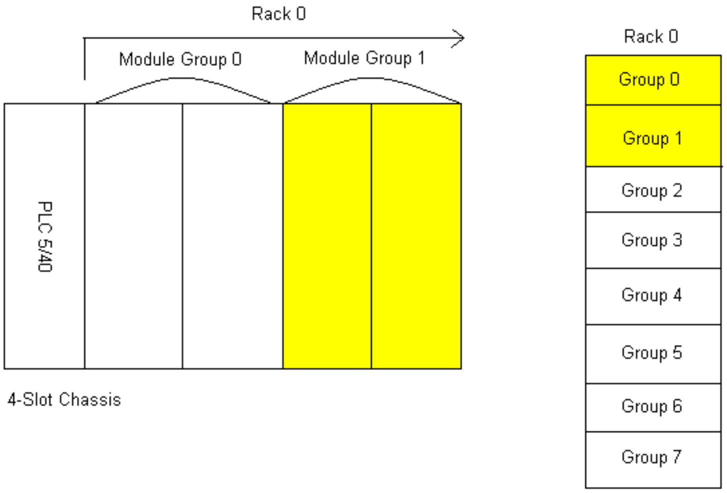
Open a project that is currently in use at your location. Configure the reports you would like to see printed, and run a print preview.

Appendix A -- Worksheets

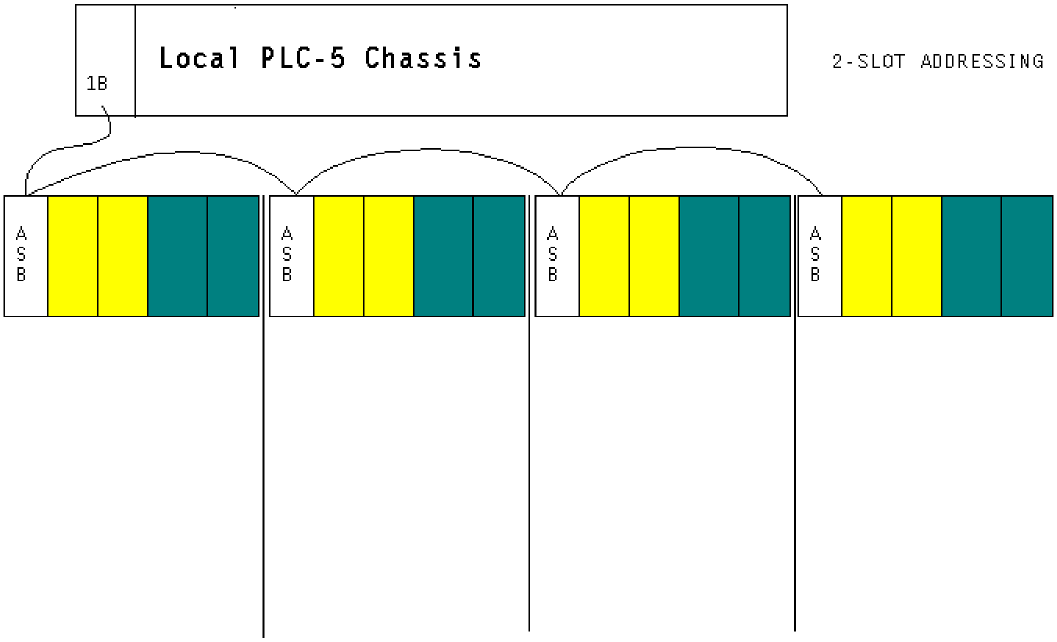
The PLC-5 Processor Worksheet:



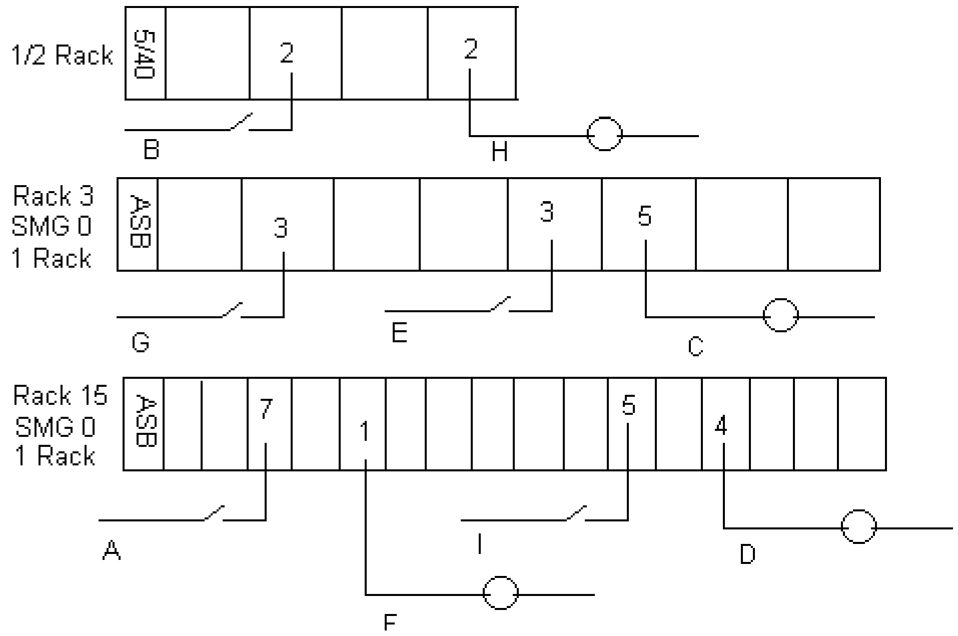
Memory Usage



Addressing Multiple chassis



Addressing Quiz 1:



A:____________________

B:____________________

C:____________________

D:____________________

E:____________________

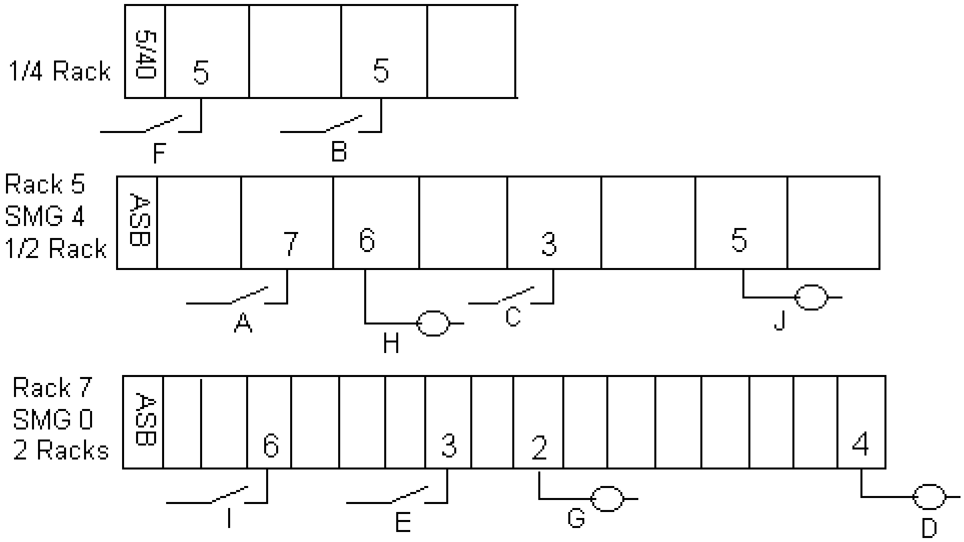
F:____________________

G:____________________

H:____________________

I:____________________

Addressing Quiz 2:



A:_________________

B:_________________

C:_________________

D:_________________

E:_________________

F:_________________

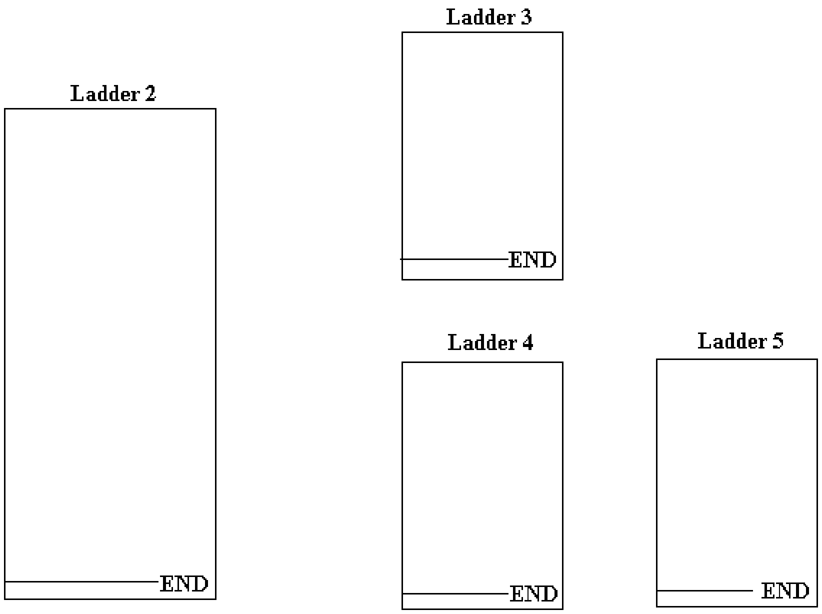
G:_________________

H:_________________

I:_________________

J:_________________

SubRoutine Worksheet


Ladder 3
END
Ladder 2
END
Ladder 4
END
Ladder 5
END