Introduction to Studio 5000 Logix Designer™

For Classroom Use Only!

LISTEN. THINK. SOLVE.®

Allen-Bradley • Rockwell Software

Rockwell Automation

Publication Number -- Date

Copyright © 2013 Rockwell Automation, Inc.

Important User Information

This documentation, whether, illustrative, printed, “online” or electronic (hereinafter “Documentation”) is intended for use only as a learning aid when using Rockwell Automation approved demonstration hardware, software and firmware. The Documentation should only be used as a learning tool by qualified professionals.

The variety of uses for the hardware, software and firmware (hereinafter “Products”) described in this Documentation, mandates that those responsible for the application and use of those Products must satisfy themselves that all necessary steps have been taken to ensure that each application and actual use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards in addition to any applicable technical documents.

In no event will Rockwell Automation, Inc., or any of its affiliate or subsidiary companies (hereinafter “Rockwell Automation”) be responsible or liable for any indirect or consequential damages resulting from the use or application of the Products described in this Documentation. Rockwell Automation does not assume responsibility or liability for damages of any kind based on the alleged use of, or reliance on, this Documentation.

No patent liability is assumed by Rockwell Automation with respect to use of information, circuits, equipment, or software described in the Documentation.

Except as specifically agreed in writing as part of a maintenance or support contract, equipment users are responsible for:

- properly using, calibrating, operating, monitoring and maintaining all Products consistent with all Rockwell Automation or third-party provided instructions, warnings, recommendations and documentation;
- ensuring that only properly trained personnel use, operate and maintain the Products at all times;
- staying informed of all Product updates and alerts and implementing all updates and fixes; and
- all other factors affecting the Products that are outside of the direct control of Rockwell Automation.

Reproduction of the contents of the Documentation, in whole or in part, without written permission of Rockwell Automation is prohibited.

Throughout this manual we use the following notes to make you aware of safety considerations:

WARNING

Identifies information about practices or circumstances
that can cause an explosion in a hazardous environment,
which may lead to personal injury or death, property damage, or economic loss.

IMPORTANT

Identifies information that is critical for successful application and understanding of the product.

ATTENTION

Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you:

- identify a hazard
- avoid a hazard
- recognize the consequence

SHOCK HAZARD

Labels may be located on or inside the drive to alert people that dangerous voltage may be present.

BURN HAZARD

Labels may be located on or inside the drive to alert people that surfaces may be dangerous temperatures.

Introduction to Studio 5000 Logix Designer™

Contents

Before you begin

About This Lab

This session provides you with an opportunity to explore the ControlLogix or CompactLogix platforms, depending on the station at which you find yourself seated. The following sections explain what you'll be doing in this lab session, and what you will need to do to complete the hands-on exercises.

What You Will Accomplish In This Lab

As you complete the exercises in this hands-on session, you will:

- Learn the primary advantages of Logix based controllers
- Design, create and download programs to a Logix controller
- Examine a controller executing a program

Who Should Complete This Lab

This hands-on lab is intended for:

- Controller users who want to become familiar and comfortable with the basics of the Logix Designer within the Studio 5000 programming environment.

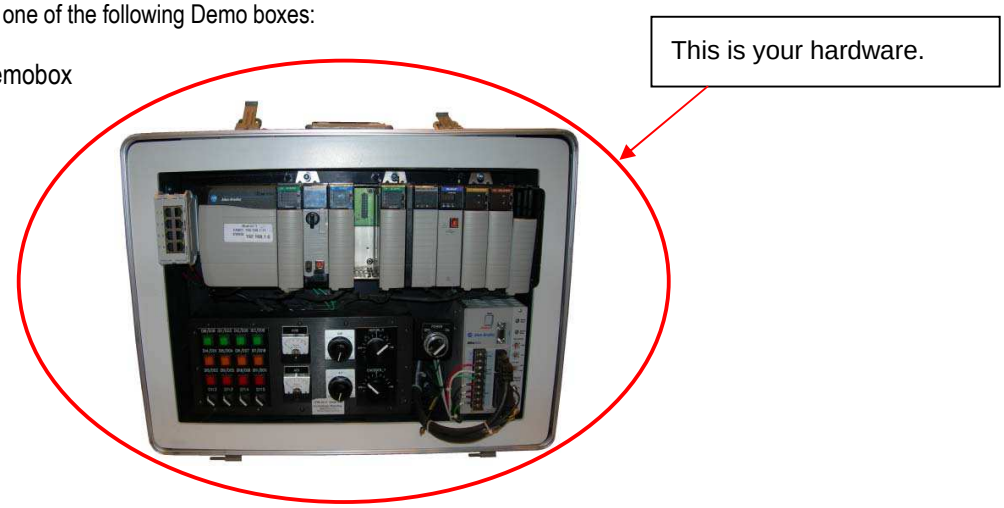
Lab Materials

For this Hands-On lab, we have provided you with the following materials that will allow you to complete the labs in this workbook.

Hardware

This hands-on lab requires one of the following Demo boxes:

1. ControlLogix demobox



2. CompactLogix demobox (either L43 or L35E)

Software

This hands-on lab uses the following software:

- Studio 5000 programming software
- RSLinx Classic software

Files

There are no starting project files for this lab; you will be creating your own file as you go.

Document Conventions

Throughout this workbook, we have used the following conventions to help guide you through the lab materials.

This style or symbol:	Indicates:
Words shown in bold italics (e.g., ***Studio 5000*** or ***OK***)	Any item or button that you must click on, or a menu name from which you must choose an option or command. This will be an actual name of an item that you see on your screen or in an example.
Words shown in Courier text, enclosed in single quotes (e.g., '***Controller1***')	An item that you must type in the specified field. This is information that you must supply based on your application (e.g., a variable). **Note:** When you type the text in the field, remember that you do not need to type the quotes; simply type the words that are contained within them (e.g., Controller1).
FYI	The text that follows this symbol is supplemental information regarding the lab materials, but not information that is required reading in order for you to complete the lab exercises. The text that follows this symbol may provide you with helpful hints that can make it easier for you to use this product.

Note: If the mouse button is not specified in the text, you should click on the left mouse button.

About Logix Controllers

ControlLogix: Perfect for high-speed, high-performance, multidiscipline control

ControlLogix brings together the benefits of the Logix platform — common programming environment, common networks, common control engine — to provide the high-performance your application requires in an easy-to-use environment. Tight integration between the programming software, controller, and I/O reduces development time and cost at commissioning and during normal operation.

ControlLogix offers the following benefits:

- Premier high-speed, high-performance control platform for multidiscipline control (sequential, process, drive, and motion).
- Fully-redundant controller architecture provides bumpless switchover and high availability.
- Widest range of communication options and analog, digital and specialty I/O.
- Select ControlLogix products are TUV-certified for use in SIL 2 applications

With memory options ranging up to 32MB, ControlLogix controllers support intensive process applications and provide fast processing of motion instructions in a single integrated solution.

ControlLogix provides modular network communications that let you purchase only what you need. Interface using ControlLogix communication modules via a ControlLogix gateway, without the need for a processor in the gateway chassis, or interface directly to a ControlLogix controller.

The ControlLogix solution also provides time synchronization capabilities, which is particularly useful in first fault and process sequencing applications.

CompactLogix: Perfect for smaller, machine-level control applications

CompactLogix brings together the benefits of the Logix platform — common programming environment, common networks, common control engine — in a small footprint with high performance. The CompactLogix platform is perfect for tackling smaller, machine-level control applications, with or without integrated motion, with unprecedented power and scalability. CompactLogix is ideal for systems that require standalone and system level control over EtherNet/IP, ControlNet, or DeviceNet. Think CompactLogix when you need economical, reliable control.

CompactLogix offers the following benefits:

- Rackless I/O for flexible installation
- High functionality in an economical platform
- Analog, digital and specialty modules cover a wide range of applications
- Advanced system connectivity to EtherNet/IP, ControlNet, and DeviceNet Networks
- Truly integrated motion control capability

With a user memory ranging from 512K to 3Mb, CompactLogix controllers offer options of USB or serial, EtherNet/IP or ControlNet channels, modular DeviceNet communications and local I/O capacity that can range from 8 to 30 I/O modules.

Use CompactLogix for small- to medium-size solutions including motion axes, I/O, and network connectivity requirements. The new 5370 CompactLogix controllers offer integrated dual Ethernet/IP ports that support Device Level Ring (DLR) topology and integrated motion on Ethernet/IP. The 1769-L3x controllers offer a built in Ethernet port, you can also add an optional 1768-ENBT communication module for L4x controllers for EtherNet/IP communications for plant-wide control.

About Studio 5000

What's new?

Studio 5000 is the first evolution of our Integrated Engineering Environment and is the foundation for the future of Rockwell Automation Engineering Design tools and capabilities. It is the one place needed for design engineers to develop all the elements of their control system. All in one intuitive tool and environment that increases development efficiencies resulting in shorter design cycles and faster time-to-market.

Logix Designer

Studio 5000 is a modular framework for engineering collaboration with plug-ins for specific engineering tasks. For example, there will be a core plug-in that will be used for developing projects for Logix controllers. This plug-in is referred to as **Logix Designer**. Logix Designer brings the existing RSLogix 5000 user interface into the Studio 5000 environment which will introduce new shared components. These components will bring even more power, flexibility, and organization to the Logix design environment. Studio 5000 will be required for all Logix controllers that are running version 21 firmware or greater.

View Designer

A future version will introduce a new core plug-in to Studio 5000. This plug-in will be **View Designer**. View Designer is the graphical design environment for the View 5000 touch screen terminals. This allows developers to design PAC and HMI applications in the same environment. The shared services between the Studio 5000 plug-ins allow major components, such as a tag database, to be shared between PAC and HMI applications.

Section 1: Creating a Project

This lab section should take roughly 20 minutes to complete.

Objective:

- Create a new project
- Write ladder logic
- Use symbolic tag names
- Use the tag monitor/editor

Launching Studio 5000 Configuration Software

In this section of the lab, you will launch the Studio 5000 software, which will allow you to configure and program a controller.

1. Read the ***Before You Begin*** section on page five of this document before proceeding.
2. Double-click on the ***Studio 5000*** icon on the Desktop to launch Studio 5000 software.

The Studio 5000 Splash Screen appears.

FYI

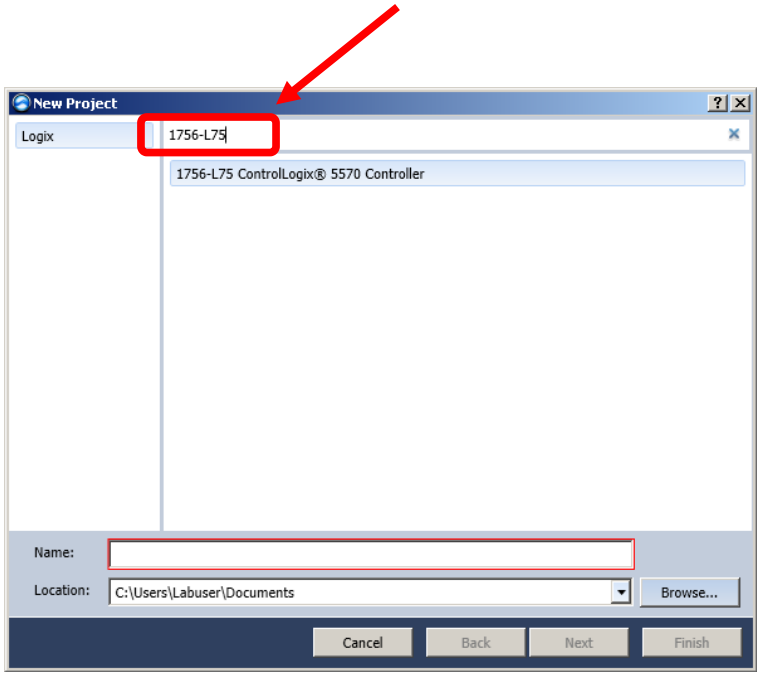
> To see what versions of Studio 5000 you have installed on your computer, select **About** under the **Explore** section.

Creating a New Controller Project

In this portion of the lab, you will create an offline project using a ControlLogix 1756-L75 controller.

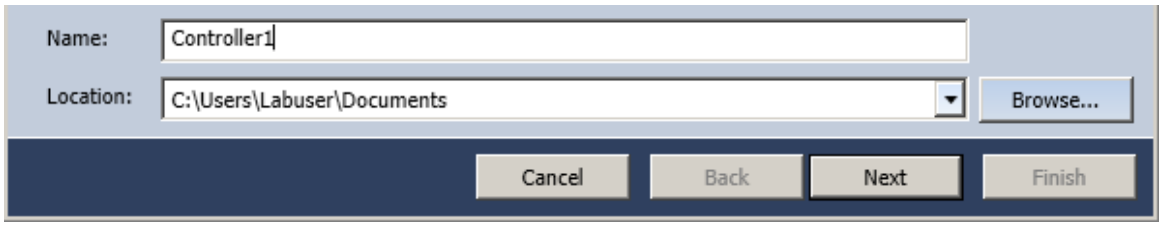
1. Select **New Project** under the Create section.

2. When the **New Project** pop-up is displayed, type ***'1756-L75'*** in the Search field.



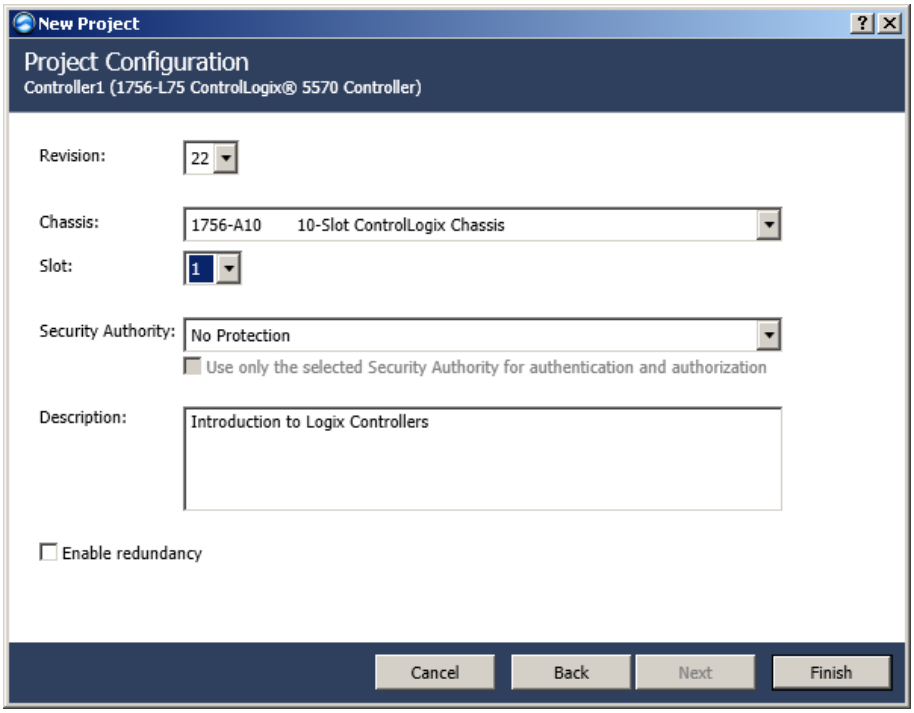
Notice the Name field is highlighted in a red box. This indicates a required field that must be configured before a project can be created.

3. Type ***'Controller1'*** into the name field.



4. Press the **Next** button.

5. When the **Project Configuration** window appears, fill it in as shown below.



- Select V22
- Select the **1756-A10 Chassis**.
- Select **Slot** 1.
- Select **No Protection**.
- Add a project description.
- Click **Finish**

> ***Important Note!*** The Logix controllers in this lab use Studio 5000 software. Be sure to choose the correct controller type that matches the equipment at your lab station. If you are unsure of the equipment at your station, refer to the pictures at the beginning of the lab to verify your hardware. The Controllers have revision 22 firmware.

FYI

From the **New Project** window the following fields are being defined for the project.

Type: This is the type of Logix controller you will use. This could be a ControlLogix, CompactLogix, or SoftLogix controller. Only one programming software package is needed for all Logix Controllers.

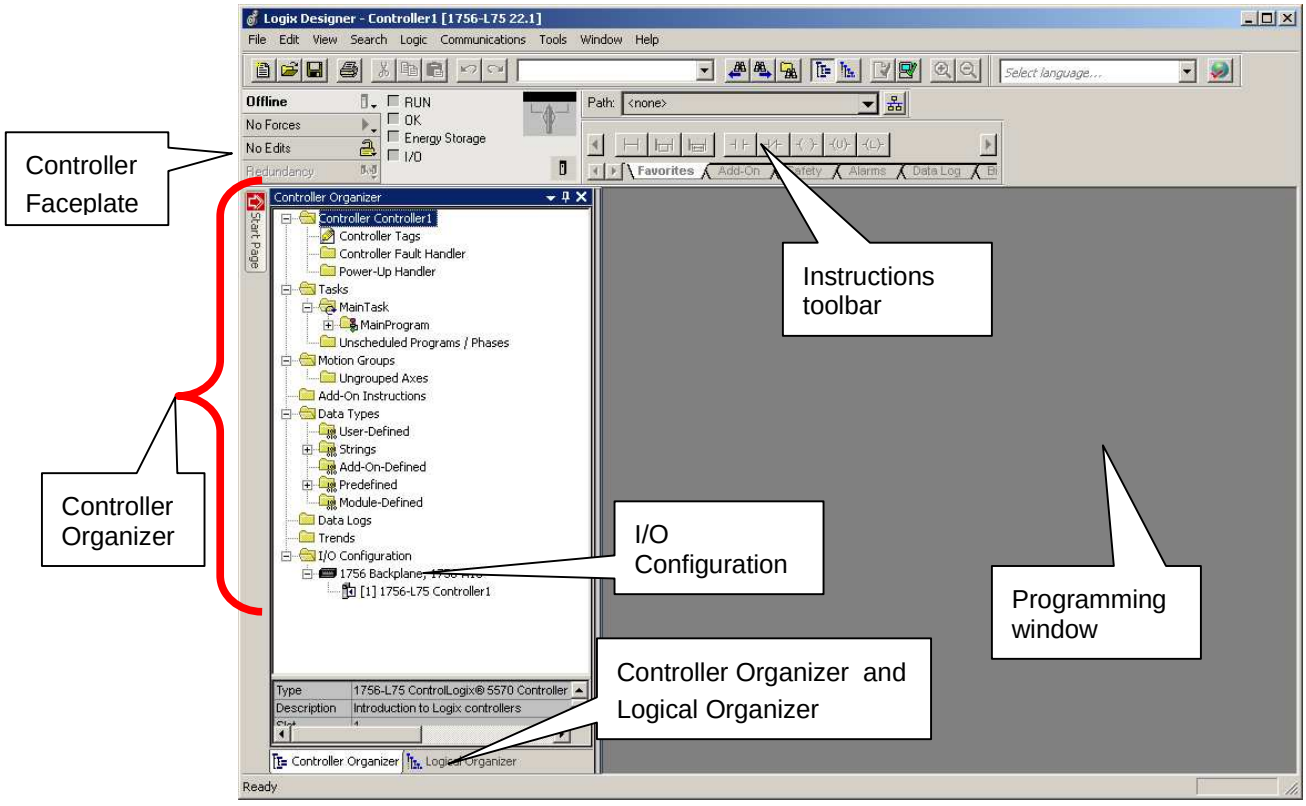
Revision: Here you are selecting the firmware revision of the project that will be created. Lab computers include revision 22.

Name: The name of the controller and project.

Chassis Type: Select the size of the chassis you will use. This is not applicable for all controller types.

Slot: The slot number where the controller will reside. Some controller types will not require a slot number. For example, CompactLogix is fixed at slot zero.

The ***Organizer Window*** appears on the left side of the Studio 5000 window, with a folder called ***Controller Controller1***. At this time, there is no I/O, tag database, or logic associated with the controller.



You have now created your first controller project!

FYI

The **Controller Organizer** is a graphical representation of the contents of your controller file. This display consists of a tree of folders and files that contain all of the information about the programs and data in the current controller file. The default main folders in this tree are:

-Controller File Name

-Tasks

-Motion Groups

-Add-On Instructions

-Data Types

-Trends

-I/O Configuration*

*NOTE: The square containing a '+' or '-' indicates whether a folder is open or closed. Click on it to expand the tree display and display the files in the folder. The - sign indicates that the folder is already open and its contents are visible. By default, the Add-On instructions folder is empty as none are installed.

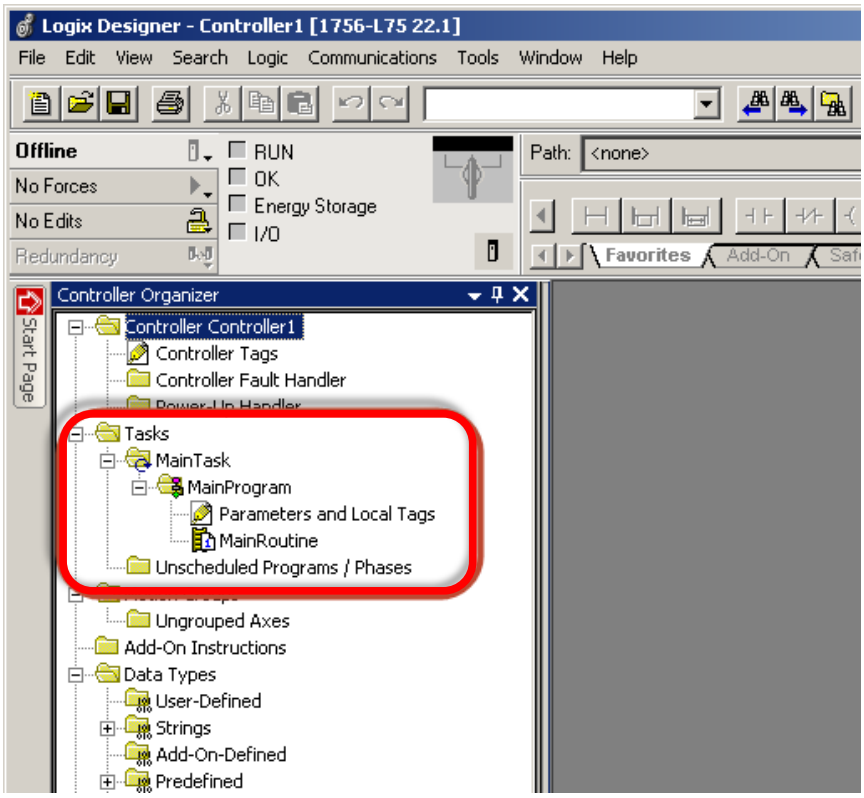
The **Logical Organizer** is not used in this lab. It is there to allow more flexibility in organizing large programs.

Adding Ladder Logic to the Main Routine

In this section of the lab you will add code for a simple motor start/stop seal-in circuit. You will experience the ease of programming with Studio 5000 software. During the labs we will only utilize ladder logic programming, but Logix controllers also can be programmed using Function Block, Sequential Function Charts, and Structured Text. This allows selection of the programming language that best fits an application.

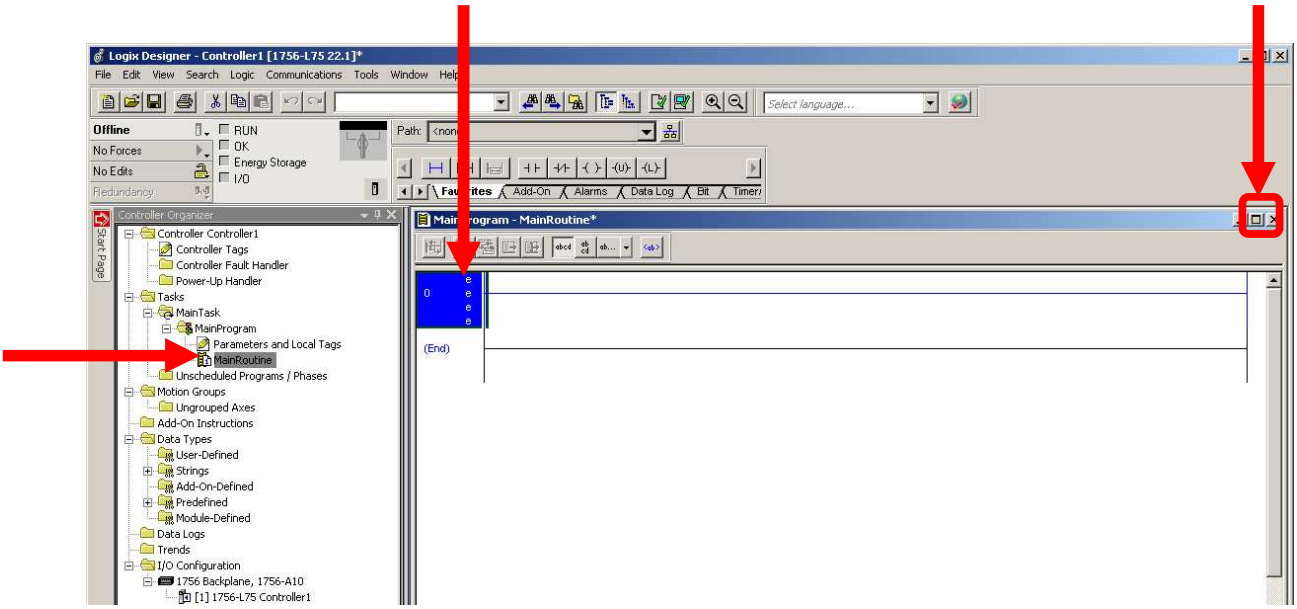
You will continue to use the project already opened.

1. In the Controller Organizer expand the ***MainProgram*** folder by clicking on the **+**. Once expanded, the MainProgram will appear as shown below:

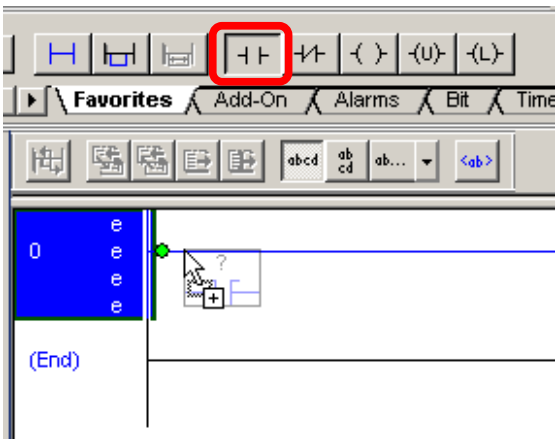


2. Double-click the ***MainRoutine*** icon and maximize the ladder window if it is not maximized.

   This will open the routine editor. An empty rung will already exist as shown below: The “e”s next to the rung indicate the rung is not yet complete.

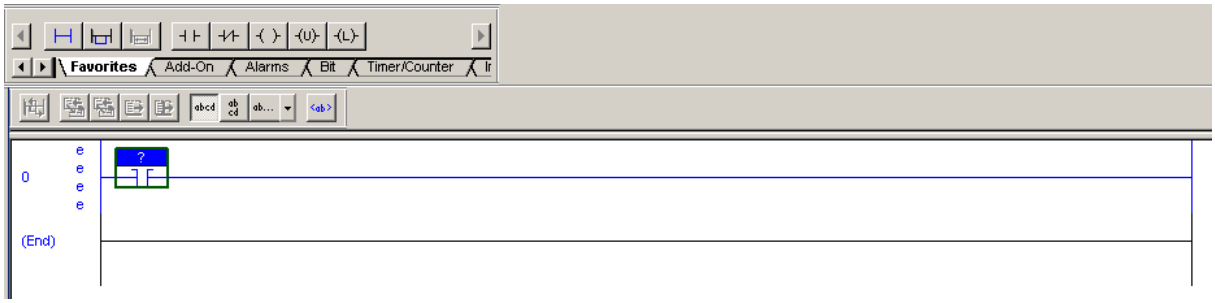


3. From the instruction toolbar, left click and hold on the ***Examine On (XIC)*** instruction. 

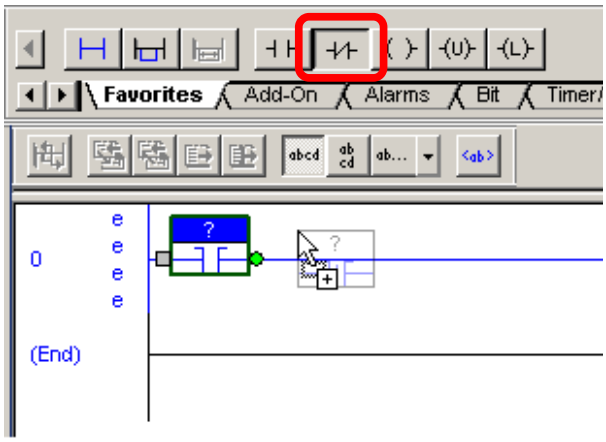


4. Drag the ***XIC*** onto rung *0* until the ***green*** dot appears as shown above. Release the mouse button at the location you wish to place your instruction.

5. Verify your rung appears like the figure below:

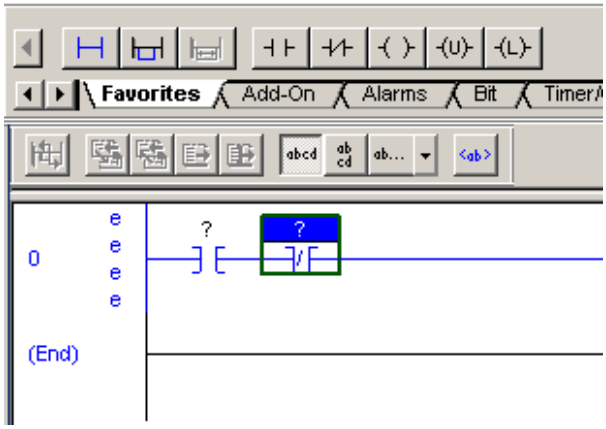


6. From the instruction toolbar left click and hold on the ***Examine Off (XIO)*** instruction. 



7. Drag the ***XIO*** onto rung *0* to the right of the XIC instruction as shown above. Again a green dot will appear to the right of the XIC instruction indicating where your new instruction will be inserted. Release the mouse button at the location you wish to place your instruction.

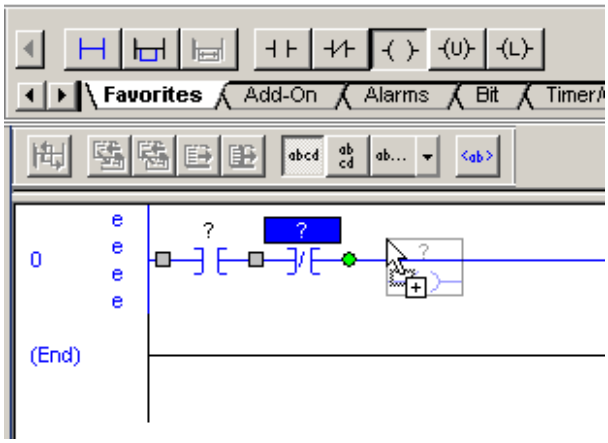
8. Verify your rung appears like the figure below:



FYI

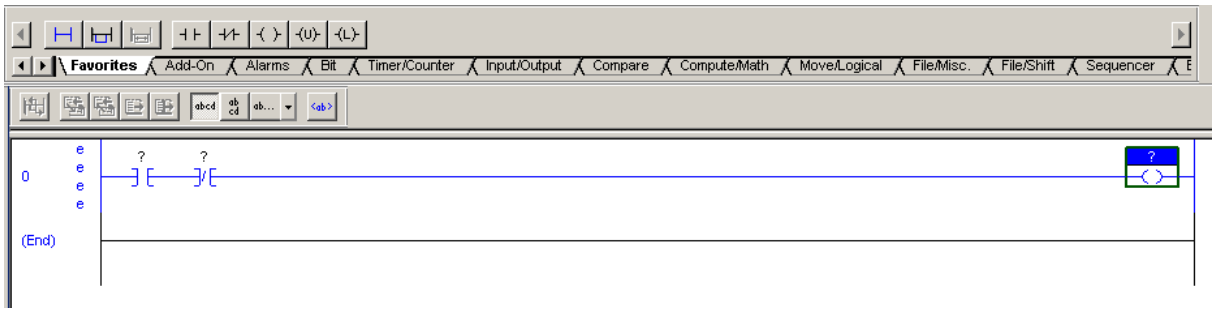
> If you place an instruction in the wrong location on a rung, simply click and hold on the instruction and drag it to the correct location.

9. From the instruction toolbar, left click and hold on the ***Output Energize (OTE)*** instruction.



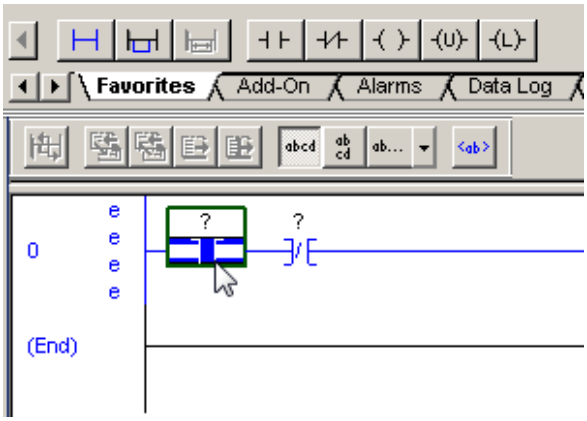
10. Drag the ***OTE*** onto rung 0 to the right of the XIO instruction as shown above. Again a green dot will appear to the right of the XIO instruction indicating where the OTE instruction will be inserted. Release the mouse button at the location you wish insert the instruction.

11. Verify the rung appears as shown below:



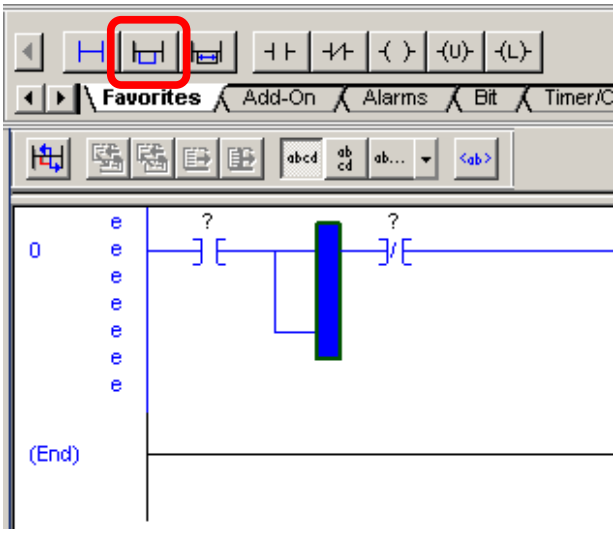
We will now add a branch around the XIC instruction.

12. Click on the ***XIC*** instruction to select it as shown below:



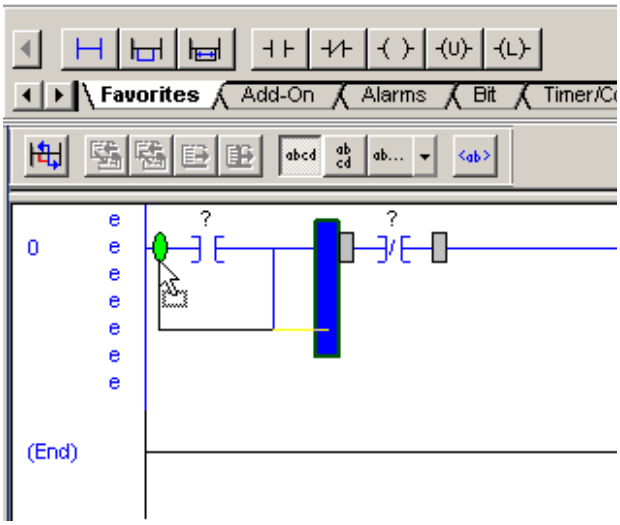
13. From the instruction toolbar click on the ***Branch*** instruction. 

A branch will be inserted on the rung.



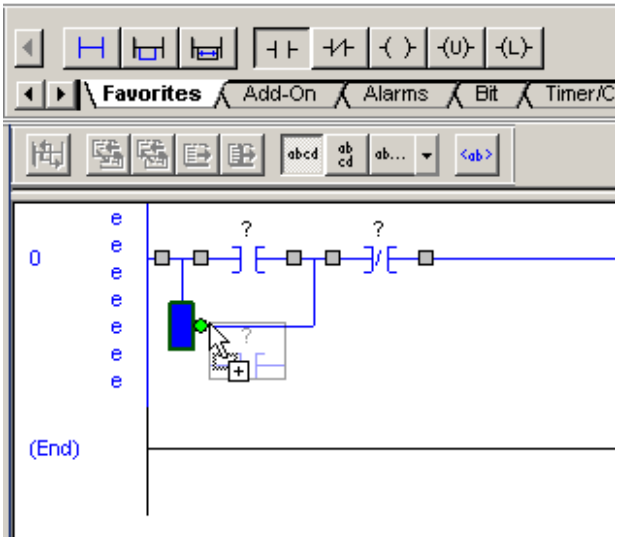
14. Left-click and hold on the ***blue highlighted part of the branch*** and drag your selected leg of the branch to the left side of the XIC instruction.

15. Place the branch over the green dot and release the mouse button.

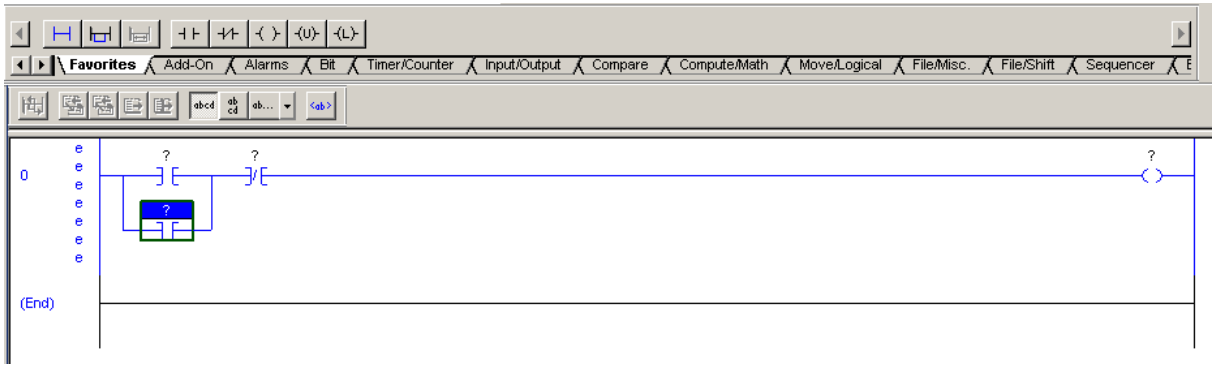


16. From the instruction toolbar, left click and hold on the ***XIC***  instruction.

17. Drag the ***XIC*** onto your newly created branch until the green dot appears.
    The rung should now appear as shown below.



18. Verify that the entire rung appears like the figure below.



19. ***Save*** the program by clicking on the ***Save icon***  on the toolbar. This will save the program in the default directory, which is C:\Users\LabUser\Documents\

FYI

> As you can see the free form editing in Studio 5000 can help speed development. You do not have to place an instruction and tie an address to it before you add the next instruction.

Creating Tags for the Ladder Code

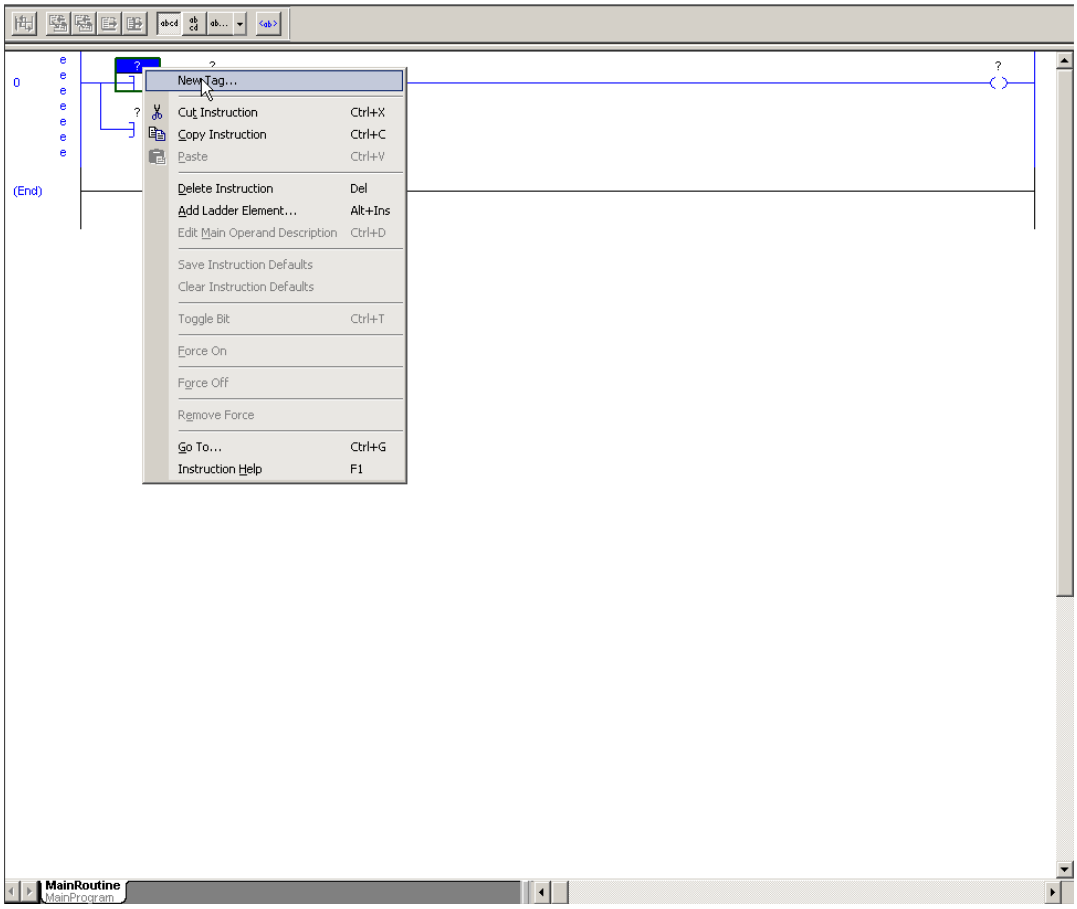
In this section of the lab you will create the tags needed for the program. In traditional PLCs, a physical memory address identifies each item of data, for example N7:0. In Logix controllers, there is no fixed numeric format. Tags are used instead and can be given any name.

FYI

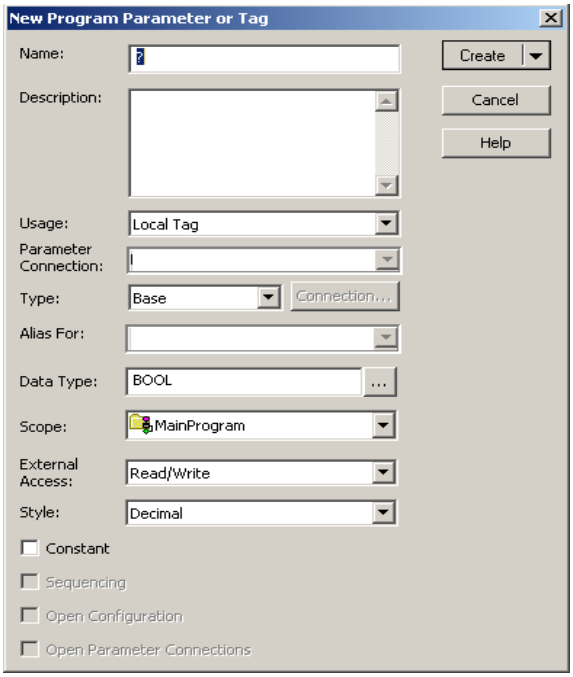
> **What is a tag and why are they better?**
>
> A tag is a text-based name for an area of memory. By using a text-based system you can use the name of the tag to document your ladder code and organize your data to mirror your machinery. For example you could create a tag named North_Tank_Pressure. This helps to speed code generation and debugging. All tag names are stored in the controller.

Continue to use the project already open. We will create 3 tags for the program: ***Motor_Start, Motor_Stop***, and ***Motor_Run***.

1. First create the tag ***Motor_Start***. To do this, right click on the ***? of the first XIC instruction.*** It will be highlighted blue. Select ***New Tag***.



A "**New Program Parameter or Tag**" window will appear.



FYI

Creating a Tag - When you create a tag there are several attributes for a tag. The main attributes we are interested in for this lab are as follows:

Usage: Defines a Local Tag or a Parameter Tag. We will use Local.

Type: Defines how the tag operates within the project

Base: Stores a value or values for use by logic within a project

Alias: A tag that represents another tag

Produced: Send data to another controller

Consumed: Receive data from another controller

Alias For: Only applies when the tag "type" is Alias. Defines the tag which the alias tag will reference.

Data Type: Defines the type of data that the tag stores. Example: Boolean, Integer, Real, String, etc.

Scope: Defines how the data is accessed in the project. It is either controller scoped, global data accessible throughout the controller or program scoped, data accessible for a specific program.

External Access: Defines the access external applications (HMIs) will have with the tag.

Read/Write: External application can read and write to the tag.

Read Only: External application can only read the tag.

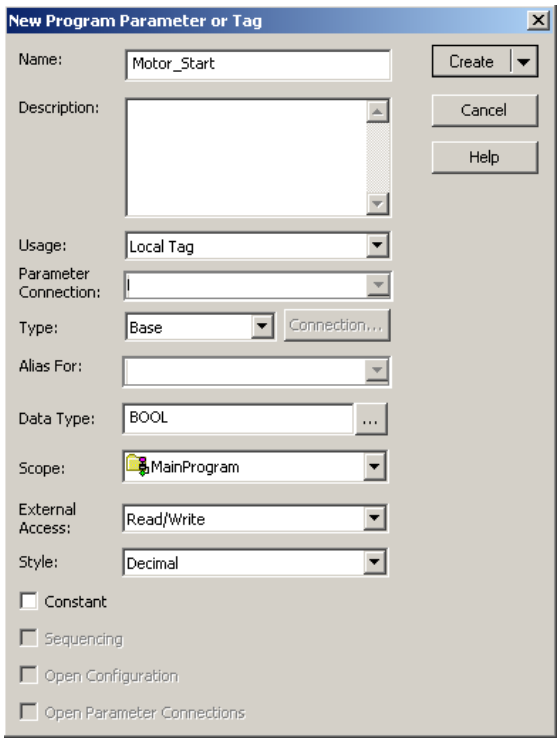
None: External application cannot read the tag or write to the tag

Constant: If checked, that tag cannot be changed programmatically.

Open Configuration: Opens the configuration wizard for complex tags (MSGs, PIDs, etc)

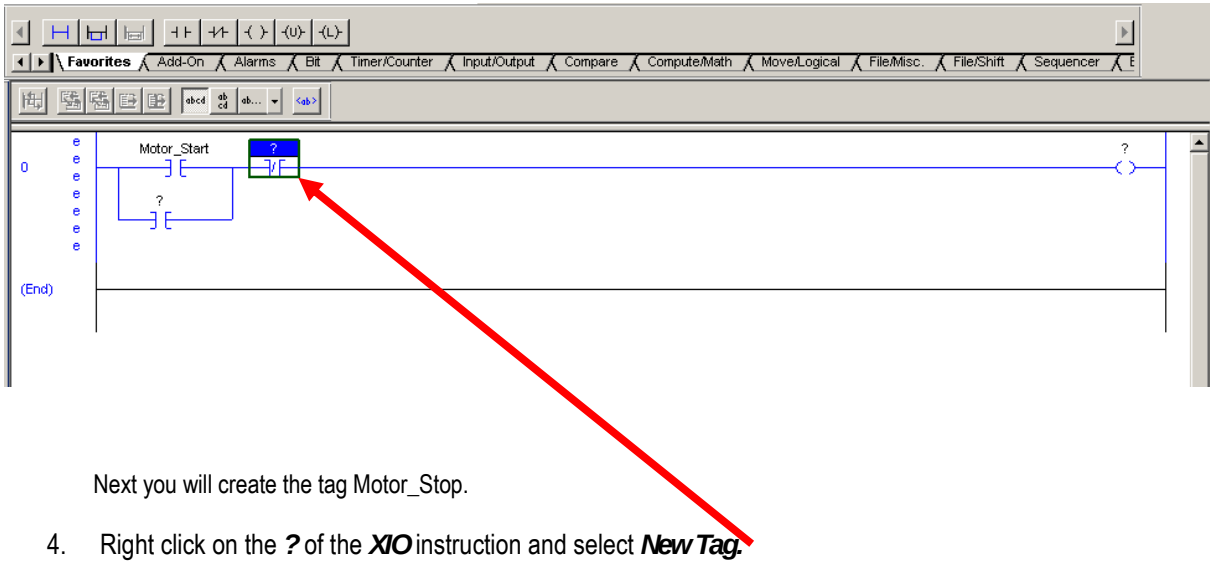
2. Enter the tag fields as shown below.

   Make sure the scope of the tag is MainProgram.



3. Click ***Create*** to accept and create the tag.

   The rung will now look like the figure below.

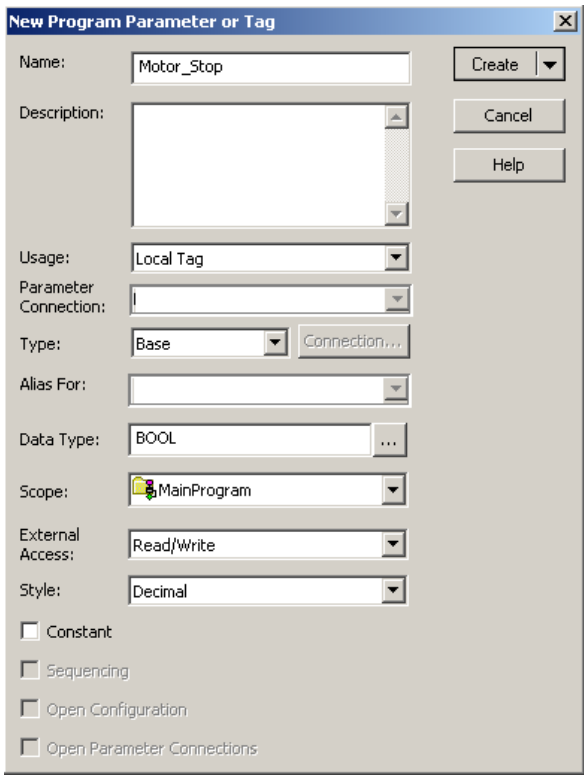


   Next you will create the tag Motor_Stop.

4. Right click on the ***?*** of the ***XIO*** instruction and select ***New Tag.***

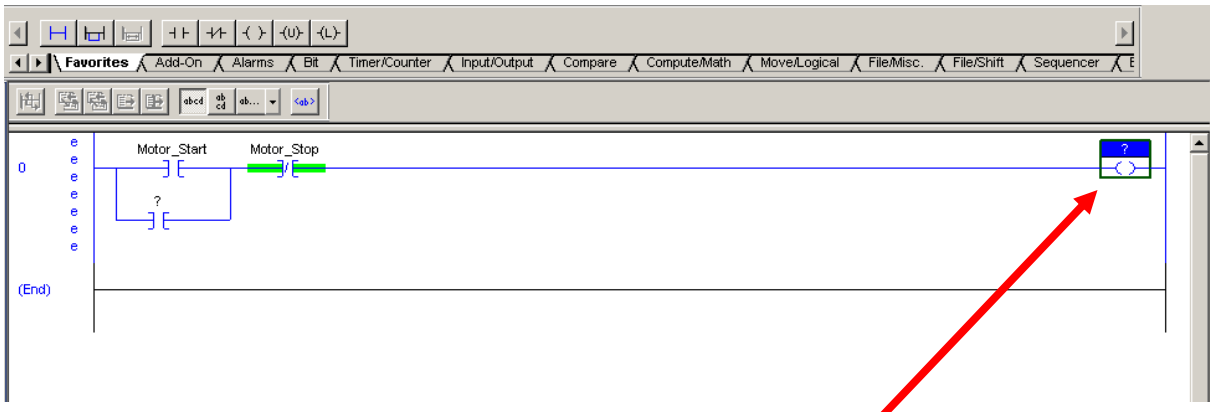
Again, the ***New Tag*** window will appear:

5. Enter the fields as shown below:



6. Click ***Create*** to accept and create the tag.

7. Verify the rung appears like the figure below:

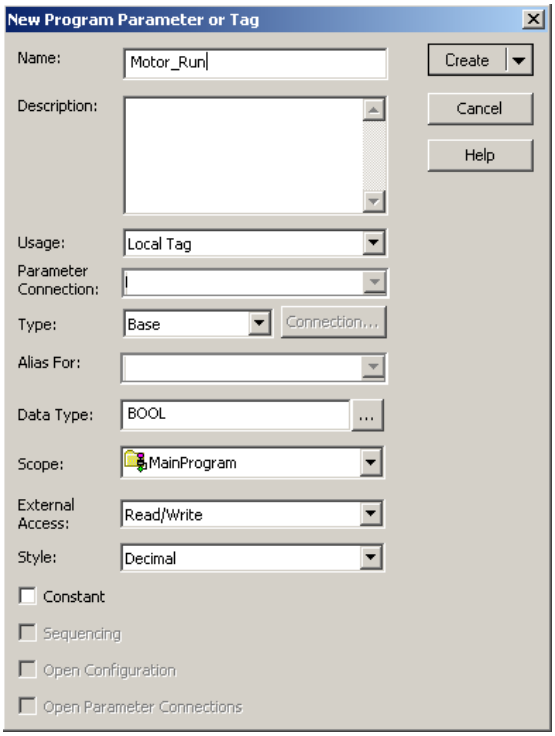


You will now create the tag Motor_Run.

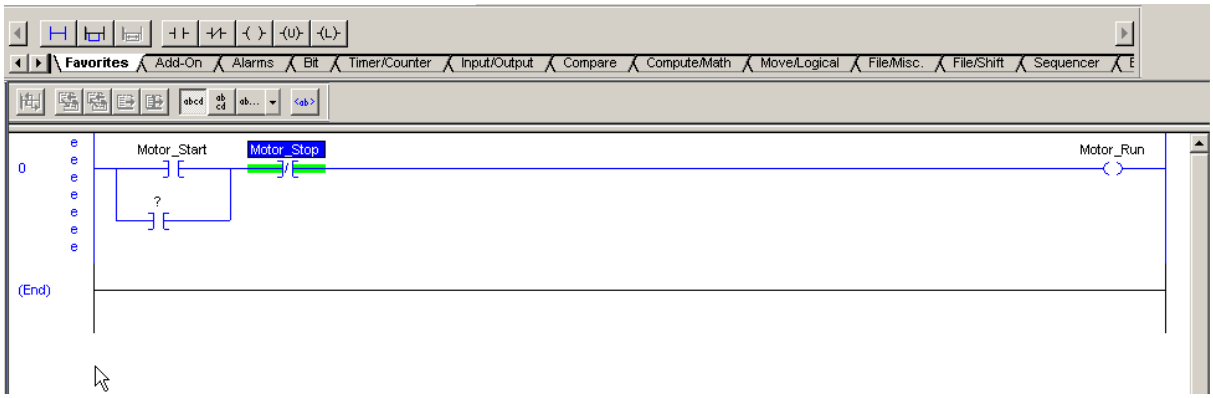
8. Right click on the ***?*** of the ***OTE*** instruction and select ***New Tag.***

The New Tag window will appear.

9. Enter the fields as shown below:

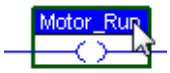


10. Click ***Create*** to accept and create the tag.
    Your rung should now appear as shown below:

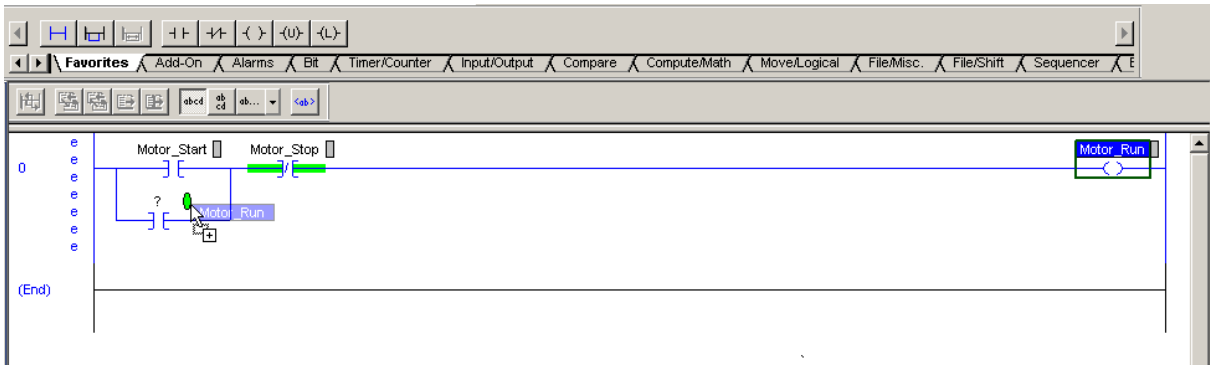


For the ***XIC*** instruction in the branch we do not have to create a tag. You will use the tag Motor_Run.

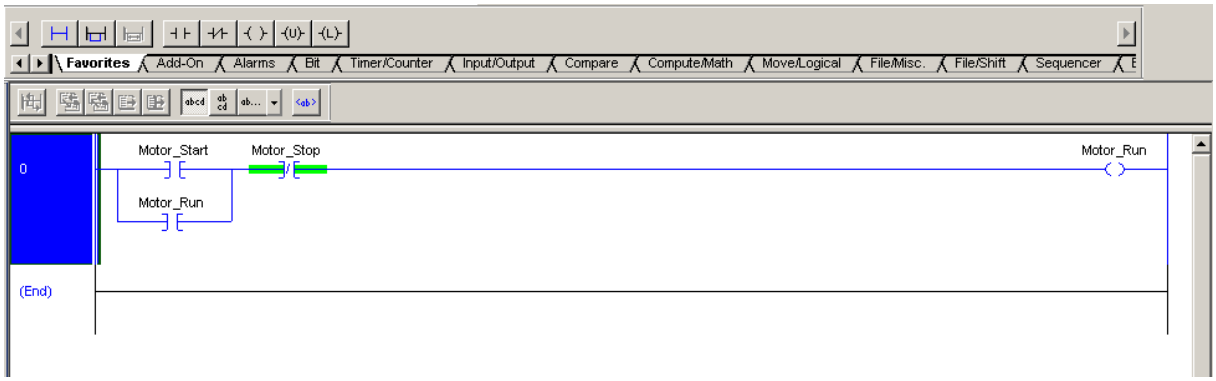
11. Left click and hold the mouse button over the tag ***Motor_Run*** on the ***OTE*** instruction.



12. Drag the tag ***Motor_Run*** over to the ***XIC*** instruction until a green dot appears next to the ***?*** then release the mouse button.

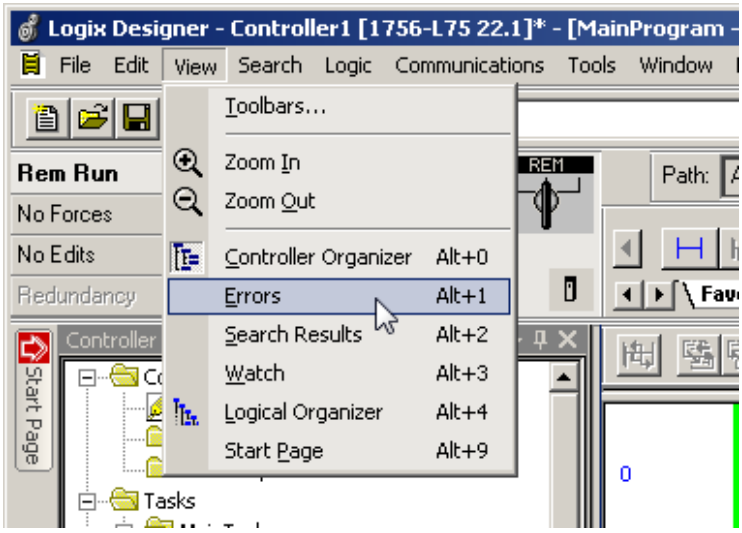


Your rung should now appear as shown below. Notice the "e's" next to rung zero have disappeared. This indicates that the rung passes auto verification and no errors are present.

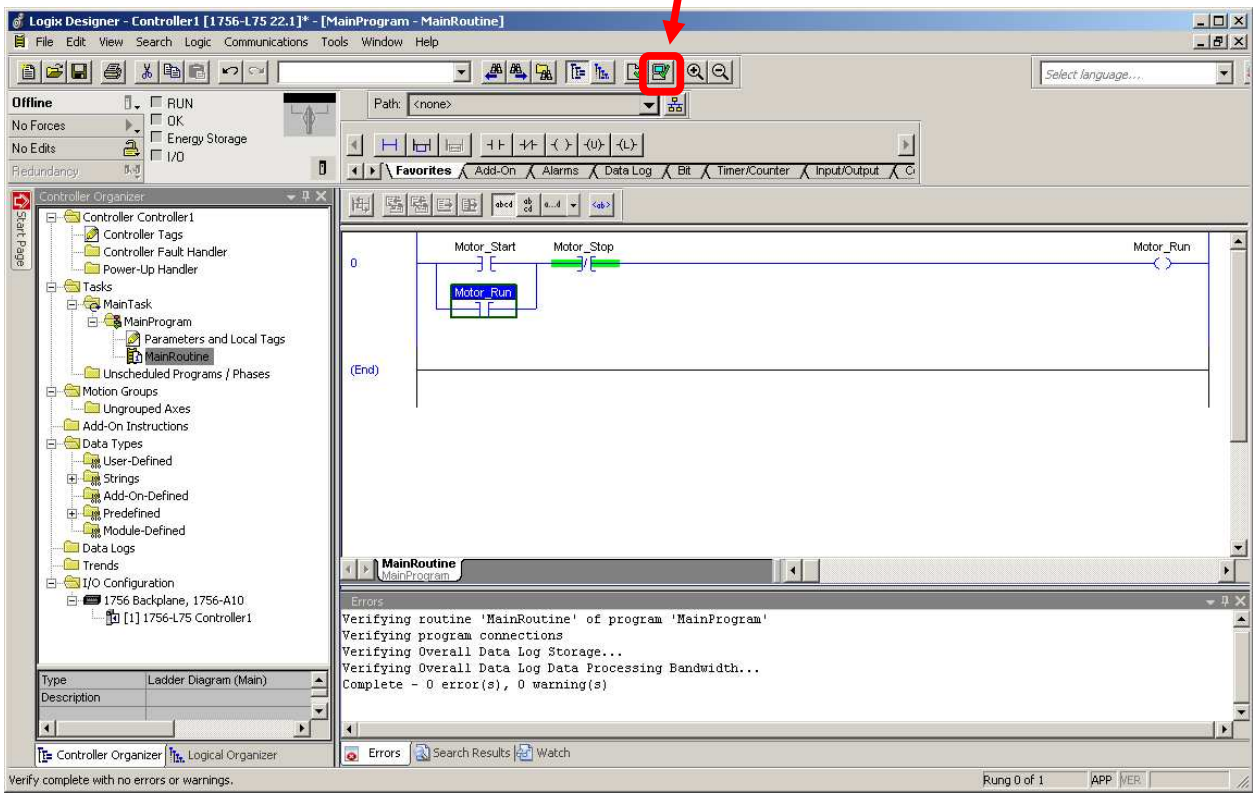


Studio 5000 software verifies each rung automatically when you click the mouse off of it. This makes programming easier!

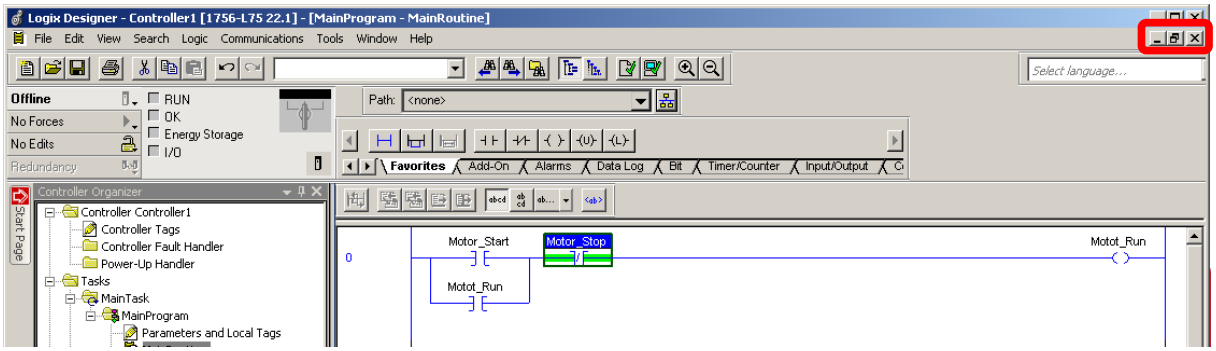
13. Prior to verifying the project, open the error window by going to the ***View*** menu and choosing ***Errors***.



14. Verify the program by clicking on the ***Verify Controller*** icon on the toolbar.
You will see if there are any errors in the status window.



15. Close the ***MainRoutine*** by pressing the ***"X"*** located at the top right corner of the screen.



16. ***Save*** the program by clicking on the Save icon  on the toolbar.

The tag database of Logix versus a traditional PLC's fixed memory addresses help you create self-documenting code. This means you do not have to use address descriptions or symbols to make code easy to read.

New starting with Version 21 – Extended Tag Properties

Extended Tag Properties can be enabled or disabled for each individual Controller or Program scope tag, as long as the data type is compatible. The Extended Tag Properties contain additional information for a given tag which can reduce the need to create custom data structure for commonly used tag information. When enabled, the user can define Engineering Units, a Minimum Limit, and a Maximum Limit for tags of the following data types:

1. DINT
2. INT
3. SINT
4. LINT
5. REAL

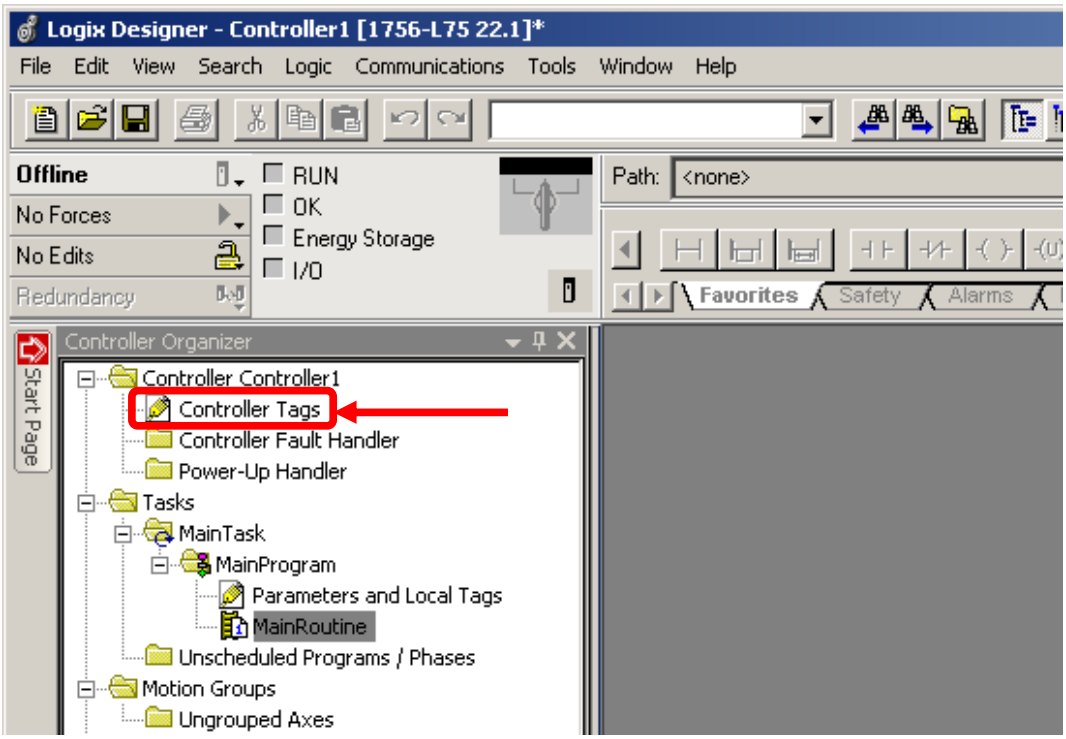
The user will be able to define Engineering Units, an On State and an Off State for BOOL data types. Unlike custom UDTs, Extended Tag Properties will not consume Data and Logic Memory and will be programmatically accessible to the user. Using Extended Tag Properties helps to standardize naming conventions resulting in less confusion, reduced design time and less downtime.

Monitoring/Editing Tags

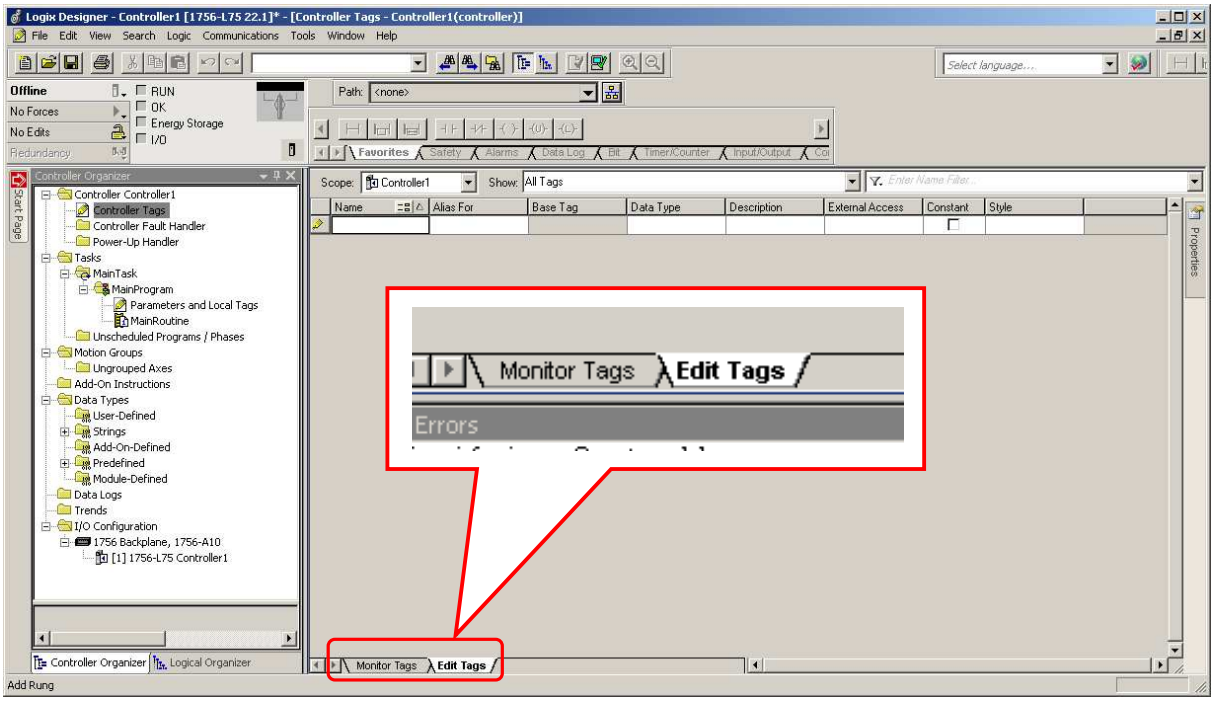
In this section of the lab, we will review the Tag Monitor/Editor in Studio 5000. We will also discuss the concept of Controller versus Program Local scoped tags.

You will continue to use the project already opened.

1. From the Controller Organizer double-click on ***Controller Tags***.



The tag Monitor/Editor window appears. You notice in the lower left corner of the window two tabs labeled ***Monitor Tags*** and ***Edit Tags*** as shown below.



FYI

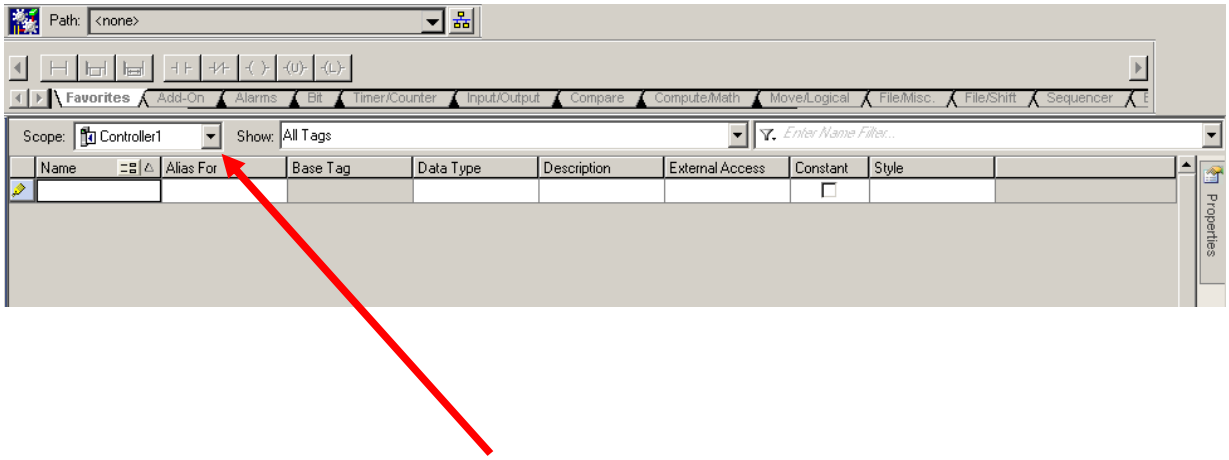
Monitor/Edit Tags Tabs

When the 'Monitor Tags' tab is selected the actual value(s) for the tags are shown and new value can be entered. The tag properties cannot be modified while on the Monitor Tags Tab.

When the 'Edit Tags' tab is selected, values are not shown. Instead, NEW tags may be created, and existing tag properties may be modified.

If you are having difficulty creating or modifying tag parameters, verify that the 'Edit Tags' tab is selected.

You notice first that there are no tags present, remember you just created three tags. These tags were created in Program Scope.

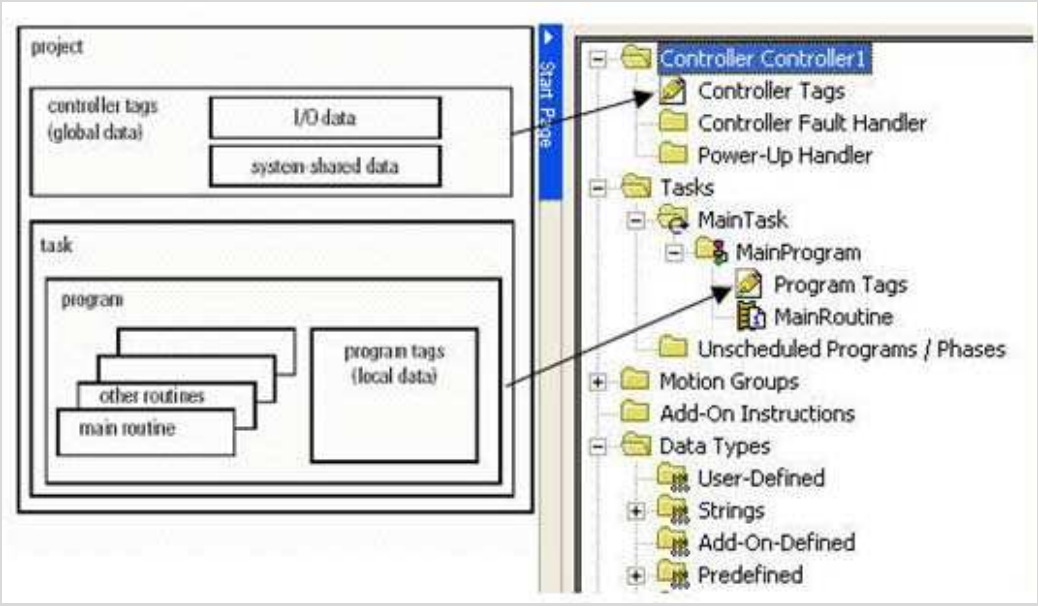


Notice a field in the upper left corner of the Tag Editor window labeled ***Scope***. Earlier in the lab we talked briefly about Controller and Program scoped tags. Currently the selection is ***Controller1,*** which are controller scoped tags.

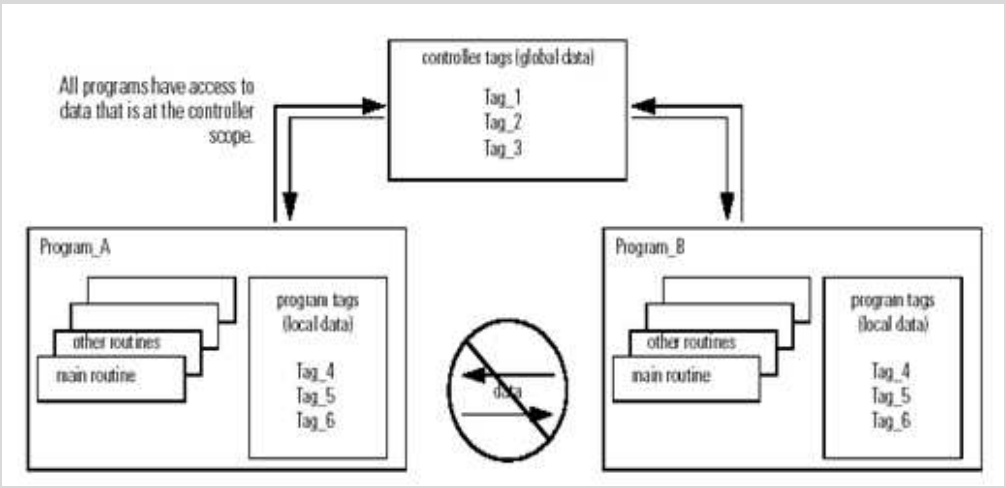
FYI

Data Scoping

When you create a tag, you define it either as a controller tag (global data) or a program tag for a specific program (local data).

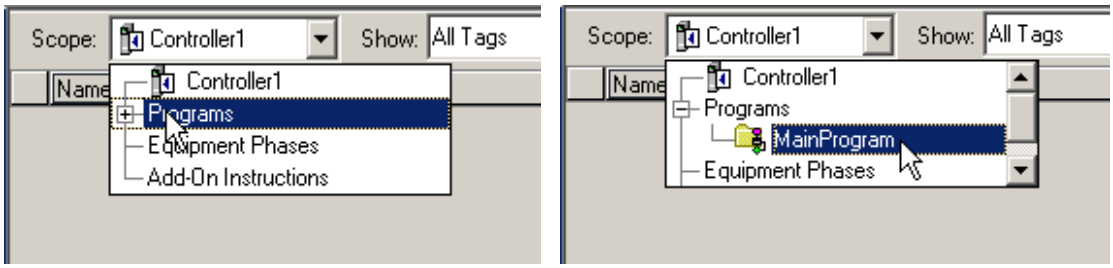


Data at the program scope is isolated from other programs. Routines cannot access data that is at the program scope of another program. Thus you can re-use the tag name of a program-scoped tag in multiple programs.

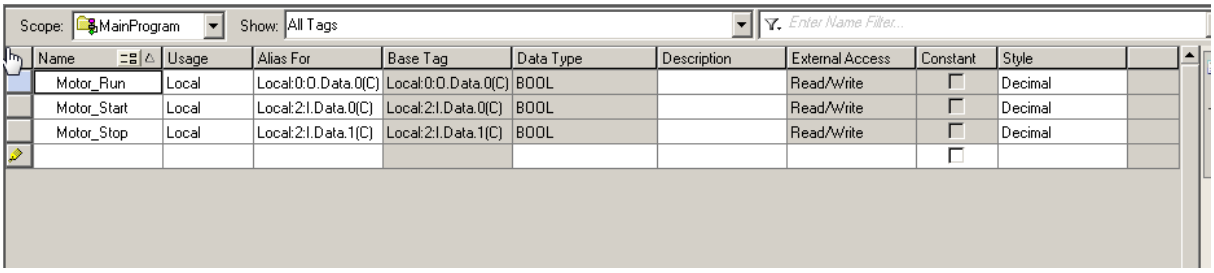


2. Click on the down arrow for the ***Scope*** selection box.

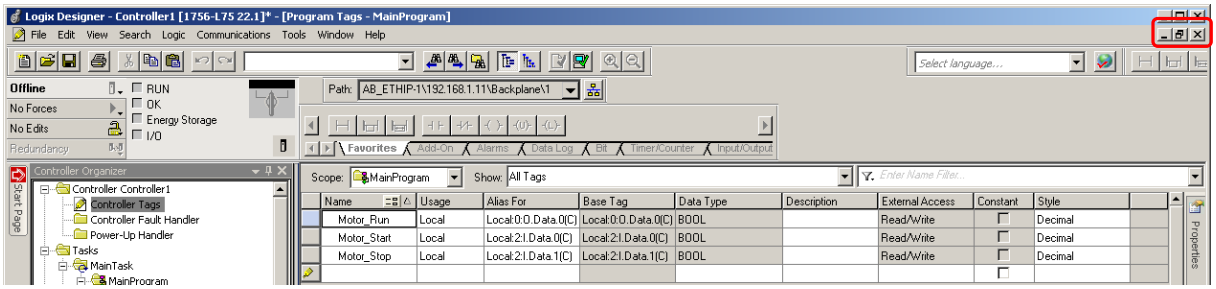
3. Select ***Programs*** → ***MainProgram***



The Tag Editor now has switched views to the program level and you see the tags you created earlier.



4. Close the Tag Editor by pressing the ***"X"*** located at the top right corner of the tag editor.



5. ***Save*** the program by clicking on the Save icon  on the toolbar.

6. Minimize the Studio 5000 software.

Congratulations! You have Completed Section 1. Please move on to Section 2.

Section 2: Connecting Your Computer to the Controller

This lab section should take roughly 5 minutes to complete.

Objective:

In this lab, we will introduce you to the online operations that you will complete with the Studio 5000 software. In this lab, you will:

- Launch RSLinx Classic communications software
- Configure your communications driver

Launching RSLinx Software

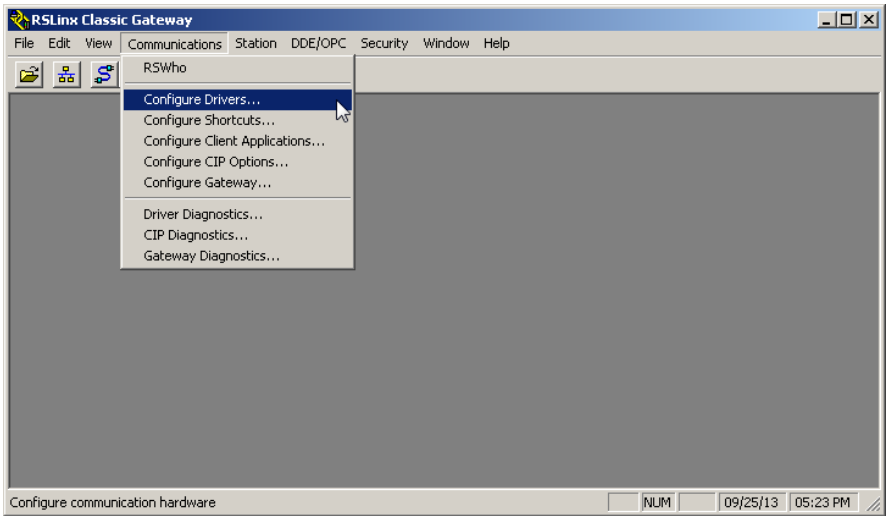
In this section of the lab, you will launch the RSLinx software, which will enable you to configure the driver you will use to communicate with the Logix processor in the Demo Box.

Double click on the ***RSLinx*** icon on the Desktop to launch RSLinx software to bring up the RSLinx Classic Gateway window.

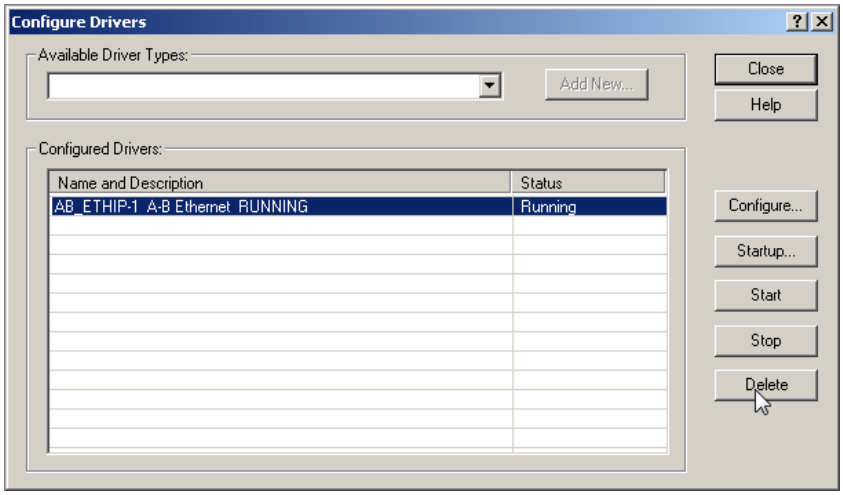
Adding the AB_ETHIP-1 (Ethernet/IP) Driver

In this section of the lab, you will add the Ethernet/IP driver that you will use to communicate with your Logix processor.

1. From the ***Communications*** menu, choose ***Configure Drivers***.

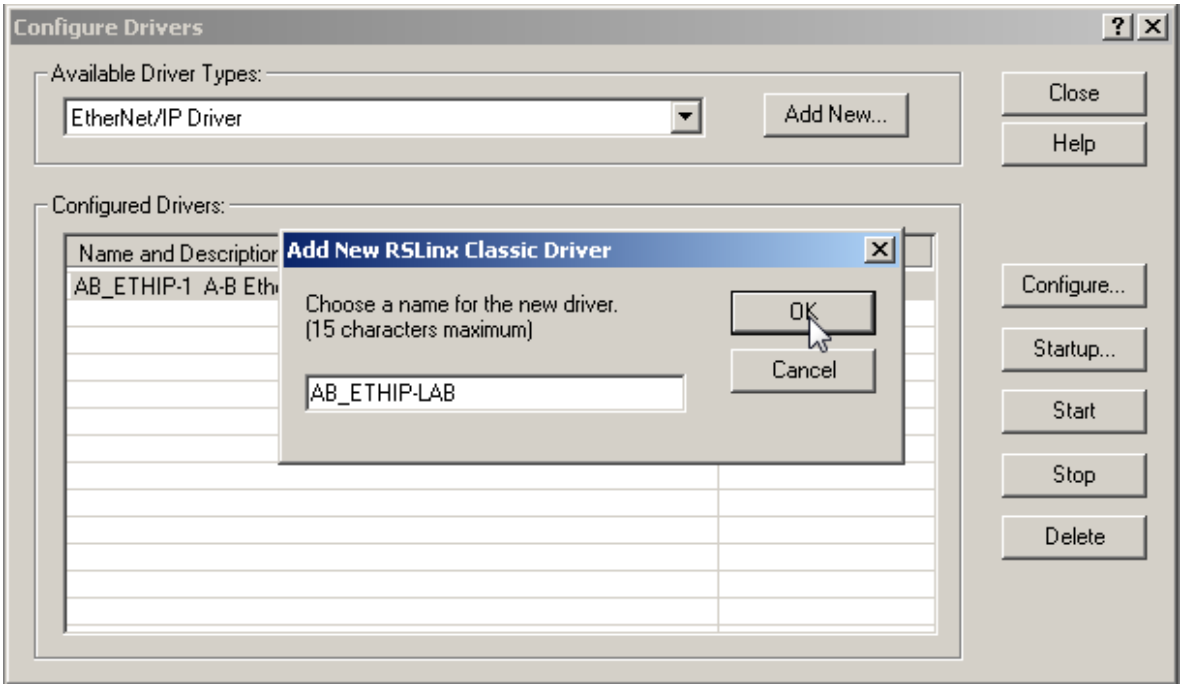


The Configure Drivers dialog appears. There is already a driver configured on this image name AB_ETH-1. However, we are going to create a new driver.

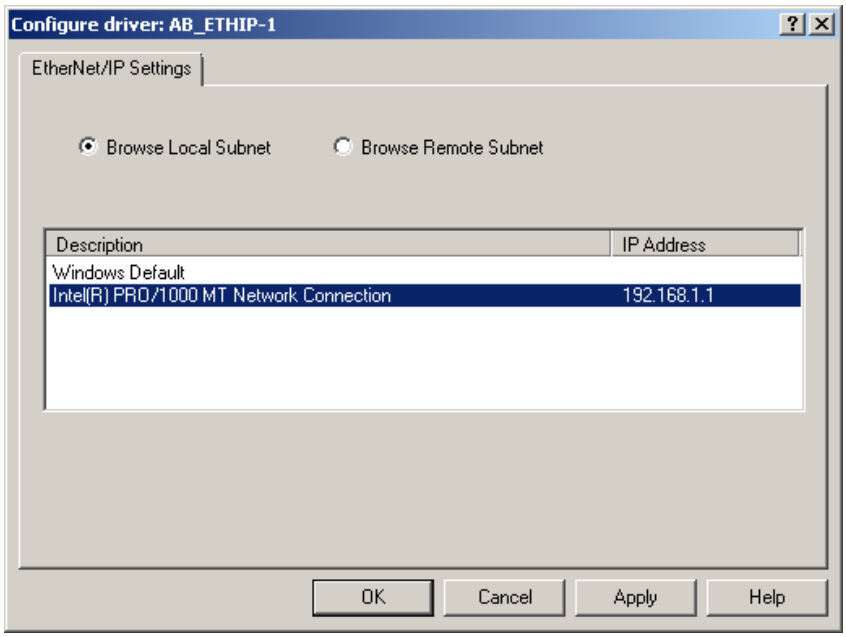


2. From the ***Available Driver Types*** pull-down menu, choose ***EtherNet/IP Driver*** then click on the ***Add New*** button.

3. Change the name of the driver from ***AB_ETHIP-2*** to ***AB_ETHIP-LAB*** as shown and click ***OK***



4. Choose **"Browse Local Subnet"** and then the **"Intel"** network driver as shown below and click ***OK***.

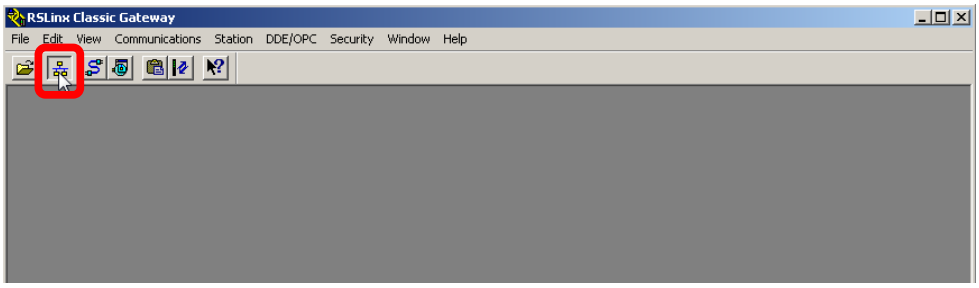


5. Exit the Configure Driver Dialog by clicking on ***Close***.

FYI

> In ***RSLinx*** you will notice two different Ethernet drivers listed: **EtherNet/IP Driver** and **Ethernet devices**. In general, you should use the newer EtherNet/IP driver... it will automatically scan for and find any EtherNet/IP compatible devices on the network. A few older Rockwell Ethernet products cannot be found using this driver. The **older Ethernet devices** driver works with all Rockwell Ethernet products, but it will only scan for IP address that you manually tell it to search for. You can have both types of drivers and/or multiple instances of each type active in ***RSLinx*** at the same time if needed.

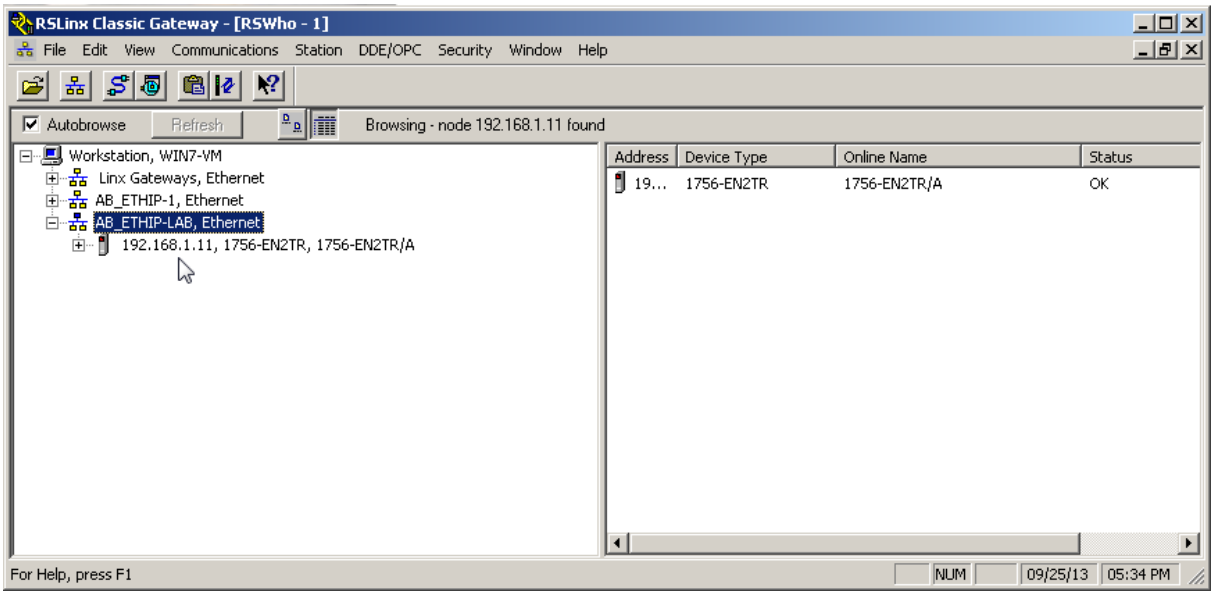
6. Click the ***RSWho*** icon  in the toolbar.



The Rockwell Software RSLinx Gateway - [RSWho - 1] screen appears.

7. Click on the **+** by the ***AB_ETHIP-LAB, Ethernet*** driver to see the Ethernet module with IP 192.168.1.11

   This is the RSLinx driver we will use in RSLogix Designer to download to the Logix controller in the next section.



FYI

RSWho

The RSWho screen is actually RSLinx's network browser interface, which allows you to view all of your active network connections.

The left pane of this display is the Tree Control, which shows networks and devices in a hierarchical view. When a network or device is collapsed, as indicated by the + sign, you can click on the + sign or double click on the network or device icon to expand the view and begin browsing. When a network or device is expanded, as indicated by the - sign, you can click on the - sign or double click on the network or device icon to collapse the view.

The right pane of the RSWho display is the List Control, which is a graphical representation of all of the devices present on a selected network.

Congratulations! You have Completed Section 2. Please move on to Section 3.

Section 3: Downloading the Project from the Computer to the Controller

This lab section should take roughly 10 minutes to complete.

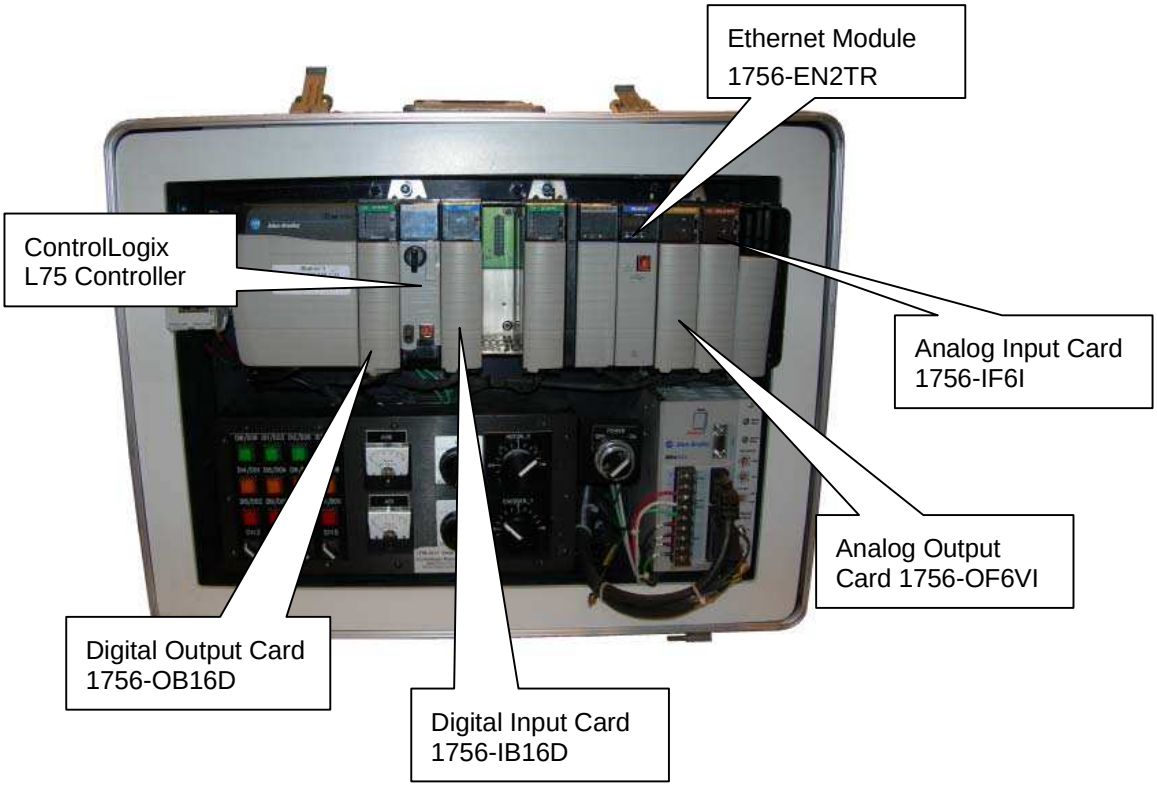
Objective:

In this lab you will open a controller project based on the lab station at which you are seated.

You will:

- Determine the type of controller you are using
- Open the project that corresponds to the controller you are using
- Download the program to the controller

You will be using the program that was created from the steps performed in Lab 1.
Look over the images below if you are unsure of the hardware associated with your lab station demo.

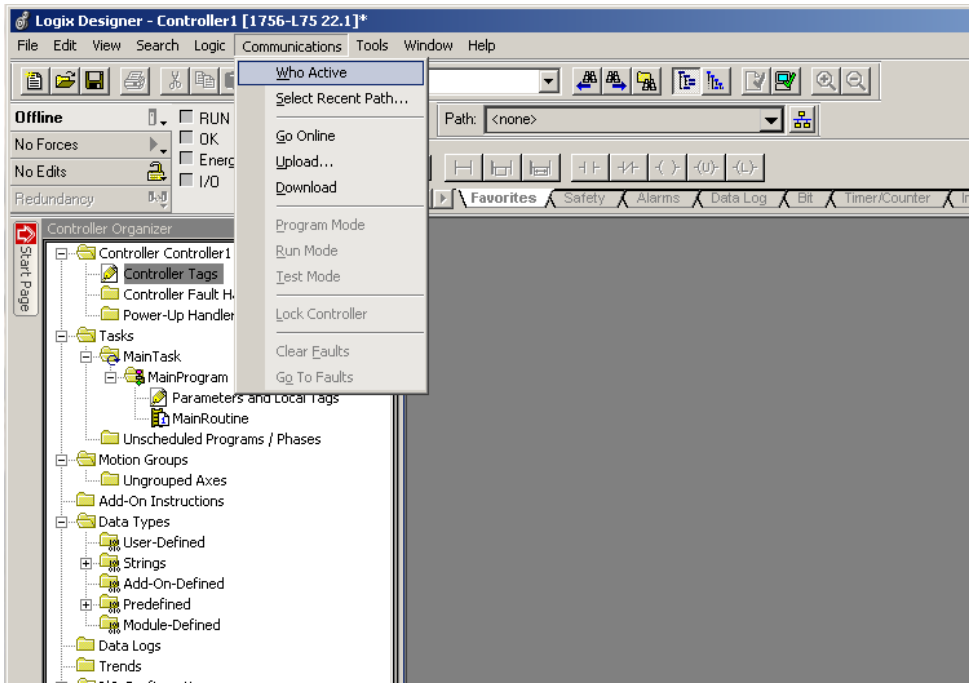


Downloading the Project to the Controller

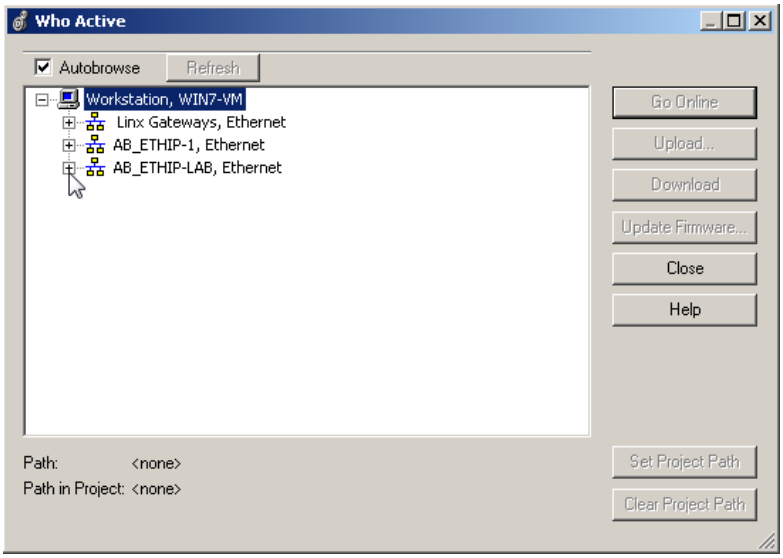
In this section of the lab you will download the project.

1. Maximize Studio 5000 and your Controller1.ACD project.

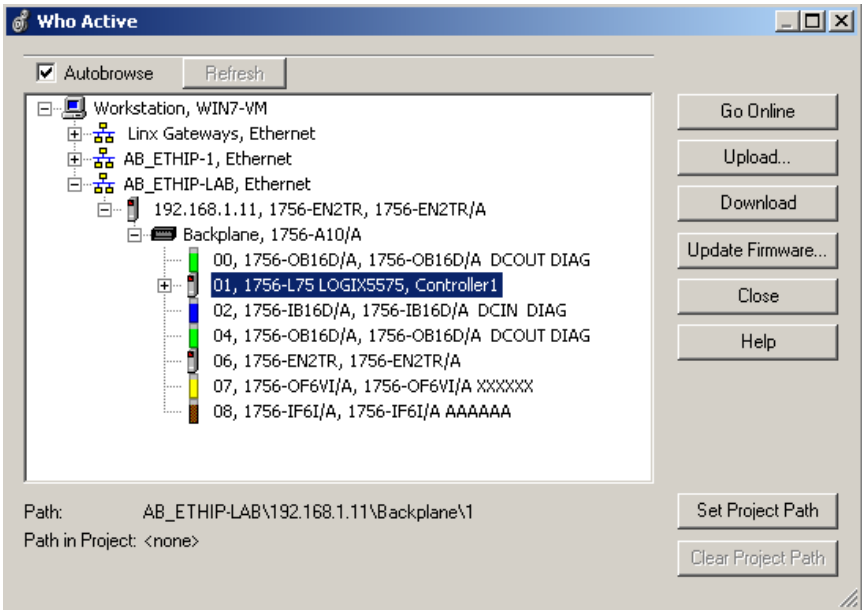
2. From the ***Communications*** menu, choose ***Who Active***.



The ***Who Active*** Screen appears.



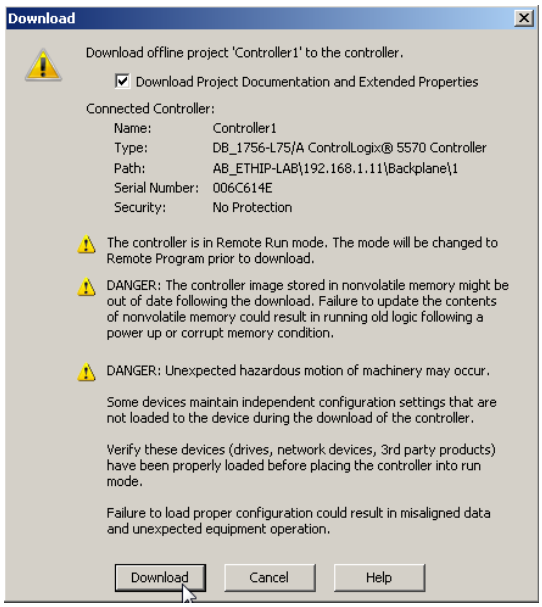
3. **Expand** the ***AB_ETHIP-LAB*** driver and keep expanding the view by clicking on the '**+**'s until you see the ***1756-L5*** controller. **Select** the controller by clicking on it.



FYI

> The Logix family of controllers in this lab all use Studio 5000 software to configure the system, but each controller type is set up slightly differently.

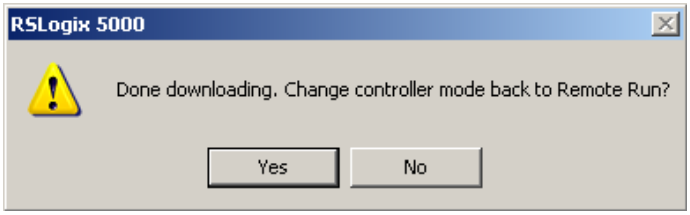
4. Click ***Download***. You will be asked to verify the download. Click ***Download*** again.
   The project will then begin to download to your controller.



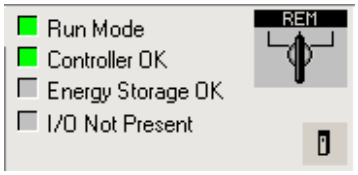
FYI

> If your controller was in the RUN mode prior to the download, you may be prompted to return to the RUN mode. If asked select **YES.**

5. When the following prompt appears, click ***Yes*** to change the controller mode to Remote Run.



At this point you will be online with the controller and the status LEDs on the controller faceplate in your project will mimic the LEDs on your controller.



Congratulations! You have Completed Section 3. Please move on to Section 4.

Section 4: Configuring I/O

We will now look at configuring I/O for our project. To communicate with I/O modules you must add modules to the I/O Configuration folder.

This lab section should take roughly 20 minutes to complete.

Objective:

This part of the lab covers adding 1756 I/O using the equipment at your lab station using several methods, including the module discovery feature.

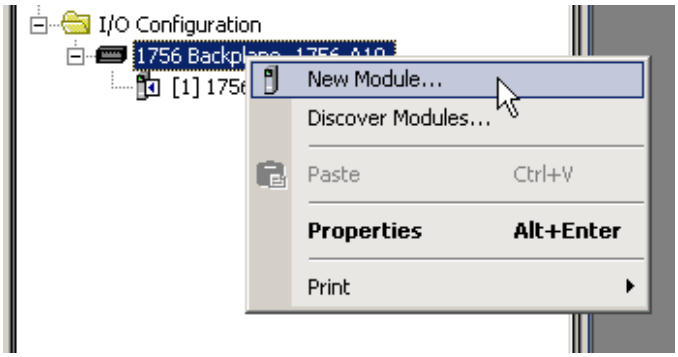
You will continue to use the project already opened.

For this lab we will add the following I/O modules. Please note the I/O that relates to the equipment at your lab station.

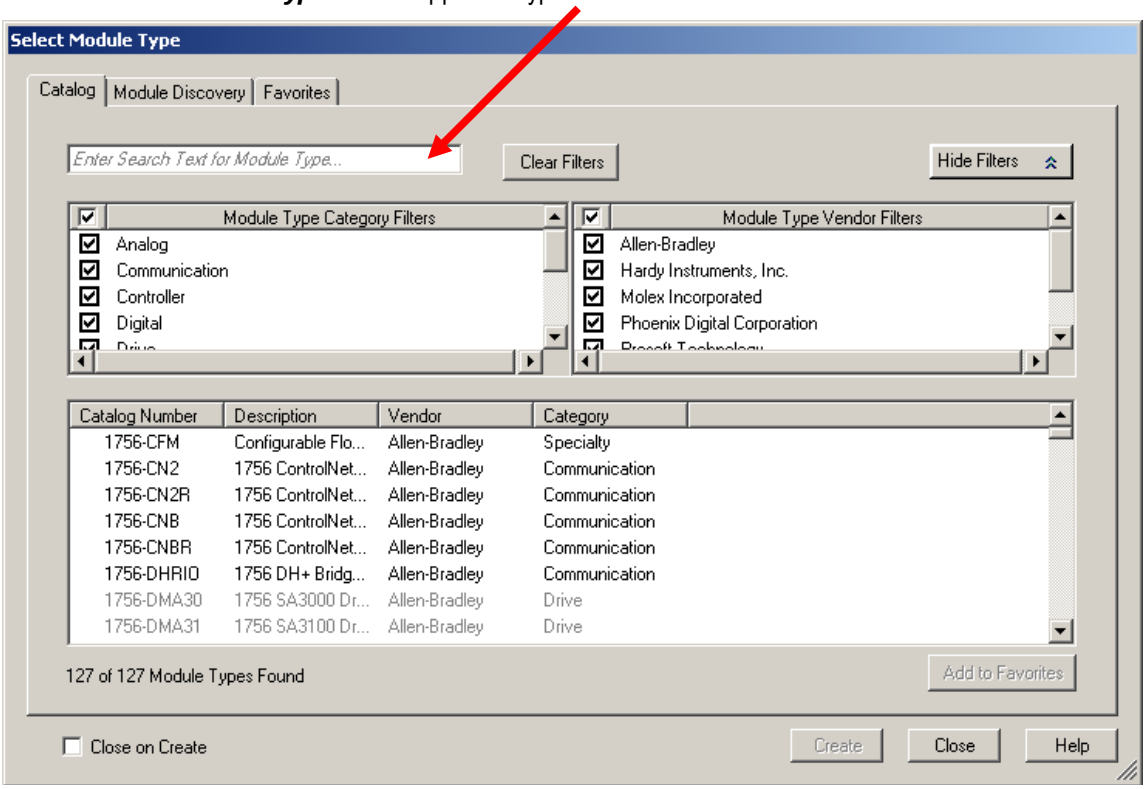
- 1756-IB16D/A – Isolated DC Input Module
- 1756-OB16D/A – Isolated DC Output Module
- 1756-IF6I/A – Isolated Analog Input Module
- 1756-OF6VI/A – Isolated Analog Voltage Output Module

Adding ControlLogix I/O Manually

1. In the ***I/O Configuration Folder***, right click on ***1756 Backplane, 1756-A10*** and select ***New Module***.



2. The ***Select Module Type*** window appears. Type "IB" in the search box.

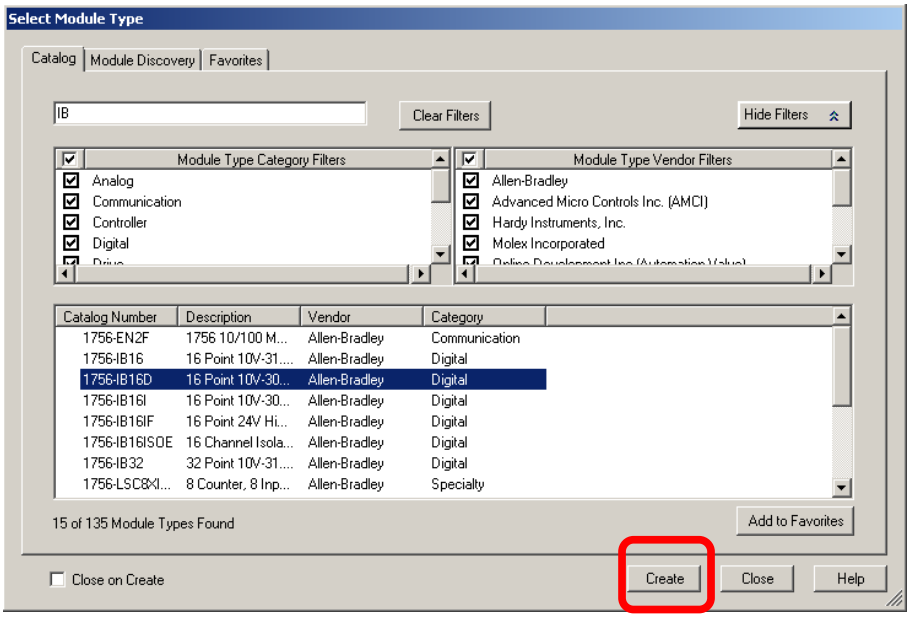


FYI

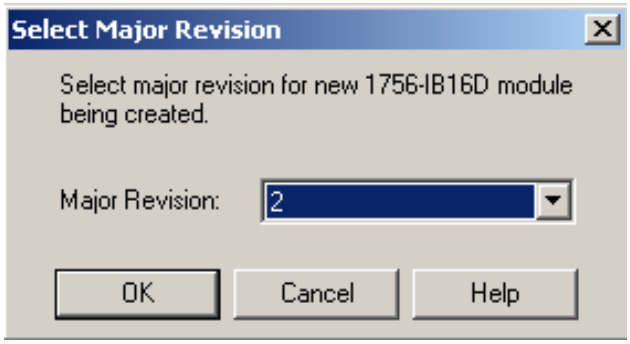
> Items that are "grayed out" are modules that cannot be added while online with the controller. You must be offline to add these modules to your I/O configuration.

3. Locate the ***1756-IB16D.***

4. Select the ***1756-IB16D*** module and click ***Create***



5. **Select** 2 from the drop-down and click ***OK*** on the Select Major Revision window.



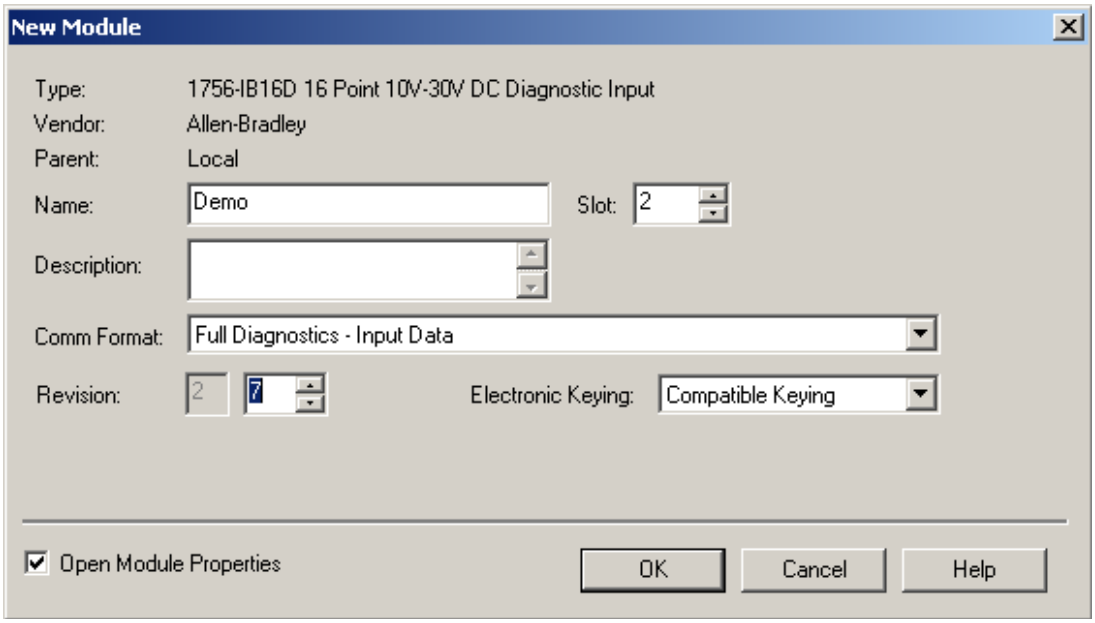
FYI

Module Configuration Wizard

Whenever you add an I/O module, to the system you will go through the Module Configuration Wizard. The Wizard allows you to step through the entire configuration needed for a module. You can access this information later by double clicking on a module in the I/O Configuration folder or through the tag monitor/editor.

With the Logix family, there are no more dip switches or jumpers needed to configure I/O modules. I/O modules are software configured. This saves time when setting up a system. The configuration for all modules is part of the controller's program and is downloaded to the module from the controller; this allows for ease of replacement if an I/O module fails.

6. Enter the ***Name*** and ***Slot*** parameters as shown below. Leave all other fields set to their default values. Click ***OK***.



The Module Configuration Wizard will appear for the 1756-IB16D.

FYI

Comm Format

Determines the data structure for the tags that are associated with the module. Many I/O modules support different formats. Each format uses a different data structure.

Electronic Keying

When you insert a module into a slot of a chassis, the controller compares the information read from the newly inserted module with what the user configured that particular slot to be in their project.

The following data is read and compared:

Vendor, Product Type, Catalog Number, Major Revision, Minor Revision.

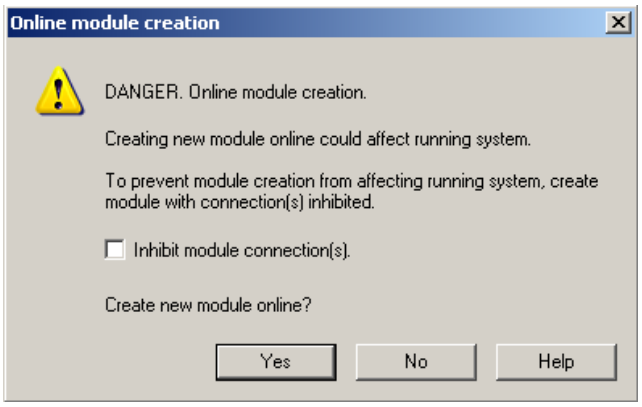
The user may select one of the following module keying options during the initial module configuration:

Exact Match – all of the parameters described above must match or the inserted module will reject the connection.

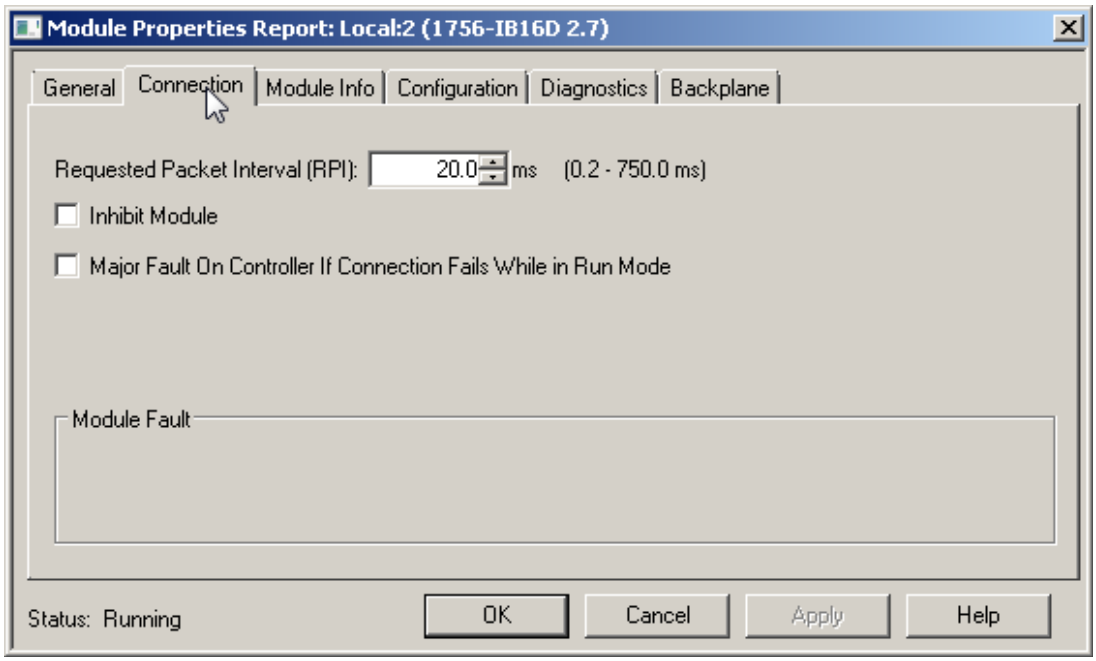
Compatible Module – The following criteria must be met, or else the inserted module will reject the connection: Module Types, Catalog Number, and Major Revision must match and the Minor Revision of the physical module must be equal to or greater than the one specified in the software

Disable Keying – No keying used at all.

7. Select ***Yes*** when the Online Module Creation box appears.



8. Click on the ***Connection*** tab to view the ***Requested Packet Interval*** data.



FYI

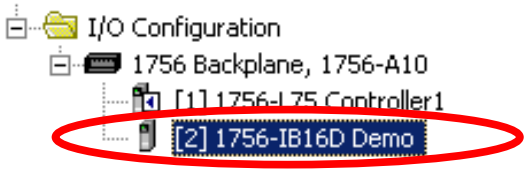
Requested Packet Interval (RPI)

The Requested Packet Interval specifies the period at which data is updated to and from the module. RPIs are configured in milliseconds. The range is .2ms to 750ms.

ControlLogix and 1768-L43 processors allow individual RPI values to be configured whereas 1769-L35E CompactLogix processors treat I/O module connections as if they were rack optimized meaning all 1769 I/O modules must share the same RPI.

9. Click on ***OK*** to close the wizard.

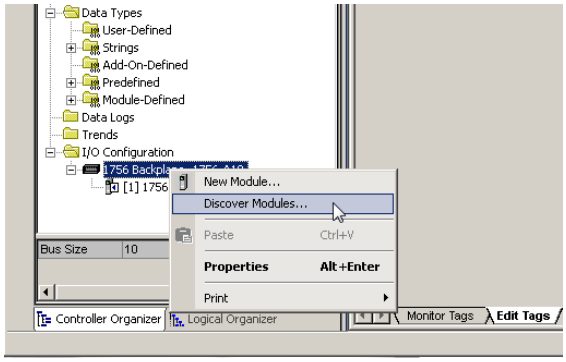
   In the Controller Organizer, the ***I/O Configuration*** folder will show the digital input module in Slot 2. It is possible that you may see a yellow triangle over the I/O module (⚠). In this instance, it indicates a firmware mismatch. Since we are deleting this module in the next step, do not worry about the I/O Fault.



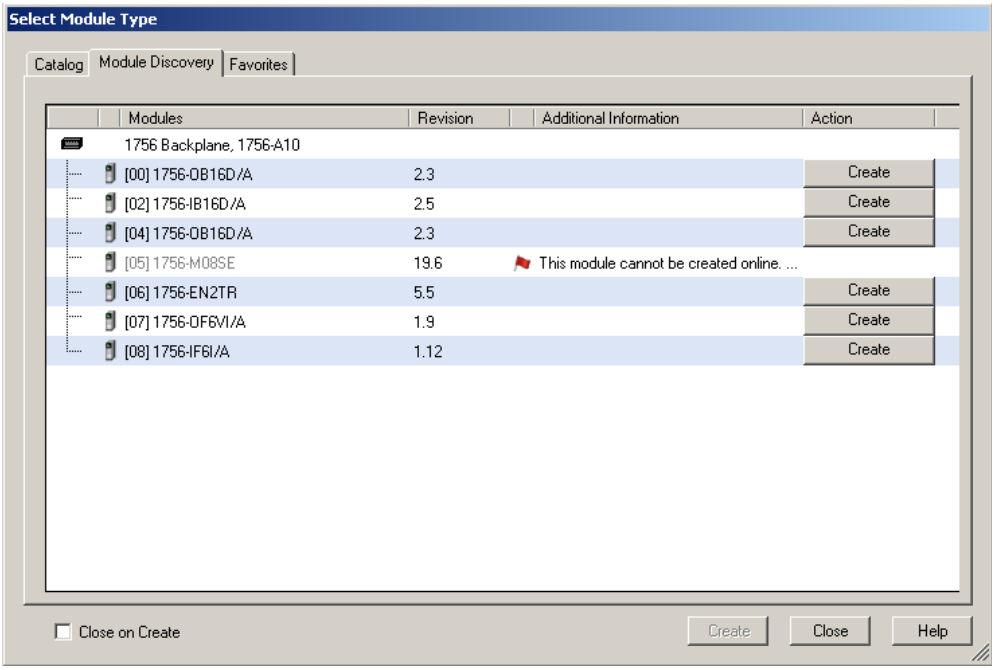
10. Highlight the ***1756-IB16D*** module in the I/O Configuration, and **Press the Delete key on the keyboard**. Click ***Yes*** when prompted to confirm.

Adding ControlLogix I/O Using "Module Discovery"

1. In the ***I/O Configuration Folder***, right click on ***1756 Backplane, 1756-A10*** and select ***Discover Modules***.

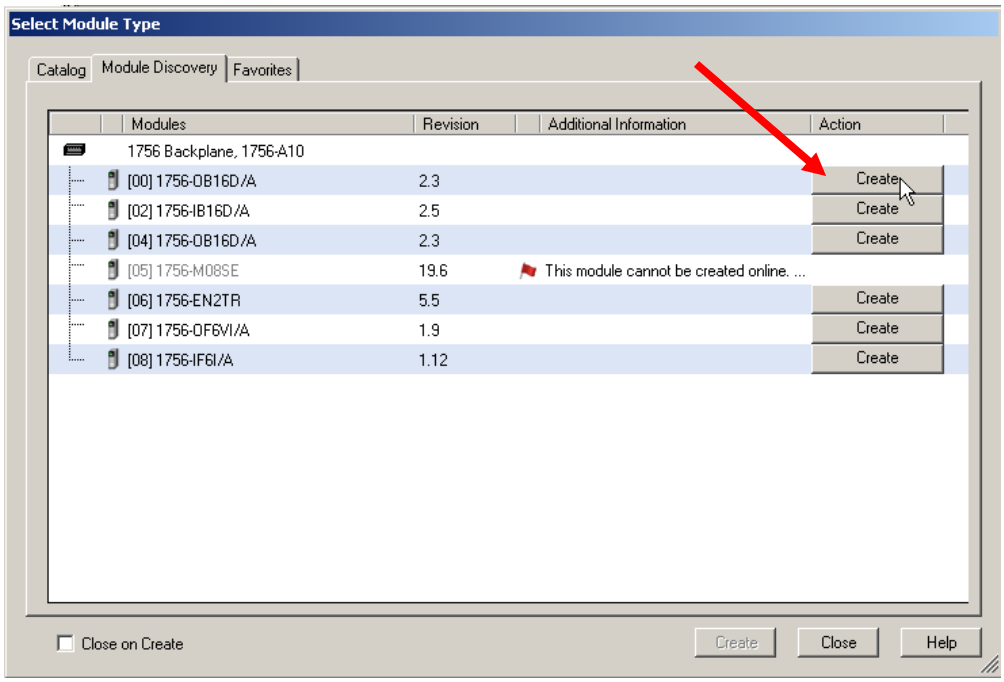


Module Discovery automatically searches the local backplane and will determine each module type and firmware revision. This simplifies the module creation process. Modules that cannot be created online will be grayed out, as shown above with the 1756-M08SE module.

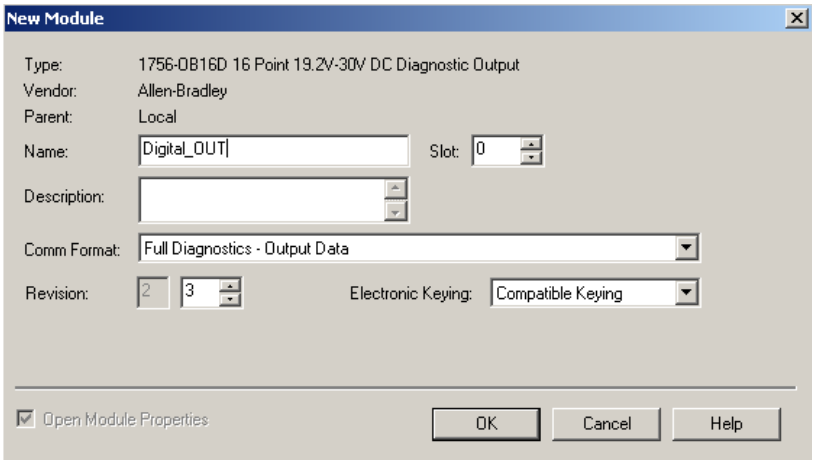


Note: The firmware revisions for the I/O modules in your lab may be different from the above screen shot. This will not affect the execution of the lab since the module discovery feature will automatically set the correct firmware.

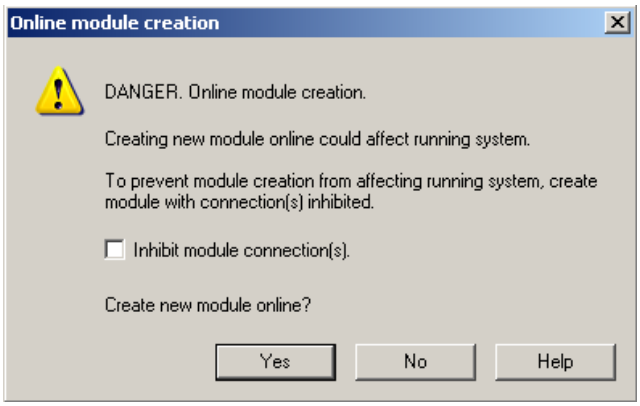
2. On the ***Select Module Type*** window, select the ***Create*** button next to the 1756-OB16D/A module in slot 0.



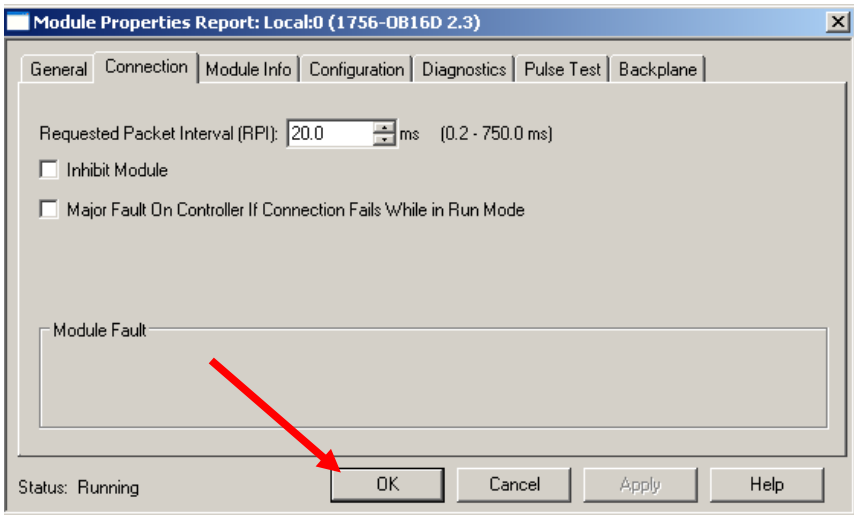
3. Enter the ***Name*** parameter as shown below. Leave all other fields set to their default values. Click ***OK***.



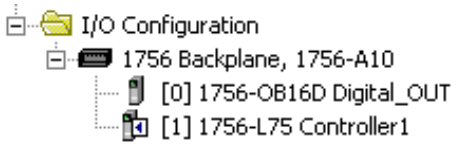
4. Select ***Yes*** when the Online Module Creation box appears.



5. Click ***OK*** on the Module Properties Report to close the window



6. The I/O Configuration should look like the following:

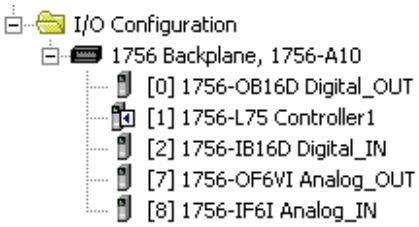


7. Using the ***Module Discovery*** feature, add the 1756-IB16D/A module in Slot 2 to the I/O Configuration. Name the module ***Digital_IN.***

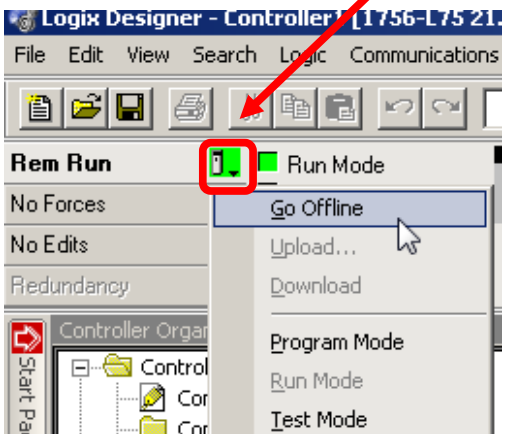
8. Using the ***Module Discovery*** feature, add the 1756-OF6VI/A module in Slot 7 to the I/O Configuration. Name the module ***Analog_OUT.***

9. Using the ***Module Discovery*** feature, add the 1756-IF6I/A module in Slot 8 to the I/O Configuration. Name the module ***Analog_IN.***

10. When you are finished, your I/O Configuration should look like the following:



11. Once you have added all of the modules listed above, click ***Remote Run*** from the controller to get the pull down menu.



12. Select ***Go Offline***.

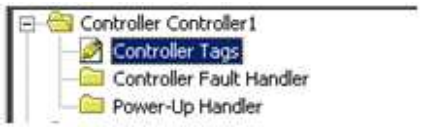
13. ***Save*** the program by clicking on the ***Save*** icon  on the toolbar.

Viewing the ControlLogix I/O Tags

Now that we have configured I/O modules in the project, let's take a look how that information is presented in Studio 5000.

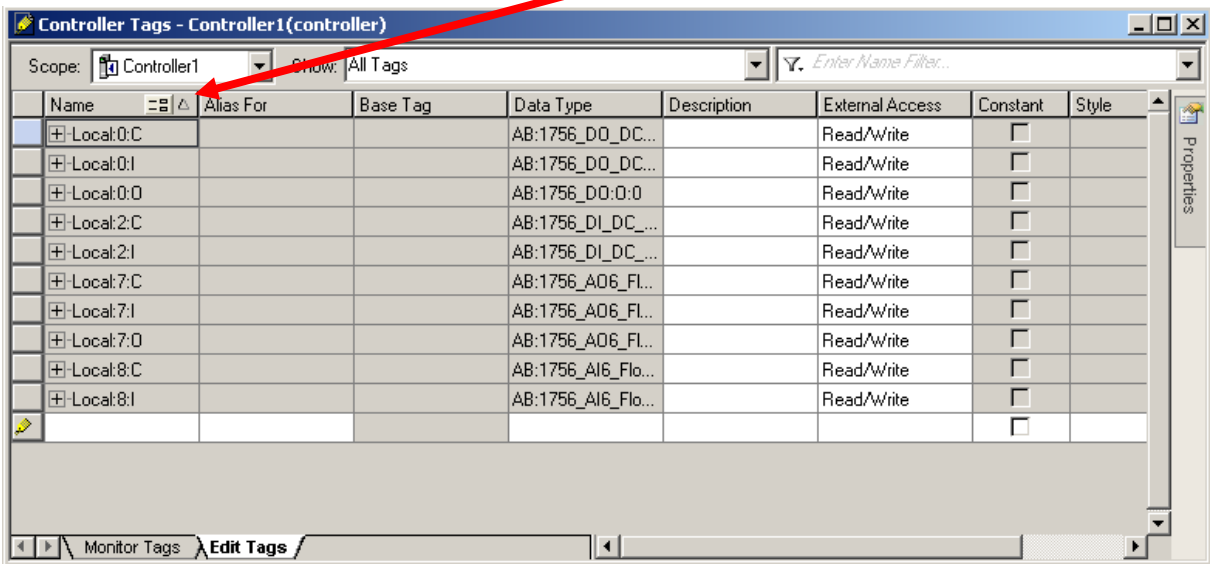
You will continue to use the project already opened.

1. From the Controller Organizer double click on Controller Tags.



The tag editor window will appear.

If necessary, drag to the right to increase the size of the Tag Name field. This will allow you to view the entire Tag Name.



FYI

I/O Address Format

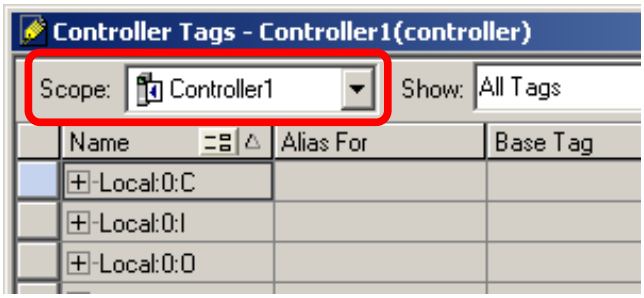
An I/O address follows this format:

`Location` `:Slot` `:Type` `.Member` `.SubMember` `.Bit`

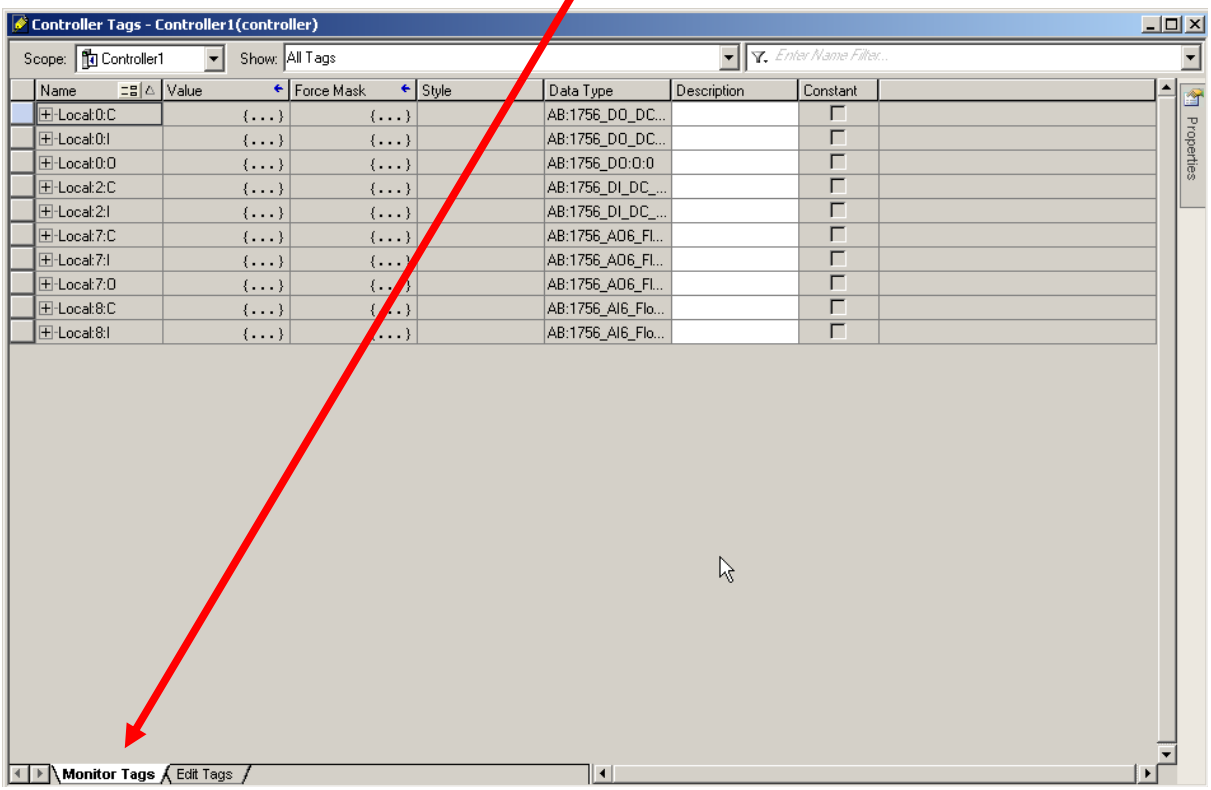
= Optional

Where:	Is:
`Location`	Network location LOCAL = same chassis or DIN rail as the controller `ADAPTER_NAME` = identifies remote communication adapter or bridge module
`Slot`	Slot number of I/O module in its chassis or DIN rail
`Type`	Type of data I = input O = output C = configuration S = status
`Member`	Specific data from the I/O module; depends on what type of data the module can store. • For a digital module, a Data member usually stores the input or output bit values. • For an analog module, a Channel member (CH#) usually stores the data for a channel.
`SubMember`	Specific data related to a Member.
`Bit`	Specific point on a digital I/O module; depends on the size of the I/O module (0-31 for a 32-point module)

You notice by looking in the upper left corner of the tag editor that you are in the Controller Scope. All I/O module tags are created in the Controller Scope.



2. Switch to ***Monitor Tags*** by Clicking on the ***Monitor Tags*** Tab.



The above entries are tag structures for the modules you added. They contain more tags than are actually displayed. Note the + sign next to the tag name, this indicates that you can expand the tag structure to see more information.

Tag Properties Pane:

This pane displays the attributes of the selected tag in the Tag editor or data monitor dialog. The Tag Properties Pane can be expanded by selecting a tag and hovering over the "Properties" icon (located in the upper right corner of the tag database window.

3. Expand and explore the tags for the I/O modules by clicking the **+**.

   What you will find under the Configuration tags, for each module, is all the data, you entered and selected from the Module Configuration Wizard.

4. ***Save*** the program by clicking the ***Save*** icon  in the toolbar.

Assigning Alias Tags

In this section of the lab you will learn about Alias Tags.
You will continue to use the project already opened.

FYI

Aliasing

An Alias tag lets you create one tag that represents another tag.

Both tags share the same value

When the value of one of the tags changes, the other tag reflects the change

Use Aliases in the following situations:

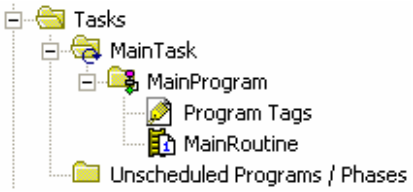
-Program logic in advance of wiring diagrams

-Assign a descriptive name to an I/O device

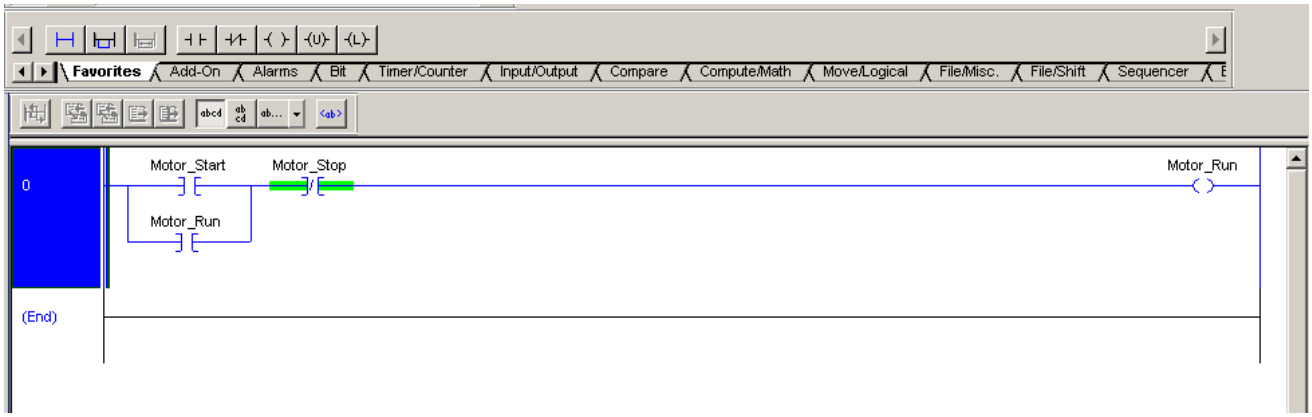
-Provide a simpler name for a complex tag

-Use a descriptive name for an element of an array

1. From the Controller Organizer double click on ***MainRoutine***.



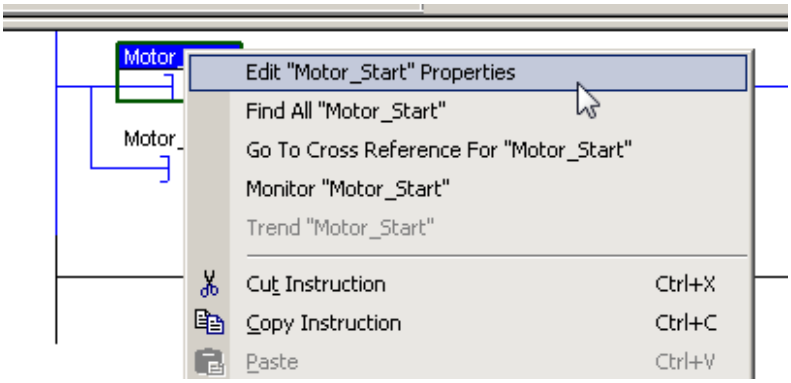
The ladder editor appears as shown below:



In the last part of the lab we added I/O modules to the project. Now it's time to Alias the tags in the program to the I/O Modules.

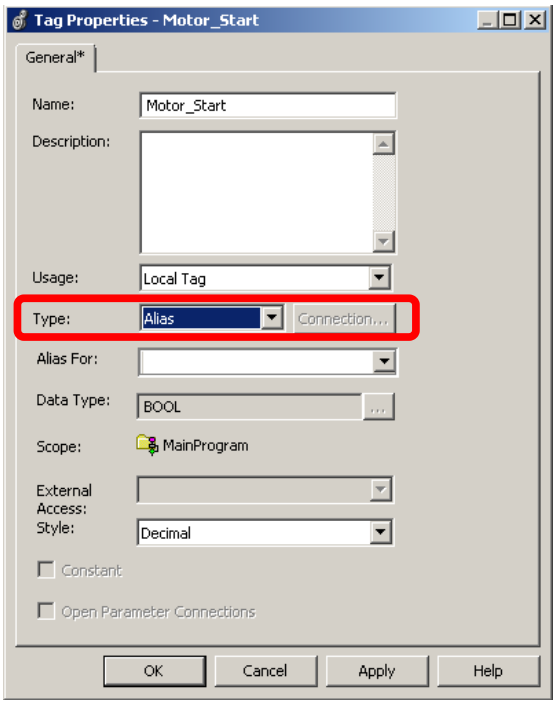
- Motor_Start will be Aliased to input point zero of the 1756-IB16D in slot two.
- Motor_Stop will be Aliased to input point one of the 1756-IB16D in slot two.
- Motor_Run will be Aliased to output point zero of the 1756-OB16D in slot zero.

2. Right click on the tag Motor_Start and select Edit 'Motor_Start' Properties.



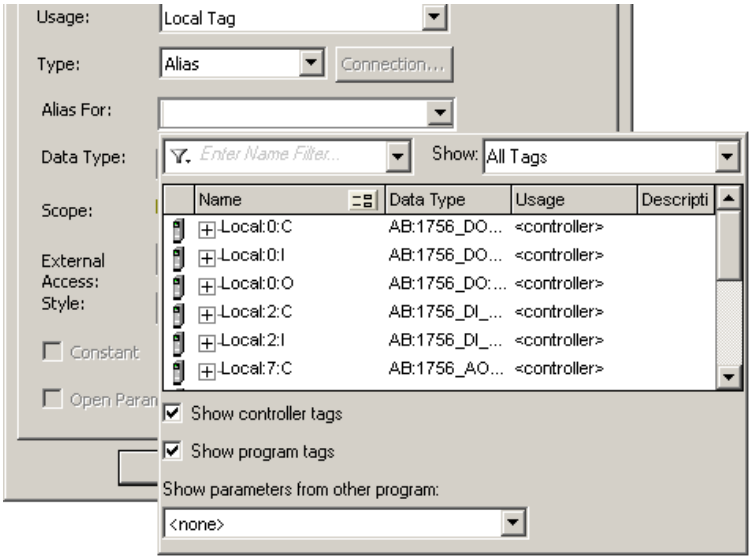
The ***Tags Properties*** window for Motor_Start will appear. Currently the tag is defined as a Base tag.

3. Select ***Alias*** as a type and notice that the ***Tag Properties*** window changed.

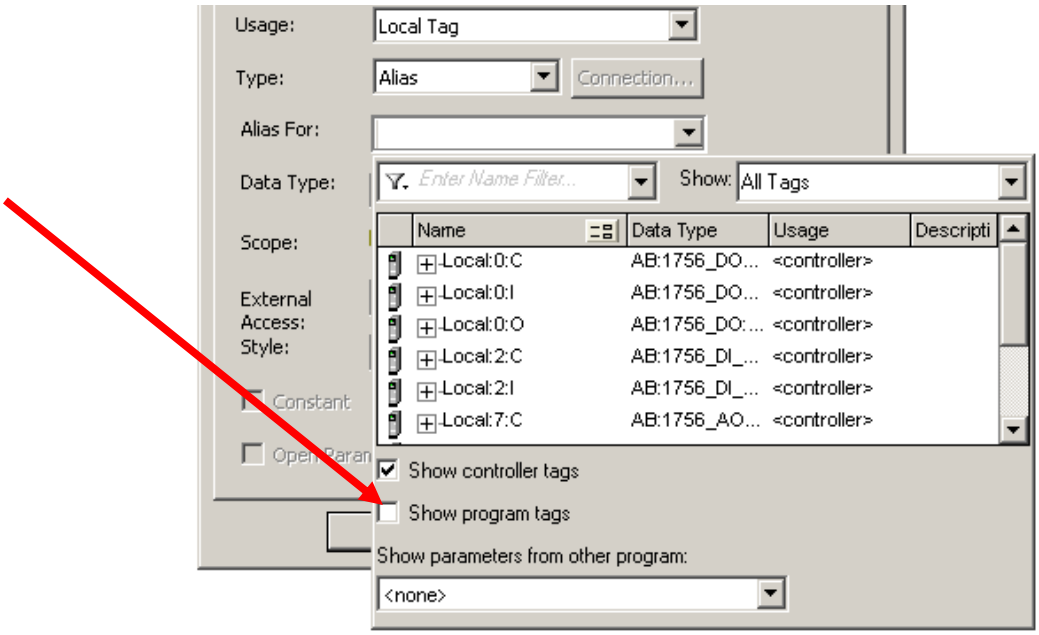


4. Click on the down arrow for ***Alias For***.

The tag browser appears. The browser shows both Controller and Program Scope Tags. You will need to select your address from controller scoped tags.

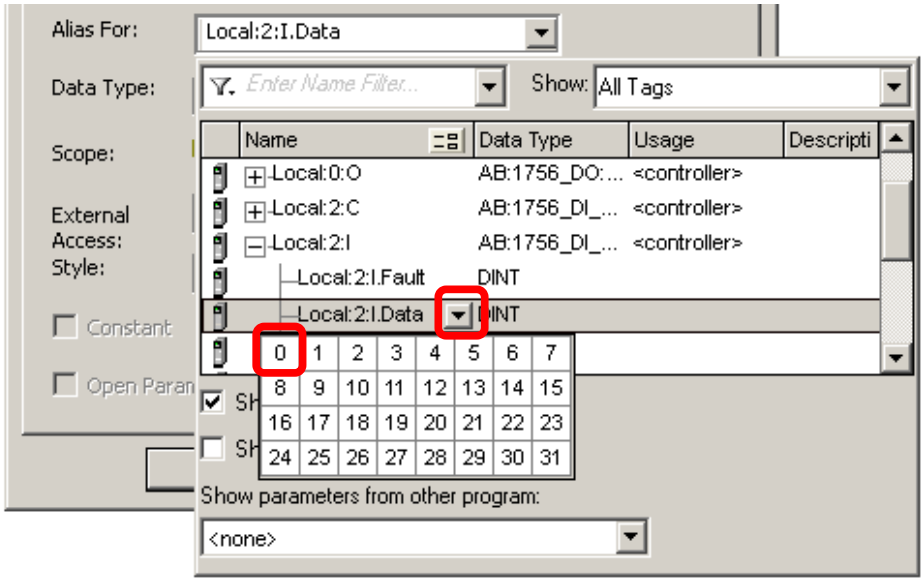


5. Uncheck the ***Show Program Tags*** checkbox to deselect Program Scoped Tags.
   The view on the screen will change to view only your Controller Scoped Tags



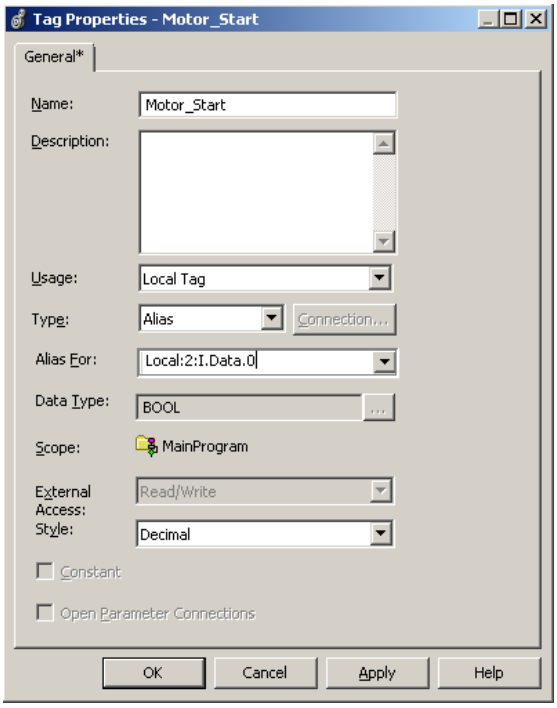
6. Expand ***Local:2:I*** by clicking on the **+** sign and select ***Local:2:I.Data.***

7. Click the **down arrow** for ***Local:2:I.Data*** as shown below.
   This will open the table of data points for the 1756-IB16D module.



8. Select ***0*** from the table.

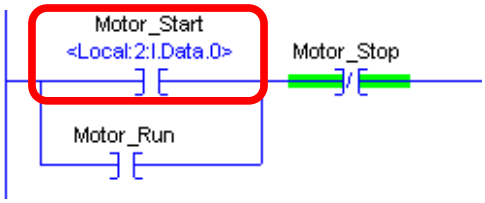
   When you select ***0*** from the tag browser the window will close. ***Tag Properties*** will now appear as follows:



   Motor_Start will now be aliased to ***Local:2:I.Data.0***. This is the 1756-IB16D in Slot 2.

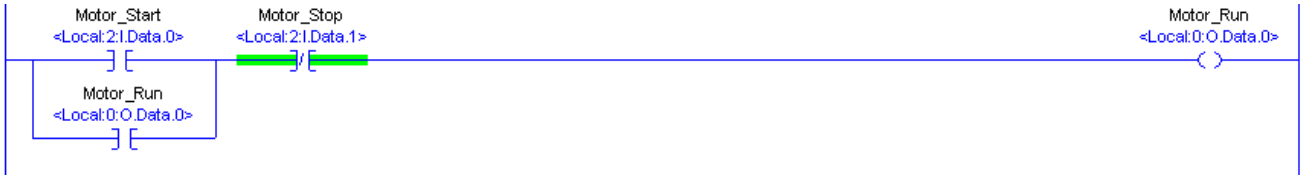
9. Click ***OK*** to close and apply the changes to the tag ***Motor_Start***.

Motor_Start has been Aliased to Local:2:I.Data.0. This means that the tags are equivalent to one another in code. It is much easier to understand Motor_Start than Local:2:I.Data.0.



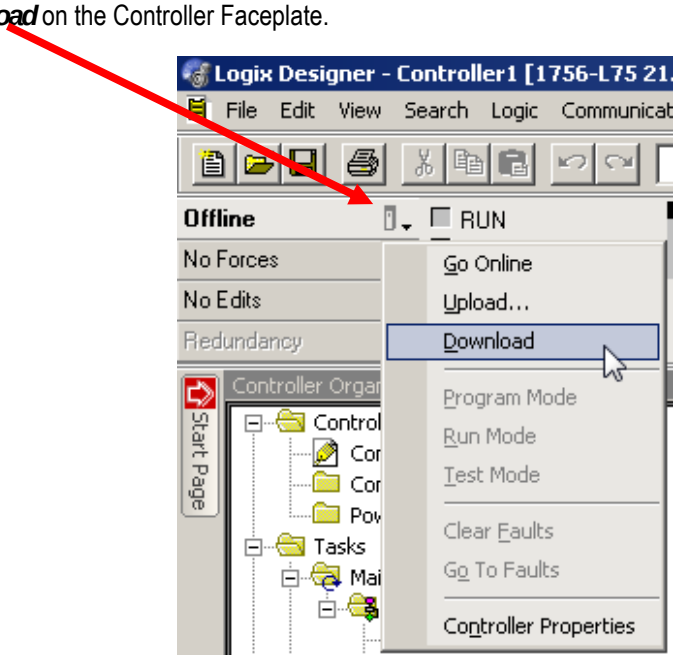
10. Using the previous steps, alias the remaining two tags.
    - Motor_Stop = Local:2:I.Data.1
    - Motor_Run = Local:0:O.Data.0

11. When you are finished the ladder code should appear as follows:



12. ***Save*** the program by clicking on the ***Save*** icon  on the toolbar.

13. Click ***Download*** on the Controller Faceplate.



14. When prompted to confirm the download, press ***Download***.

15. Click ***Yes*** when prompted to change the controller mode to Remote Run.

Congratulations! You have Completed Section 4. Please move on to Section 5.

Section 5: Testing Your Logic Program

This lab section should take roughly 5 minutes to complete.

Objective:

In this lab you will verify the operation of your program.

FYI

I/O Mapping

For the lab there are a group of push buttons on the Demo Box. The push buttons are mapped as follows:

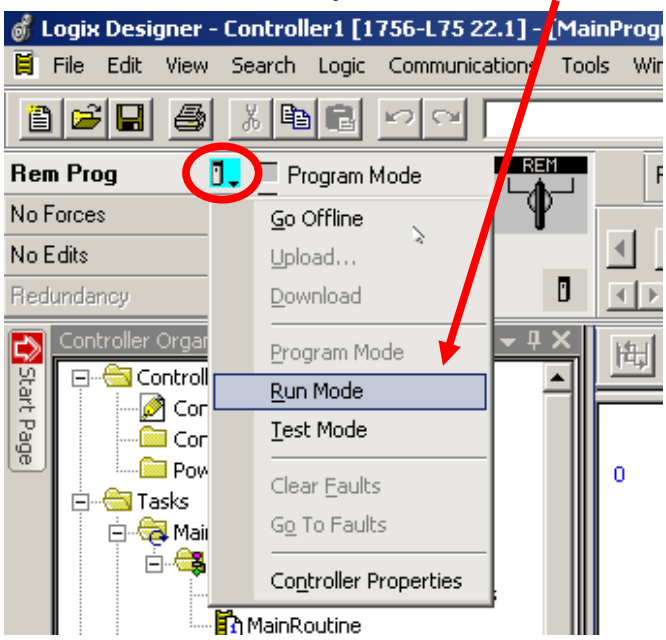
Motor_Start = DI0

Motor_Stop = DI1

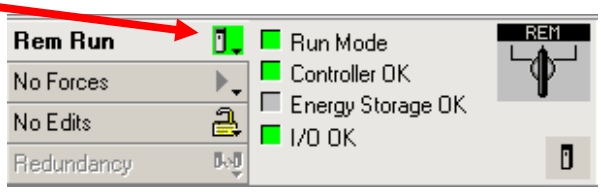
Motor_Run = DO0

Switching the Controller into Run Mode and Testing the Program

1. If not already in run mode, click the ***Controller Faceplate*** and select ***Run Mode***.

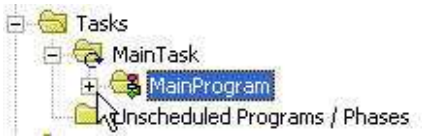


The controller will go into run mode. This can be verified by looking at the Run LED on the controller. It should now illuminate green. It can also be verified through Studio 5000 by viewing the controller faceplate.

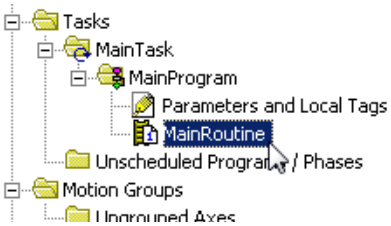


Notice that this is a replica of your controller's status.

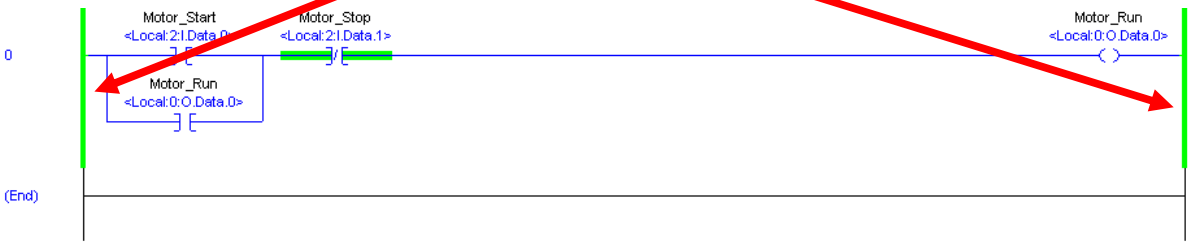
2. From the Controller Organizer expand the ***MainProgram*** by clicking on the "**+**".



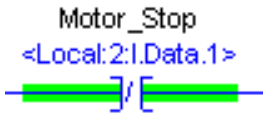
3. Double-click on the ***MainRoutine*** to open the ladder editor.



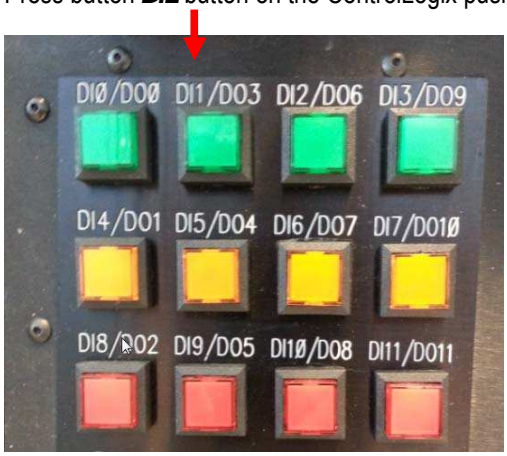
You will now see the ladder logic. Notice the green power rails on both sides of the ladder. This indicates you are online and the routine is executing.



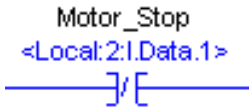
Notice that the XIO instruction Motor_Stop is green. This means that this instruction is in the 'true' or 'on' state. This is because the Motor_Stop Pushbutton is not pressed.



4. Press button ***DI1*** button on the ControlLogix pushbutton panel.

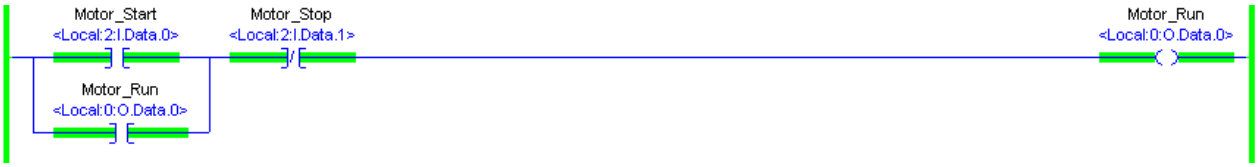
5. 

This correlates to the XIO instruction for Motor_Stop. Notice it's no longer be green. This is because the instruction is no longer true.

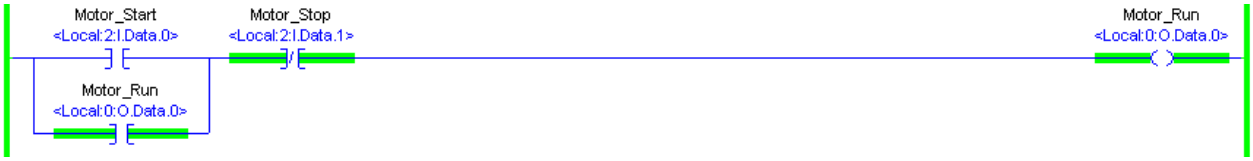


6. Press button ***DI0*** (Motor_Start).

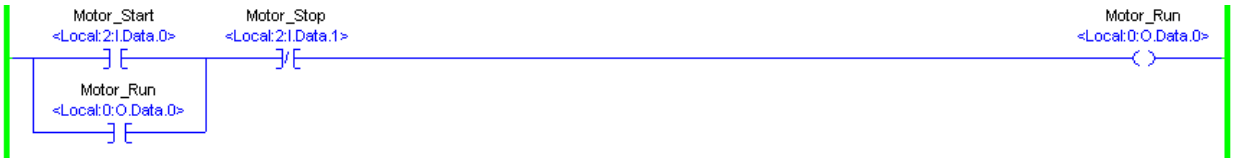
The XIC instruction will become true and turn green. Motor_Run will energize (turn green). And the pilot light DO0 on your lab station will illuminate.



7. Verify that output ***DO0*** (Motor_Run) stays illuminated when you release pushbutton ***DI0*** (Motor_Start).
   The ladder logic you have just written is a simple 3-wire control or motor start/stop seal-in circuit.



8. Press pushbutton ***DI1*** (Motor_Stop) and verify that output ***DO0*** (Motor_Run) turns off.



Congratulations! You have Completed Section 5. Please move on to Section 6.

Section 6: Adding Logic and Tags Online

This lab section should take roughly 15 minutes to complete.

Objective:

In this lab we will explore online editing. You will:

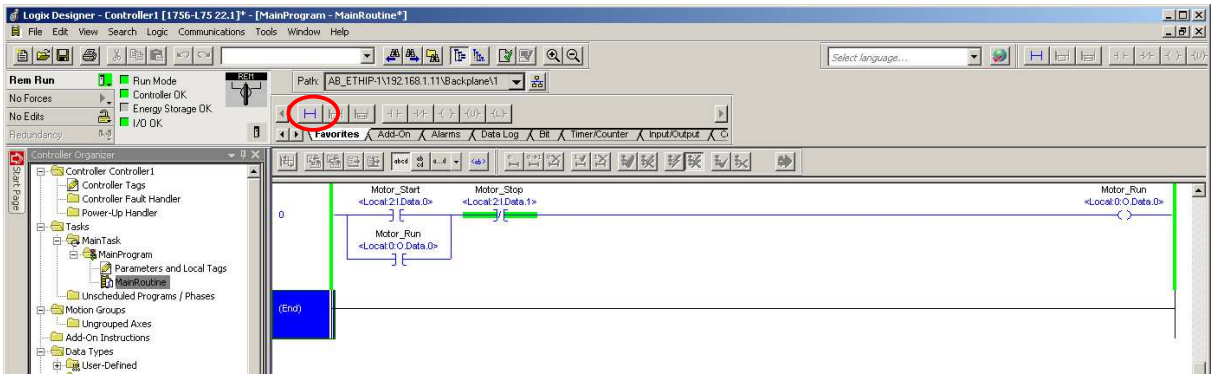
- Add a MOV instruction
- Add a timer to the logic and its execution will be based on the motor running
- Add ladder logic to reset the timer when the motor is stopped.

You will continue to use the project already opened.

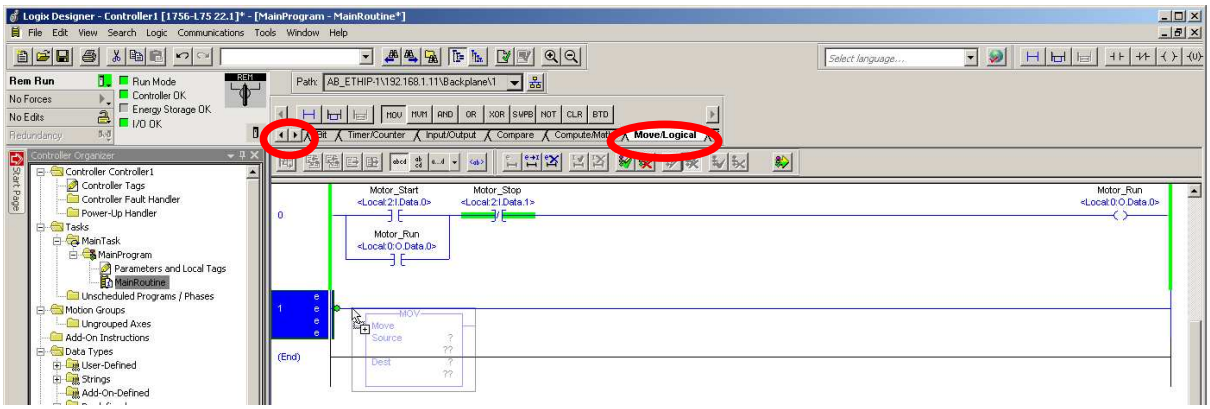
Adding a MOV Instruction to the Logic

1. **Click** on Rung 0 of the ***MainRoutine*** in the ladder editor.

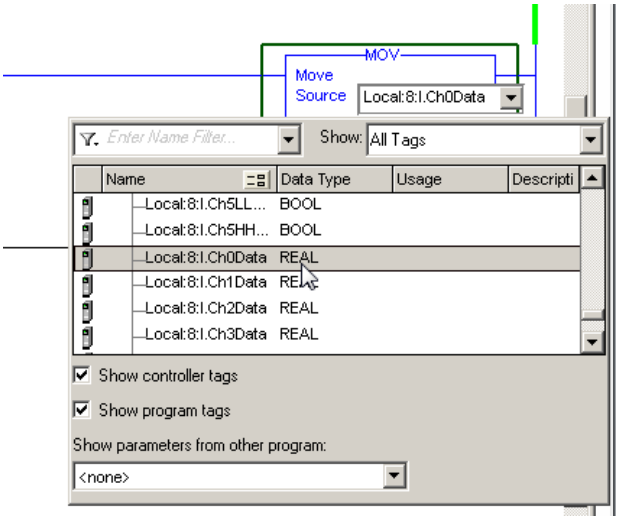
2. **Add** a rung by clicking the rung button  on the toolbar.



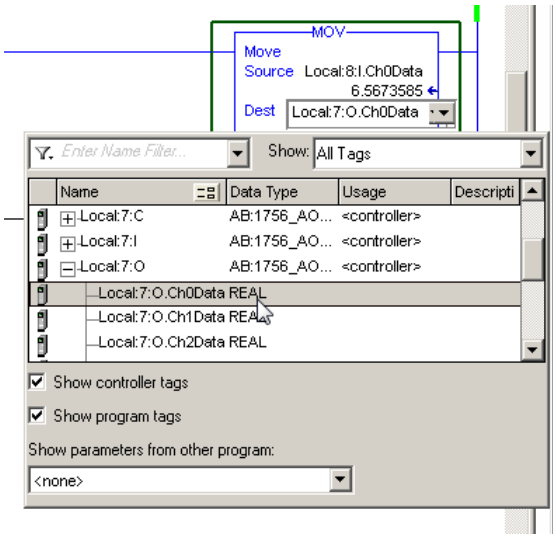
3. Use the ***scroll buttons*** if necessary to scroll to the ***Move/Logical*** instruction group tab in the instruction toolbar. Under the **Move/Logical** category tab, click and drag a ***MOV*** instruction to the new rung.



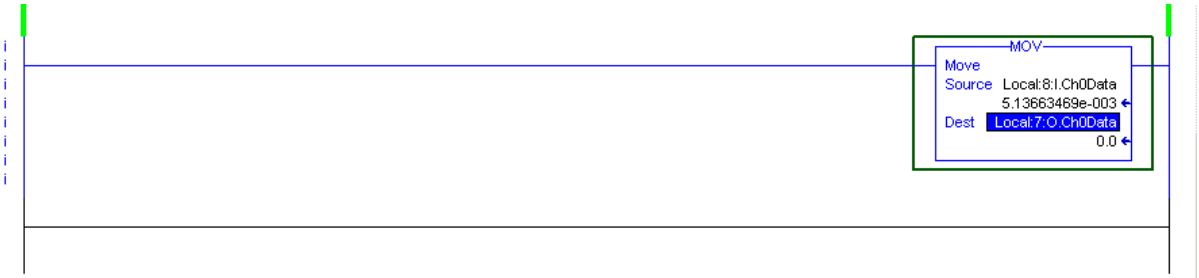
4. Double click the '***?***' by the source. Select ***Local:8:I.Ch0Data*** by double-clicking the tag. You will have to scroll down to near the end to find the Channel data tags.



5. Double click the '***?***' by the destination. Select ***Local:7:O.Ch0Data*** by double-clicking the tag. You will have to scroll down to find the Channel data tags.

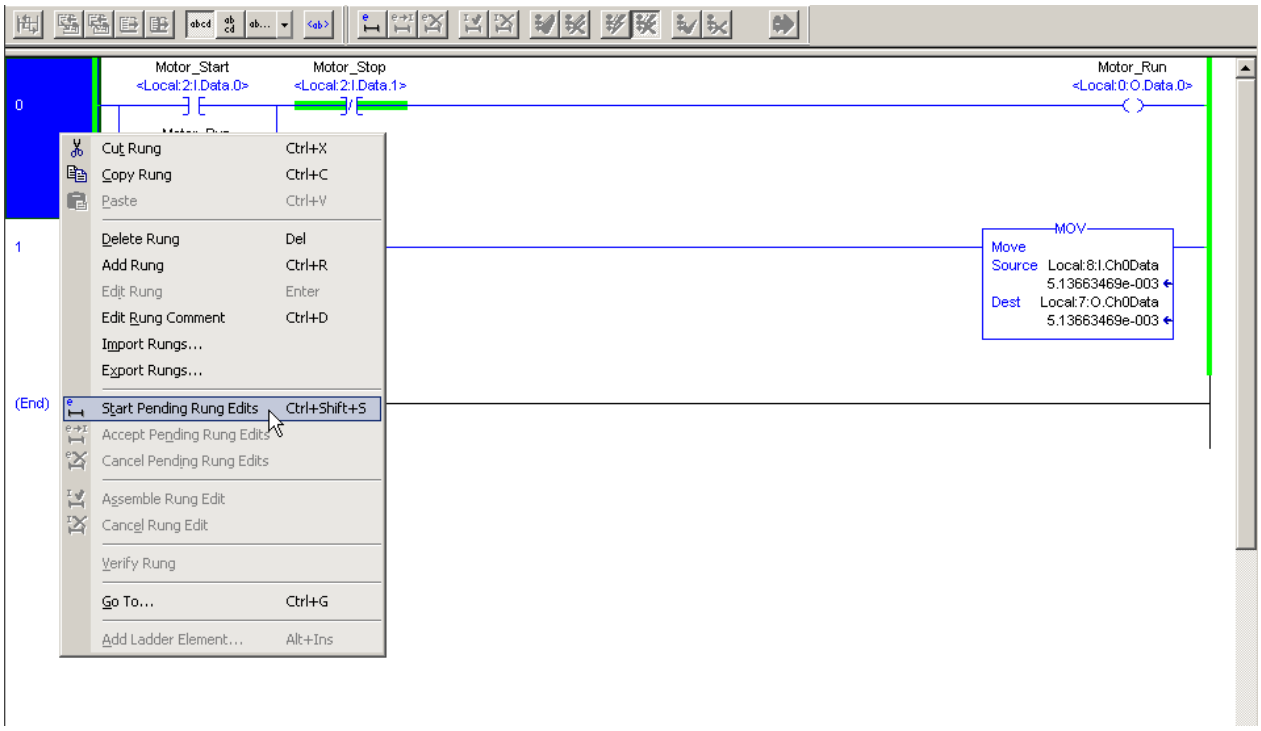


6. The rung should look like the following.

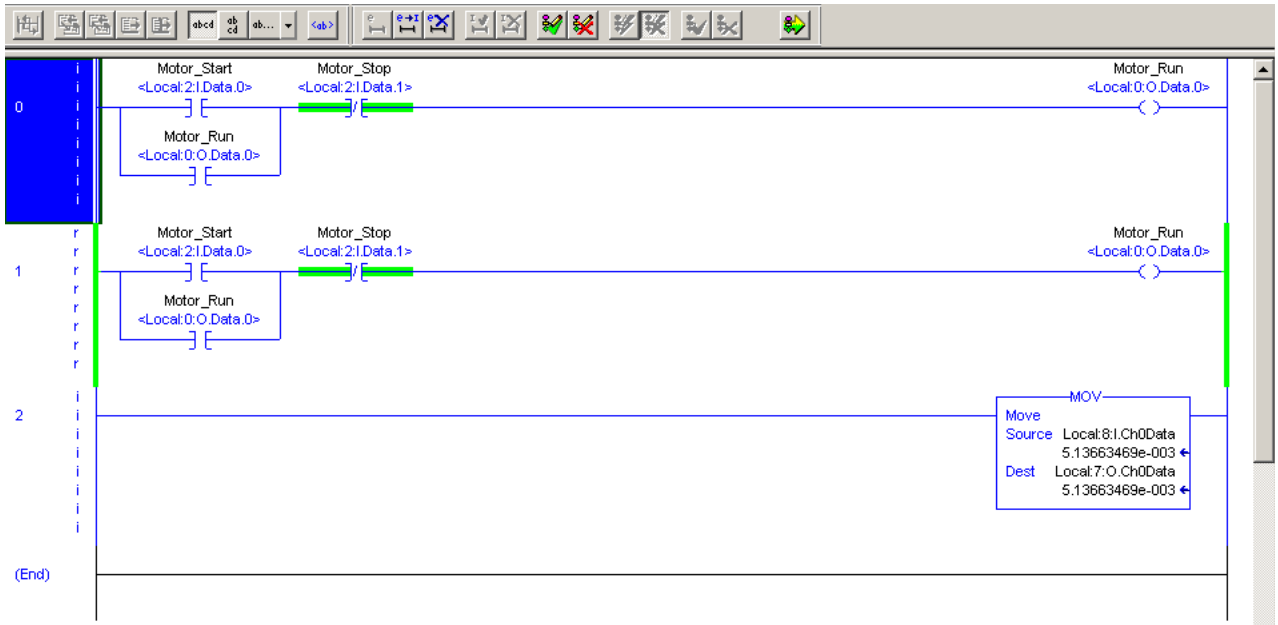


Adding the Timer to the Logic

1. Select rung 0. Right click in the ***blue highlighted area*** to the left of rung zero and select ***Start Pending Rung Edits***.

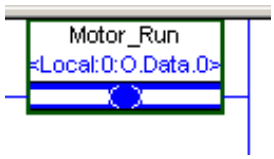


2. The ladder editor will now look similar to the following:

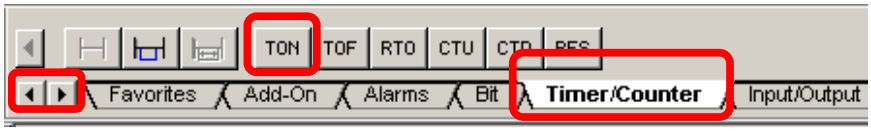


The rung with the lower case 'i's on the power rails is the rung you will perform the edits on.

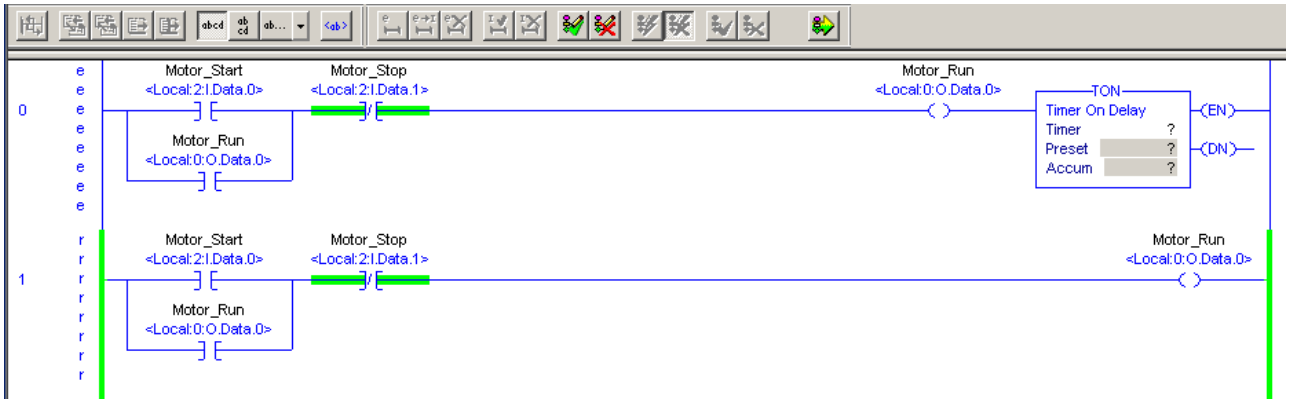
3. Click the ***OTE*** instruction so it becomes highlighted.



4. From the ***Instruction Toolbar*** click on the ***Timer/Counter*** tab, click the **Timer On (TON)** icon



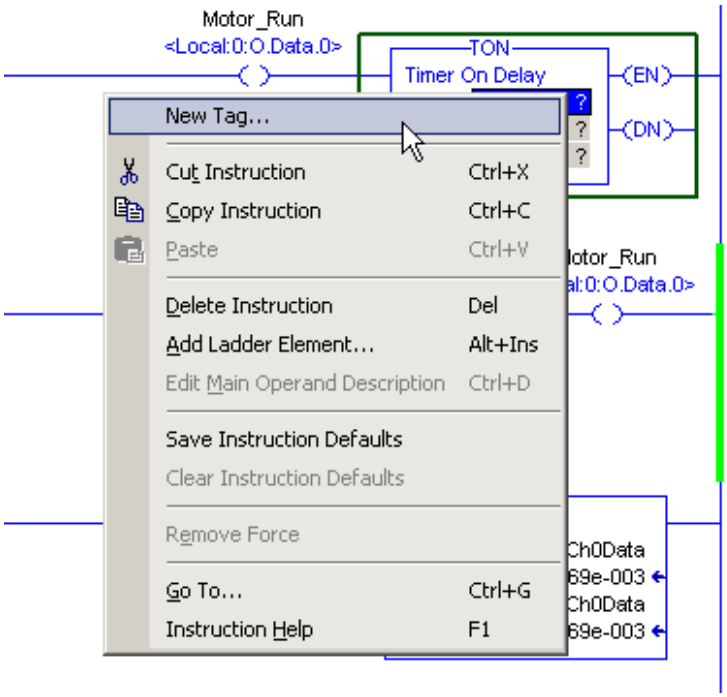
A timer is inserted into the code to the right of the OTE instruction.



FYI

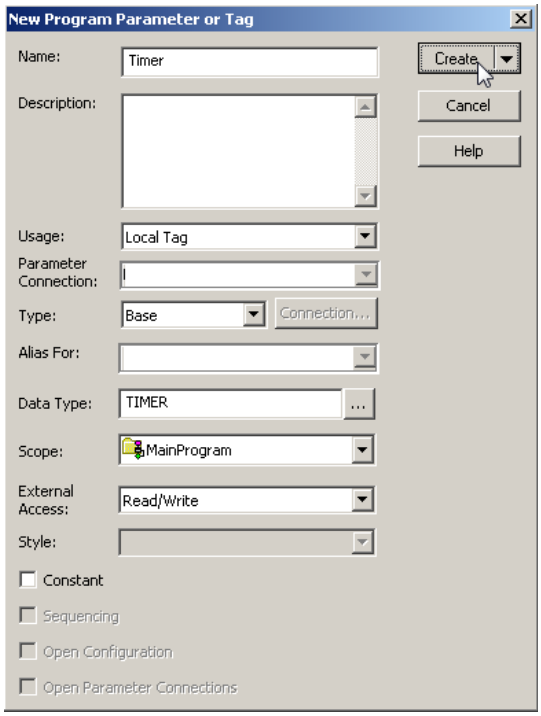
> In ***RSLogix 5000*** you can string output instructions together. You do not have to create branches.

5. On the timer instruction right click in the ***blue area*** next to the word Timer and select ***New Tag***.

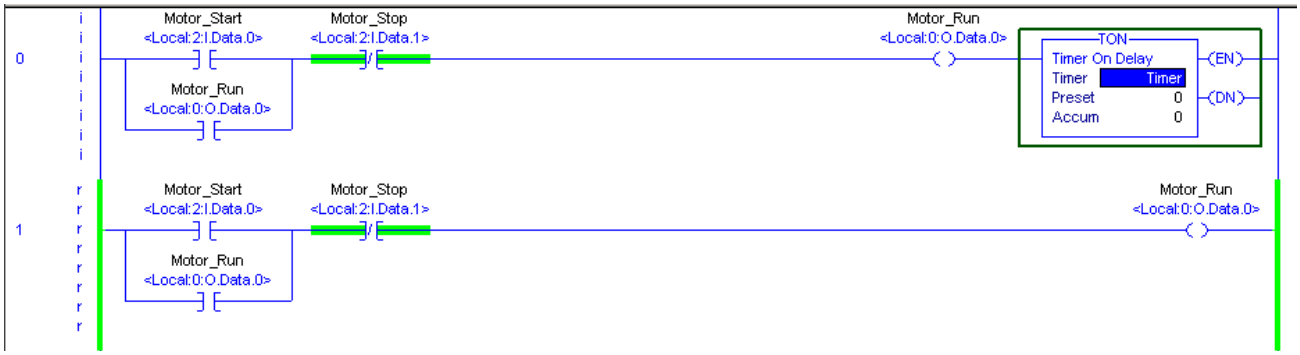


The New Tag window appears. Notice that the Data Type is already set to **TIMER**. This is because you are creating a tag in the timer instruction.

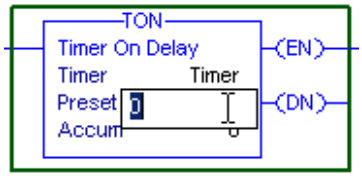
6. In the Name field enter '***Timer***' then click ***Create***



7. Verify that the tag has been created in the timer instruction as shown below:



8. Double-click on the ***0***, in the timer instruction, next to the word ***Preset***.

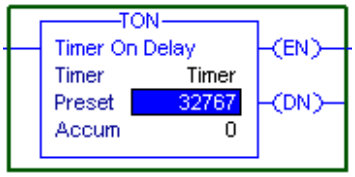


9. Enter a value of ***32767***.

FYI

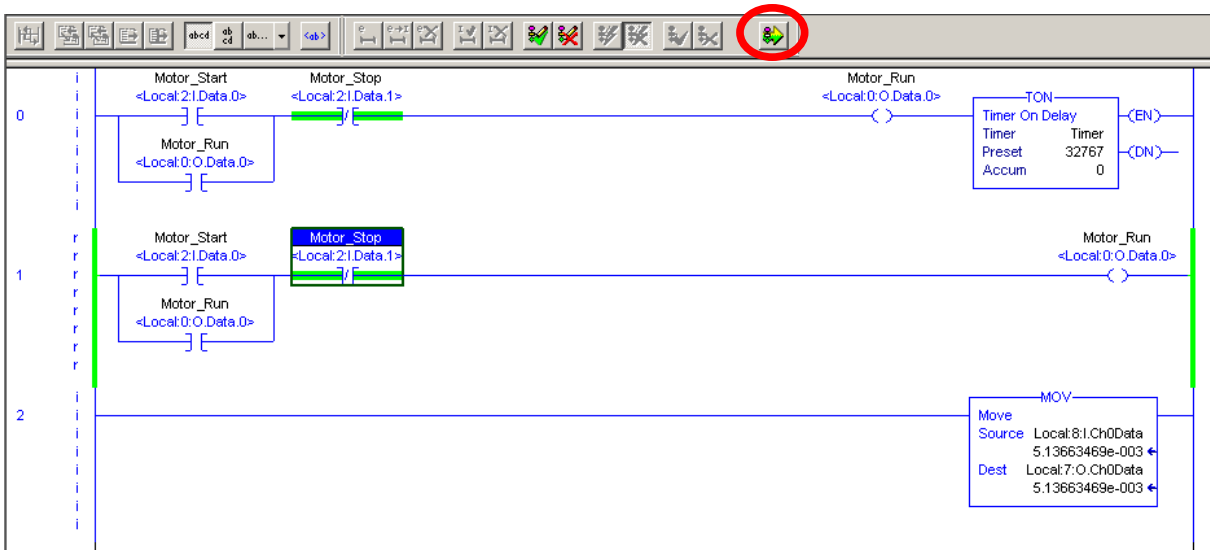
> In Logix the Timer Preset is a 32-bit DINT which means the maximum value for your timers can be: *2,147,483,647*

10. Press ***Enter***. Your TON instruction should now appear as shown below.

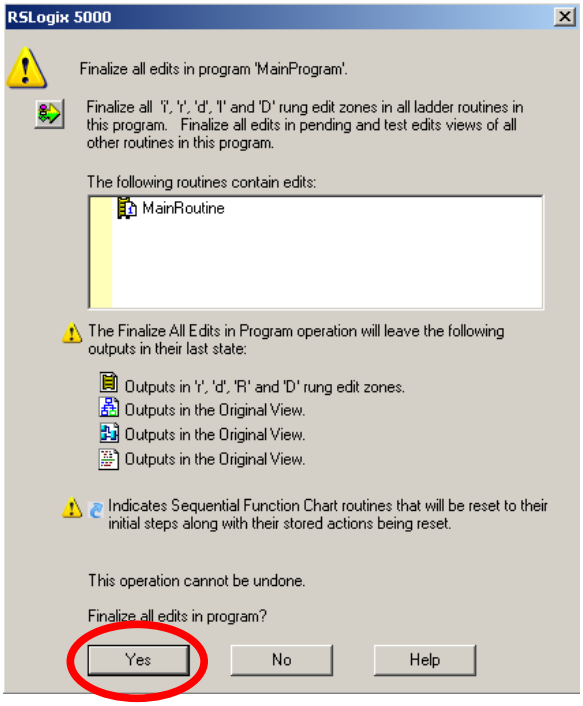


Your Preset value is now 32767 milliseconds (= 32.767 seconds). Leave the accumulated value set to zero. You are now ready to verify the edits you made.

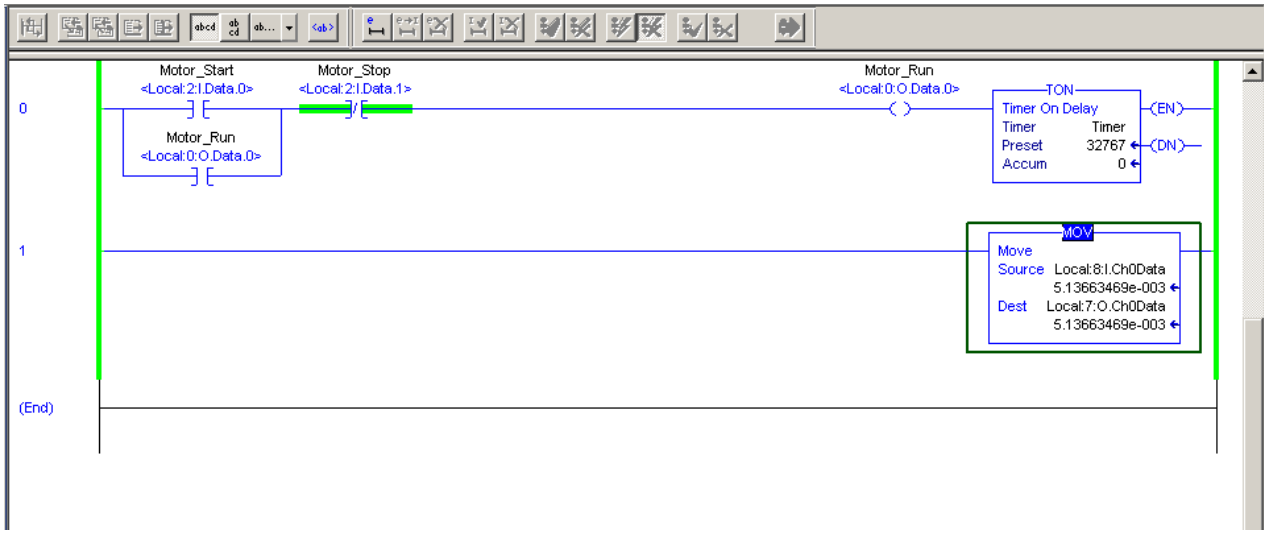
11. Click on the ***Finalize All Edits*** icon



12. When asked to finalize all edits click on ***YES***.



The ladder editor will now appear as follows:



Testing Your Logic

1. Press the ***DI0*** (Motor_Start) pushbutton.
2. Verify that ***DO0*** (Motor_Run) illuminates and the Timer instruction starts incrementing.
3. Now, press push button ***DI1*** (Motor_Stop).
4. Verify that ***DO0*** turns off and the Timer resets.
5. Turn the ***AI0*** potentiometer to 5.
6. Verify that the ***AO0*** meter reads 5 Volts.
7. Turn the ***AI0*** potentiometer to MAX.
8. Verify that the ***AO0*** meter reads 10 Volts.

Congratulations! You have Completed Section 6. Please move on to Section 7.

Section 7: Creating and Running a Trend

This lab section should take roughly 5 minutes to complete.

Objective:

In this lab we will explore the built-in trending capabilities of Studio 5000.

In this Lab you will:

- Create a trend to watch the Timer instruction's accumulated value.

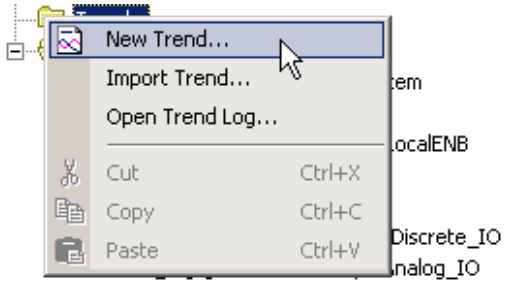
This will be done online with the program from the previous Lab.

FYI

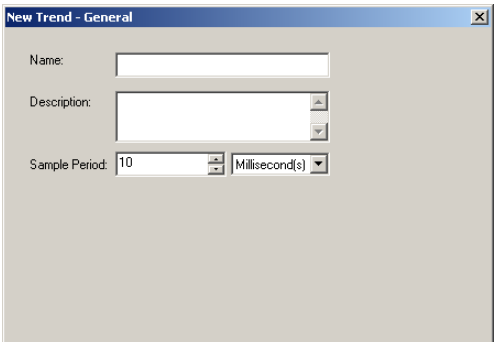
> **Trending**
>
> Basic Trending in Studio 5000 allows you to view data sampled over a time period in a graphical display. Data is sampled at a periodic rate that is configurable from 10 milliseconds to 30 minutes. Studio 5000 will allow you to create a trend and save it as part of your project file.
>
> Basic Trending has these constraints: you can trend data elements of type BOOL, SINT, INT, DINT, and REAL, you are limited to sampling eight unique data elements in a single trend.

Creating and Running a Trend

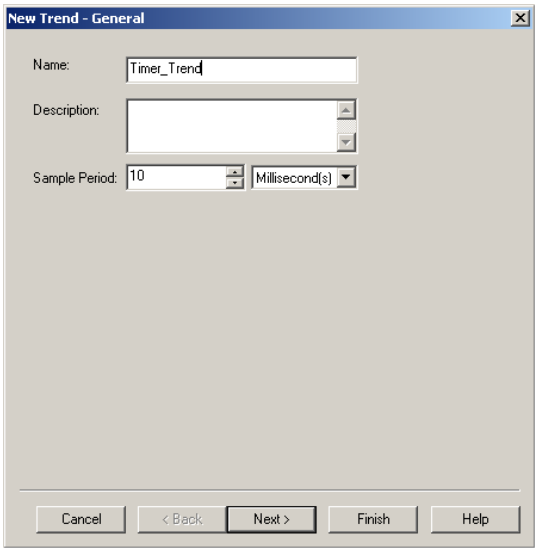
1. From the Controller Organizer, right click on ***Trends*** and select ***New Trend***.



The **New Trend** window appears.

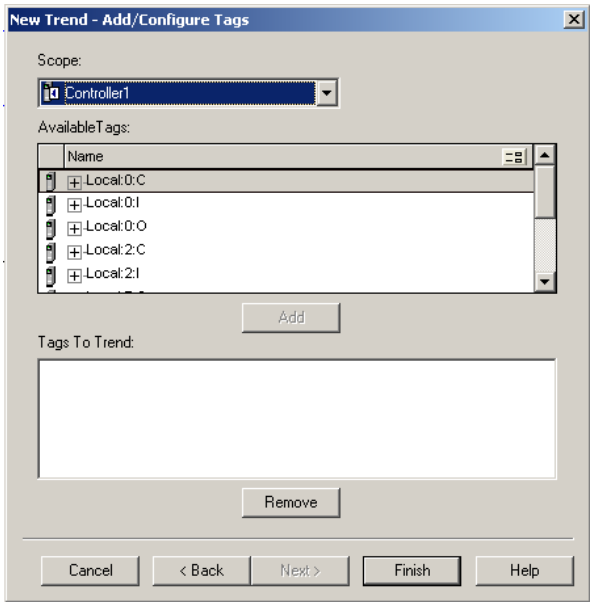


2. In the ***Name*** field enter '***Timer_Trend***'.

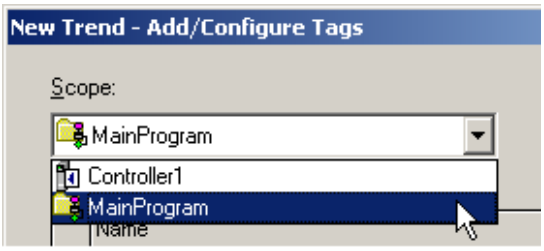


3. Click ***Next***.

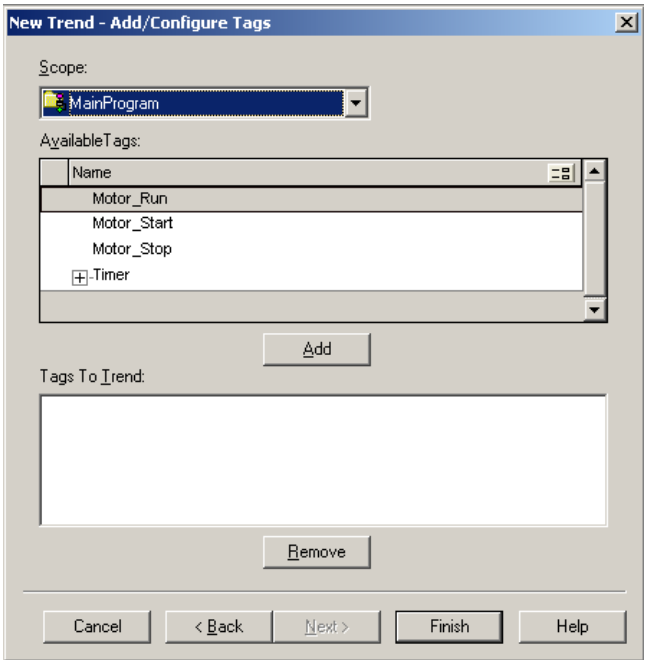
   The New Trend Add/Configure Tags window appears.



4. We want to trend the timer accumulator value. When you added the timer the tag was created in the Program Scope, so we must select the ***MainProgram*** tags as shown below:

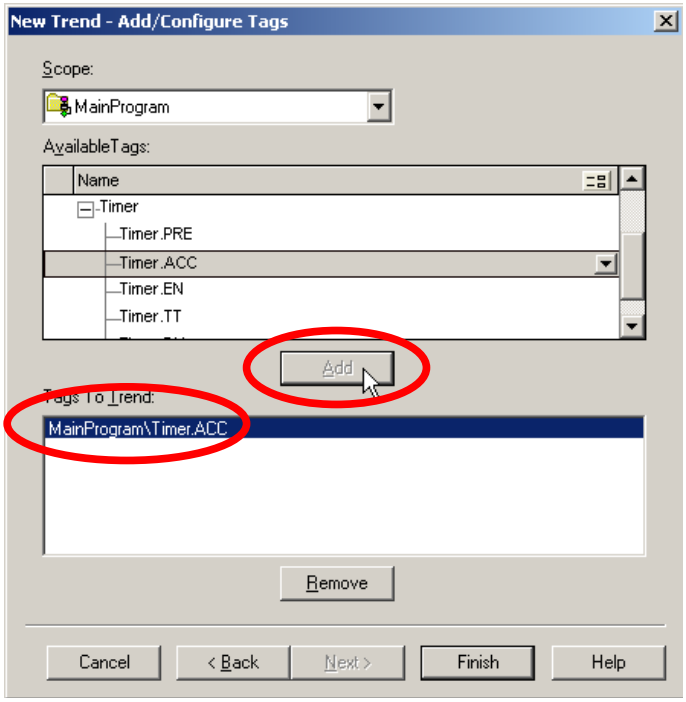


Now only the tags for the ***MainProgram*** are shown.



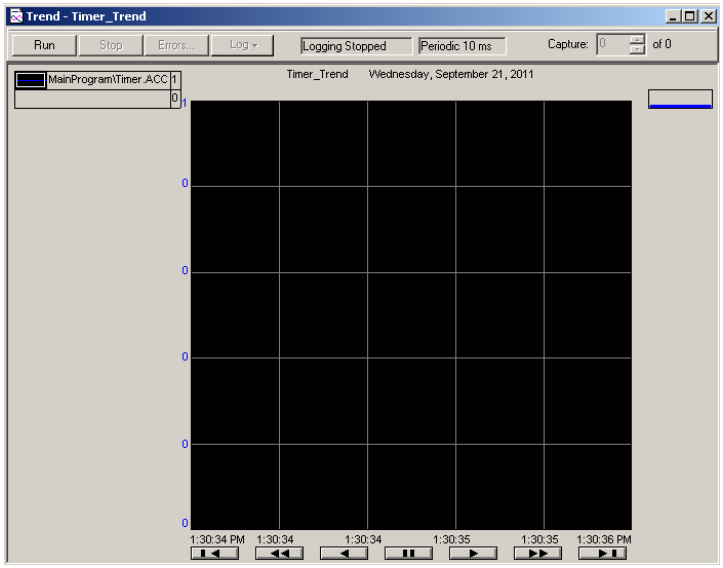
5. Expand the ***Timer*** tag by clicking on the **+**.

6. Select ***Timer.ACC*** and then click the **Add** button. This will add the tag Timer.ACC to the ***Tags To Trend*** list.

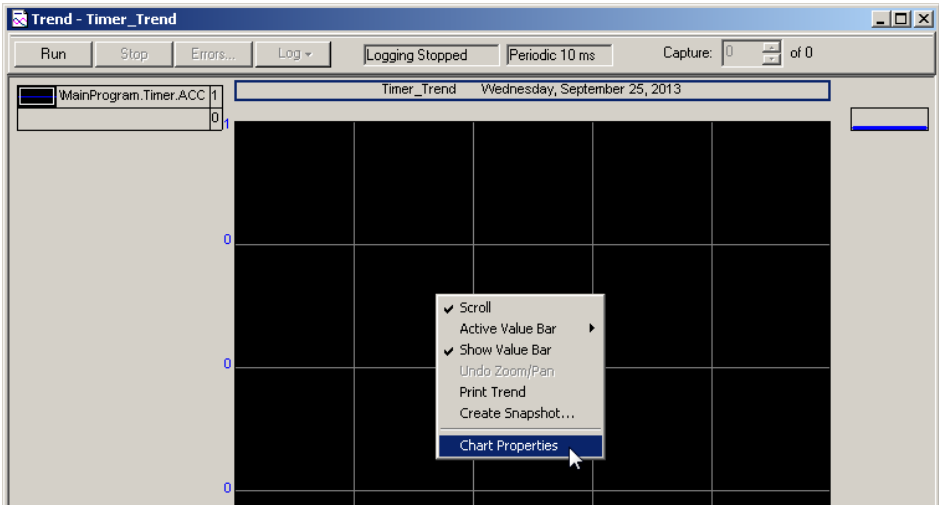


7. Click on ***Finish***.

   The Trend window will now appear.



8. *Right click* on the Trend graph background and select ***Chart Properties.***

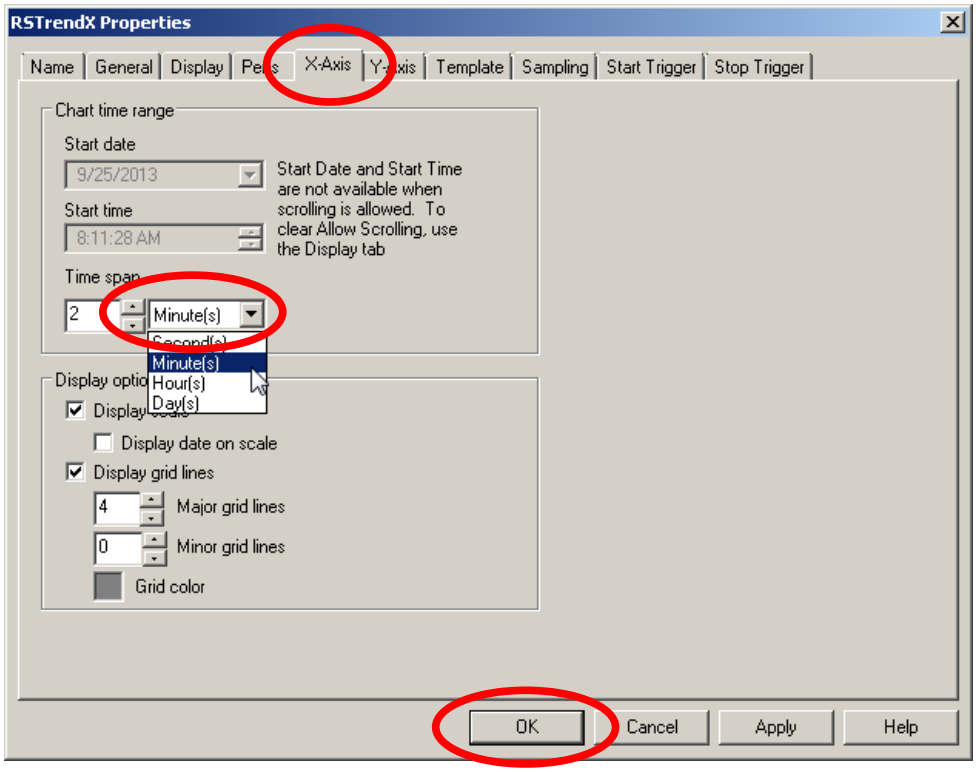


   The RSTrendX Properties window will now appear.

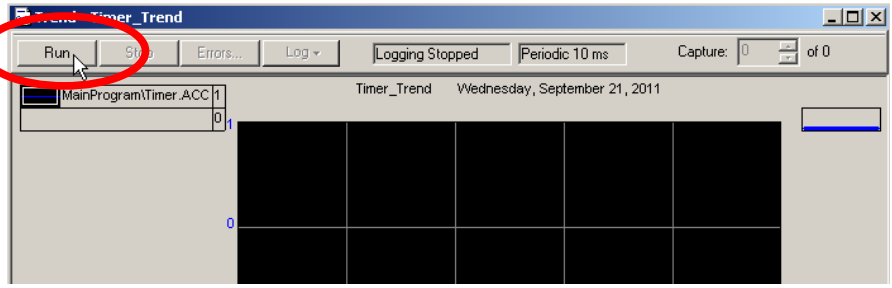
9. Click on the ***X-Axis*** tab.

10. Change the Chart time range - **Time span** from Second(s) to ***Minute(s)*** .

11. click ***OK.***

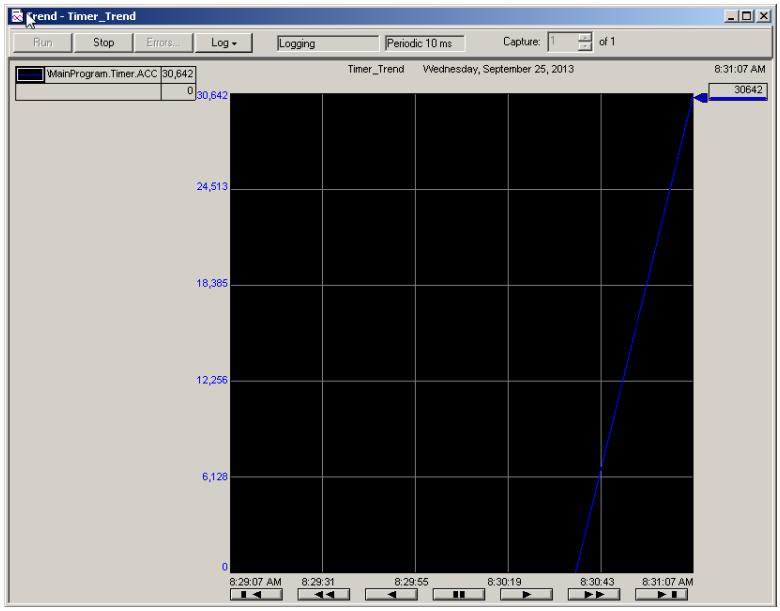


12. Start the trend by clicking on the ***RUN*** button located toward the upper left of the Trend dialog box.



13. Start the Timer in the program by pressing the **DI0** pushbutton on your lab station.

14. Verify that you see the Trend begin capturing the data of the Timer.ACC as shown below:



15. Try pressing the **DI1** pushbutton and watch the trend.

16. When you are finished investigating the trend, click ***Stop*** and ***close*** the trend window.

Congratulations! You have Completed Section 7. Please move on to Section 8.

Section 8: (Optional) Creating and Using User Defined Types (UDT)

This section should take about 10 minutes to complete.

Objective:

This lab section covers creating and using custom data structures.

- Create a User Defined Type (UDT)
- Create a tag from a UDT
- Use the tag in an instruction
- Use the tag monitor/editor to see the tag

Creating User Defined Types

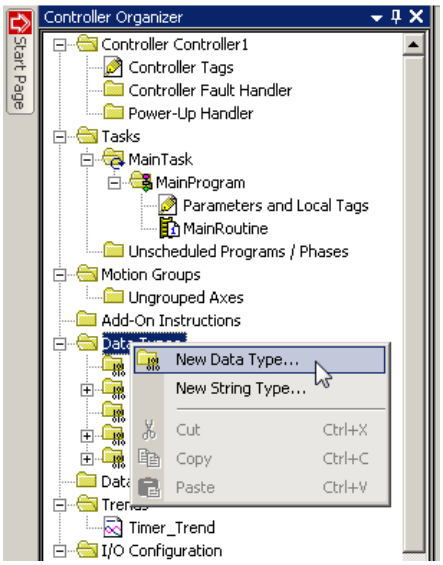
In this section of the lab you will create a custom User Defined Type (UDT).

FYI

> **What is a UDT and what is it good for?**
>
> A UDT is good for organizing related data into a single place. A UDT allows a single tag to hold multiple data fields call members. Each member can be given a unique name to describe the data it holds. The members are accessed by the main tag name, followed by a period, followed by the member name.

Continue to use the project already open.

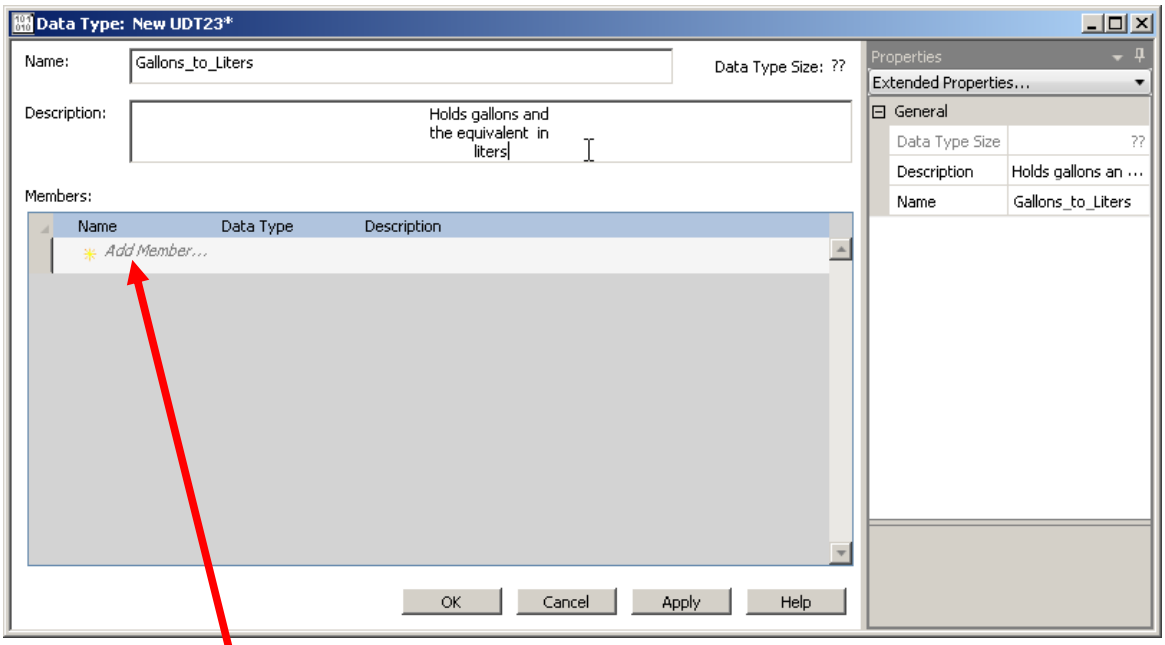
1. *Right click* ***DataTypes*** in the Controller Organizer and select ***New Data Type….***



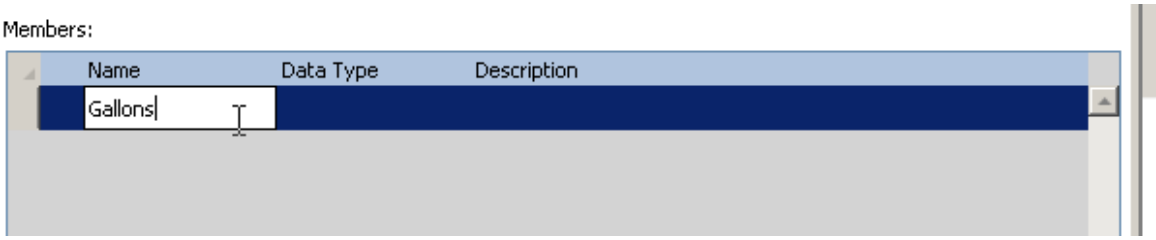
A new **Data Type** window will appear.

2. Fill in the Name field with "***Gallons_To_Liters***" as shown.

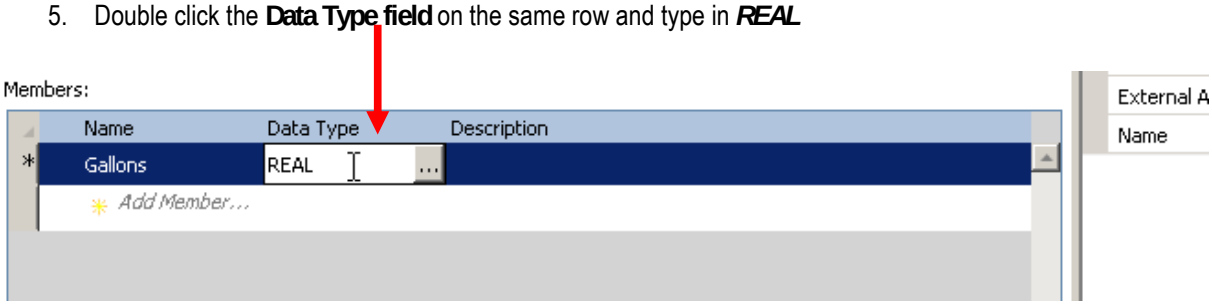
3. Fill in the description field with "***Holds gallons and the equivalent in liters***" as shown.



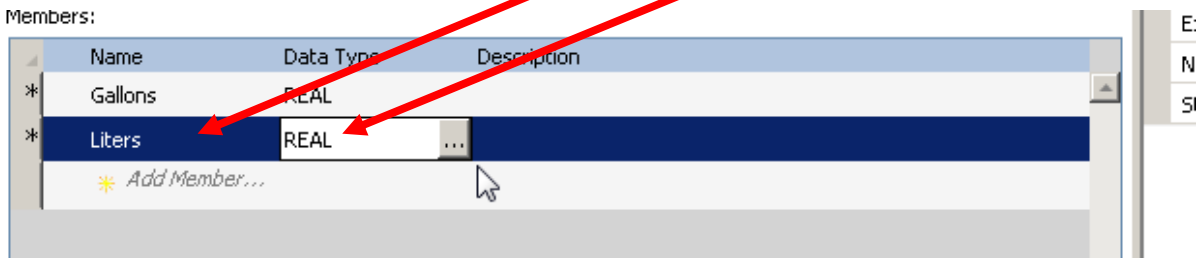
4. Click on **"Add Member…"** and type in ***Gallons***



5. Double click the **Data Type field** on the same row and type in ***REAL***

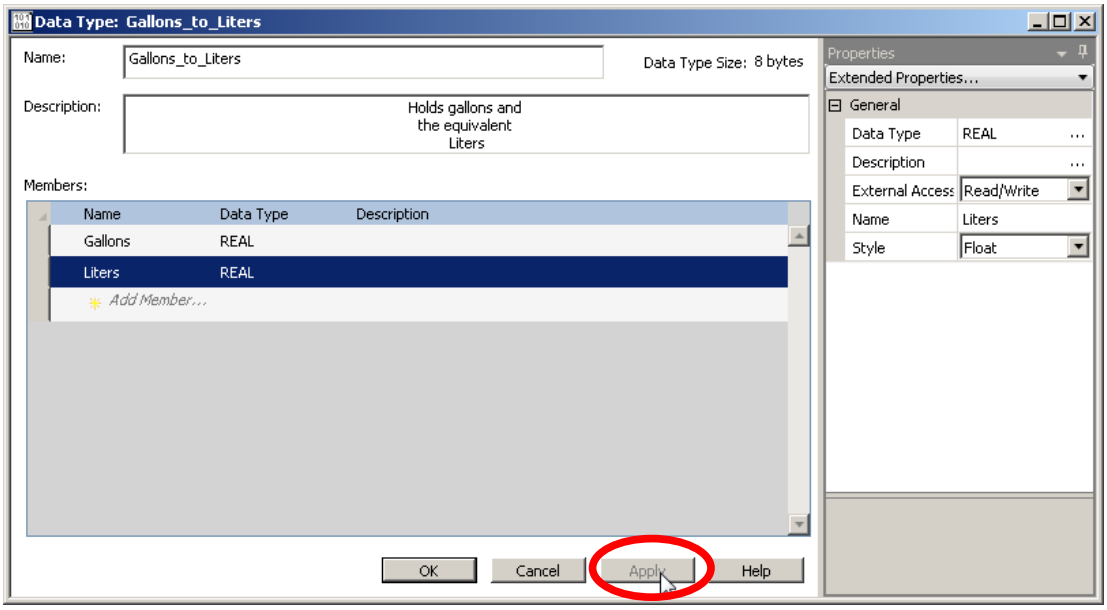


6. Follow the same steps to enter the next row for "***Liters***" and "***REAL***" as shown.



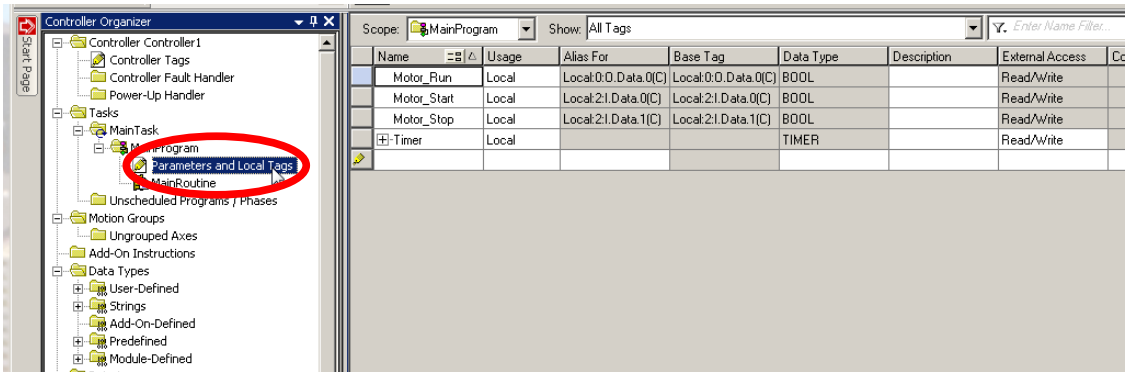
7. Click ***Apply.***

The window should appear as shown.



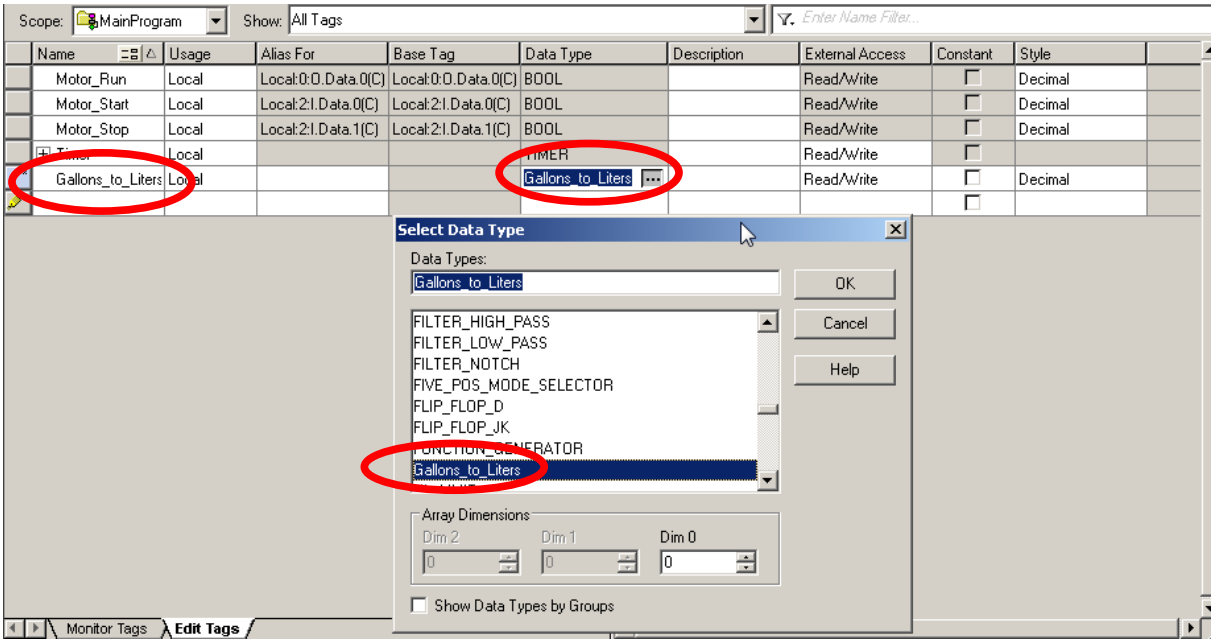
8. Click ***OK*** to close the window.

9. Double click on ***Parameters and Local Tags*** under the MainProgram as shown to open the tag window.



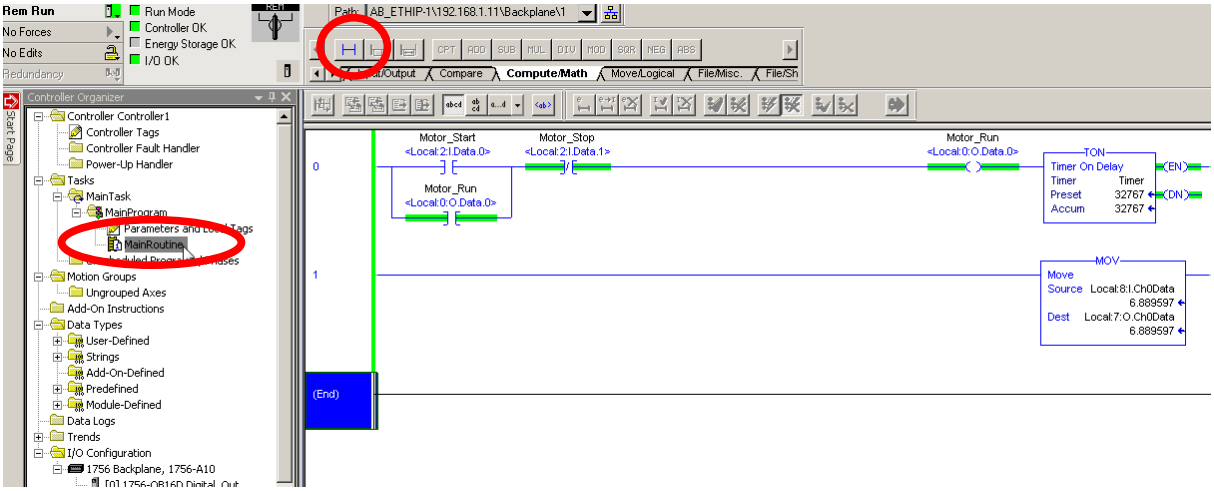
10. On the blank row, fill in "***Gallons_to_Liters"*** for the tag Name.

11. On the same row, select "***Gallons_to_Liters"*** for the Data Type as shown and click ***OK.***



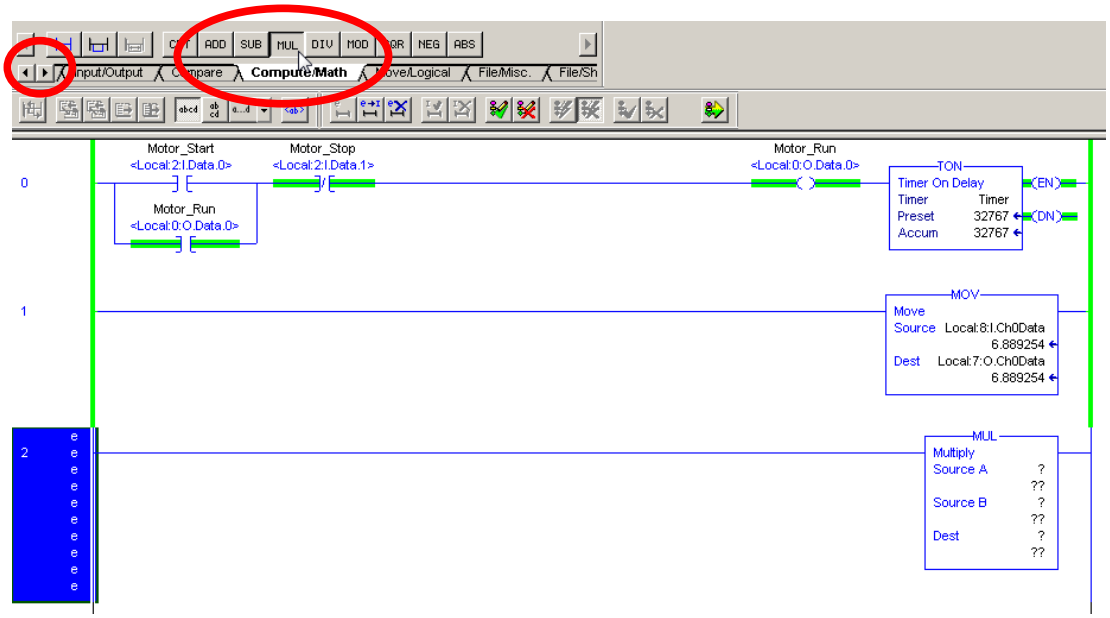
Add the UDT tag to an instruction

12. Double click on the MainRoutine.



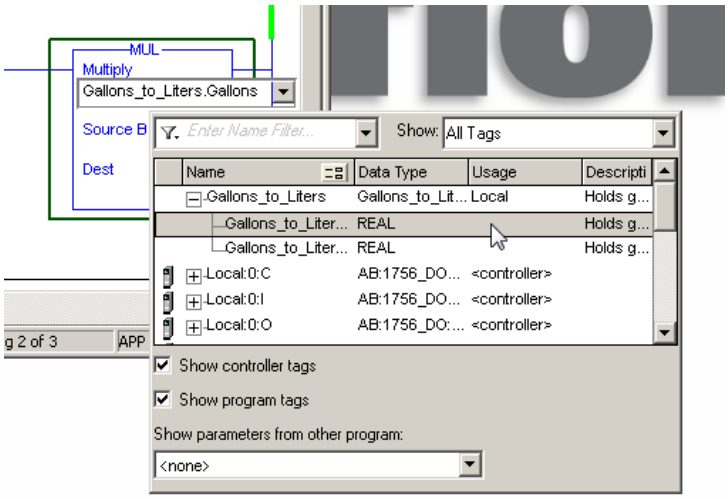
13. Make sure the End rung is highlited. Click on the  insert rung icon to create a new rung.

14. Find the ***Compute/Math*** tab on the instruction tool bar and click on the MUL instruction.



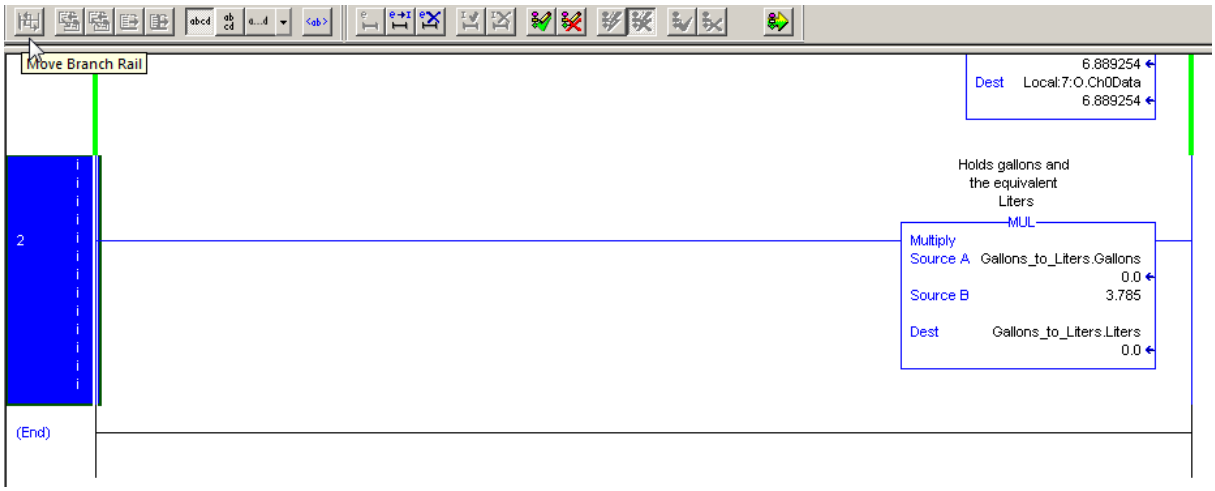
15. Double click on the "***?***" in the Source A filed of the MUL (multiply) instruction and select the Gallons_to_Liters.Gallons tag.

    Note: the Gallons_to_Liters tag will need to be expanded to select the Gallons member.



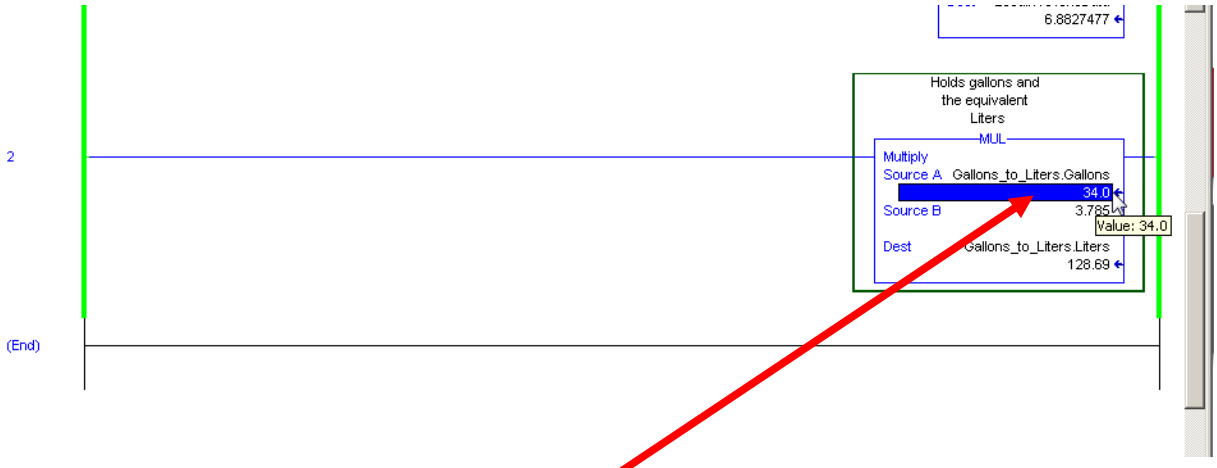
16. Fill in 3.785 for source B (the conversion constant to convert gallons to liters).

17. Double click on the ***?*** in the destination field and select the Gallons_to_Liters.Liters tag as shown.



18. Click on the ***finalize edits button***  and click on ***Yes*** to accept the changes.

    Notice that the the values of the tags are shown on the instruction. The multiply instruction converts the number in gallons to liters.



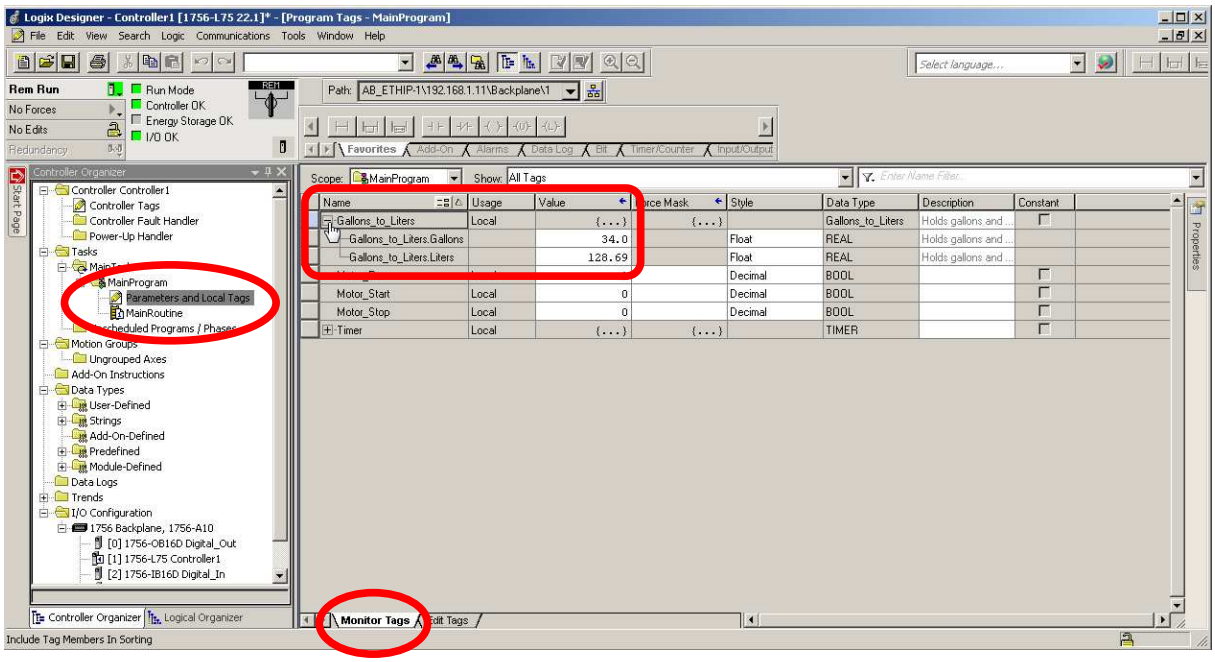
19. Click on the number ***0*** just below gallons, type "***34***", and press enter.

    Notice that the Liters value updates automatically.

Monitoring UDT Tags

20. Double click the ***Parameters and Local Tags*** under the MainProgram and expand the ***Gallons_to_Liters*** tag. Notice the values are also shown here. Make sure to select the ***Monitor Tags*** tab.

    The values for gallons can be modified directly in the monitor screen by changing the ***value*** in the Value column. Change the gallons value and watch that liters updates to corresponding value.



FYI

> The UDT allows associated data to be stored under a single main tag instead of using completely separate tags. This makes it easier to keep track of data and keep it more organized. The UDT name itself can document what the data is for.

Congratulations! You have Completed Section 8. Please move on to Section 9.

Section 9: (Optional) Using Studio 5000 Help

This lab section should take roughly 15 minutes to complete.

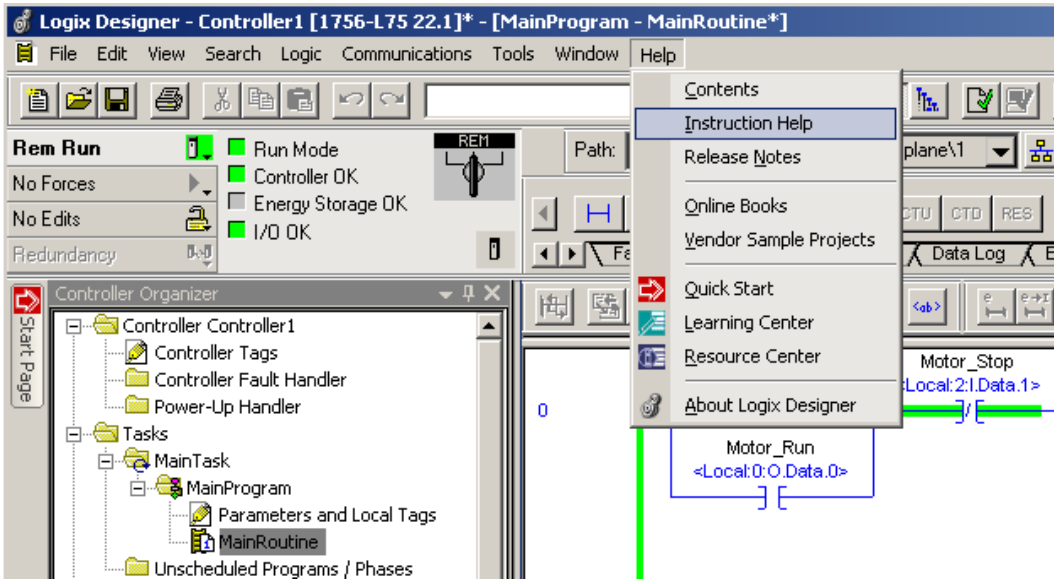
Objective:
In this lab we will explore the extensive online Help system in Studio 5000.

In this lab you will be viewing:

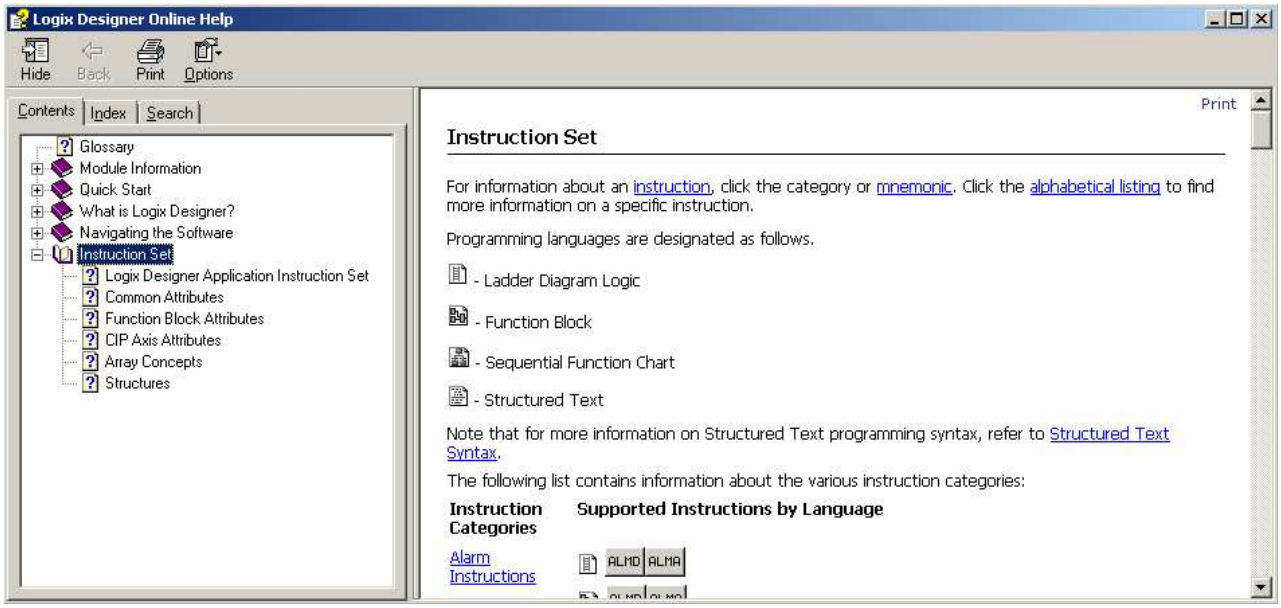
- Instruction help
- Module wiring diagrams
- On-line reference materials
- 3rd party vendor sample projects
- The Start Page – Quick Start

Instruction Help

21. From the ***Help*** pull down menu select ***Instruction Help.***



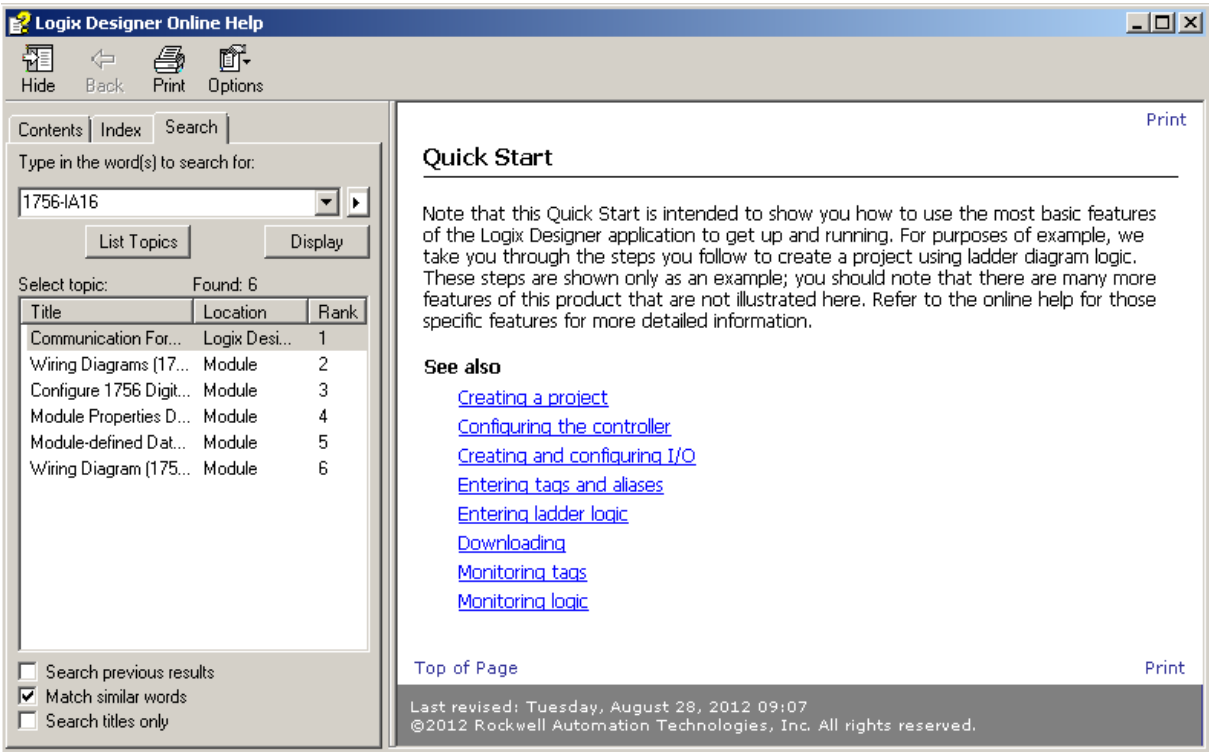
The following window will appear.



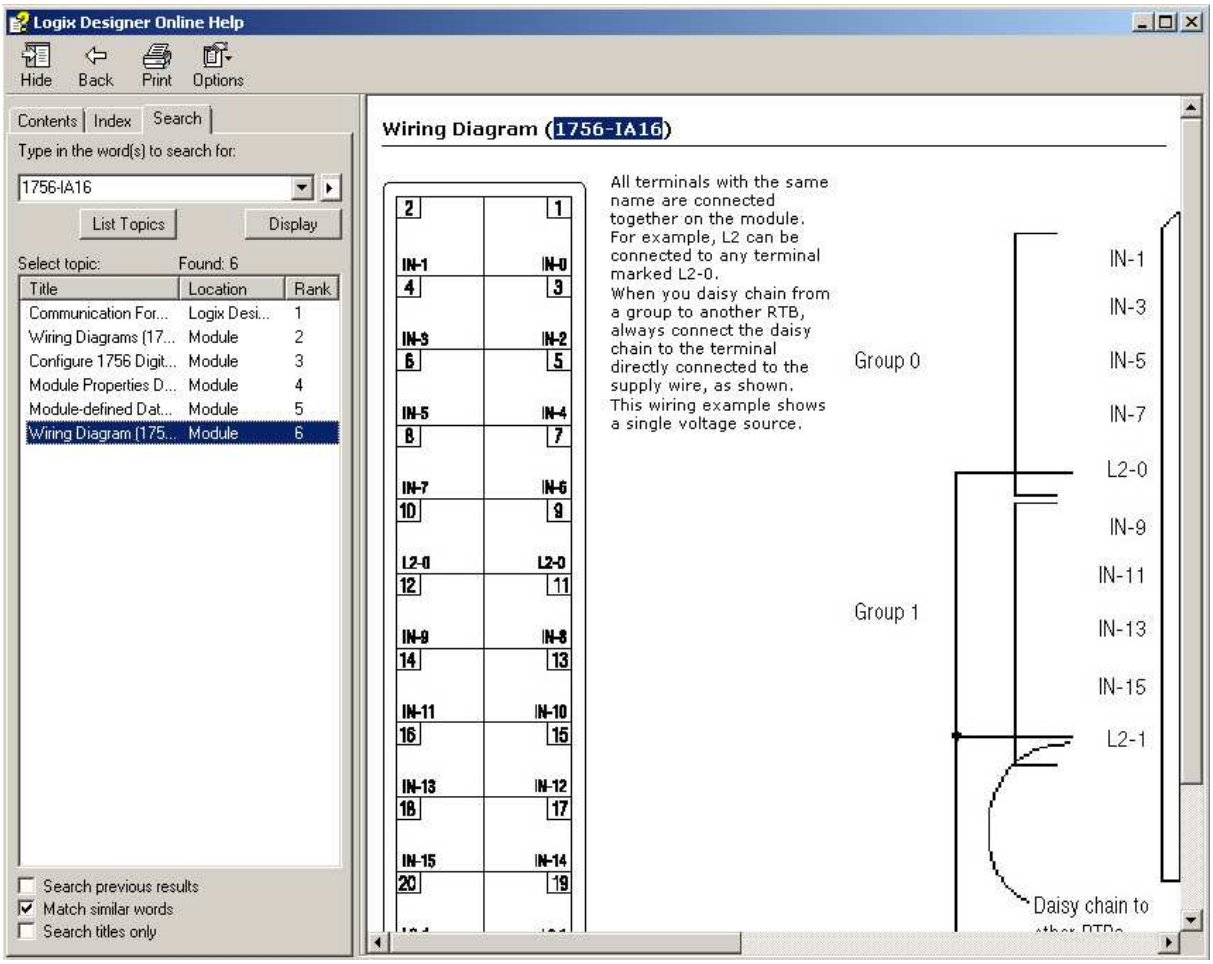
22. Click on an instruction to locate its description, details about its parameters, and related instructions along with examples on how to use the instruction.

Viewing I/O Module Wiring Diagrams

1. From the ***Help*** pull down menu select ***Contents.***
2. Select the ***Search*** tab if it is not already selected.
3. Type in ***1756-IA16*** as the keyword to find then click on **List Topics**.
4. Select a topic to display from the list such as, Wiring Diagram.



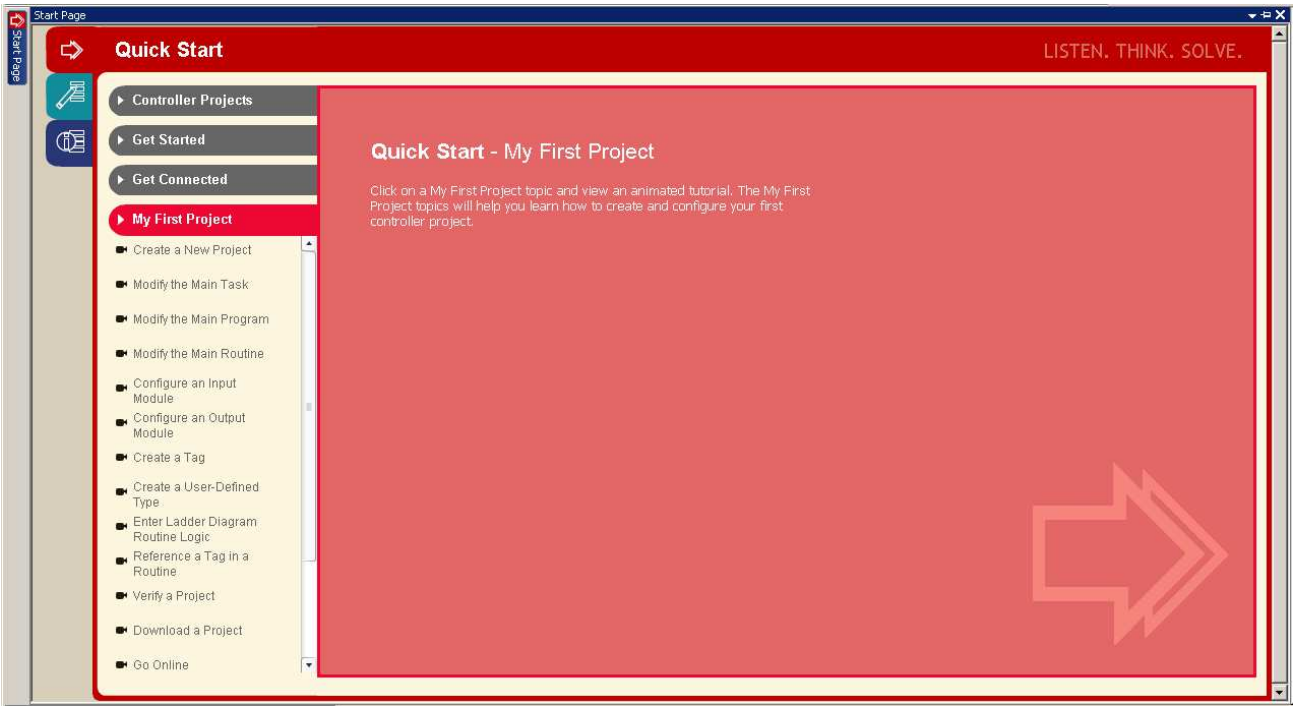
5. Click ***Display*** to view the wiring diagram for this module. Note you may need to maximize your screen.



6. When you are finished viewing the wiring diagram close the display window.

Using Start Pages

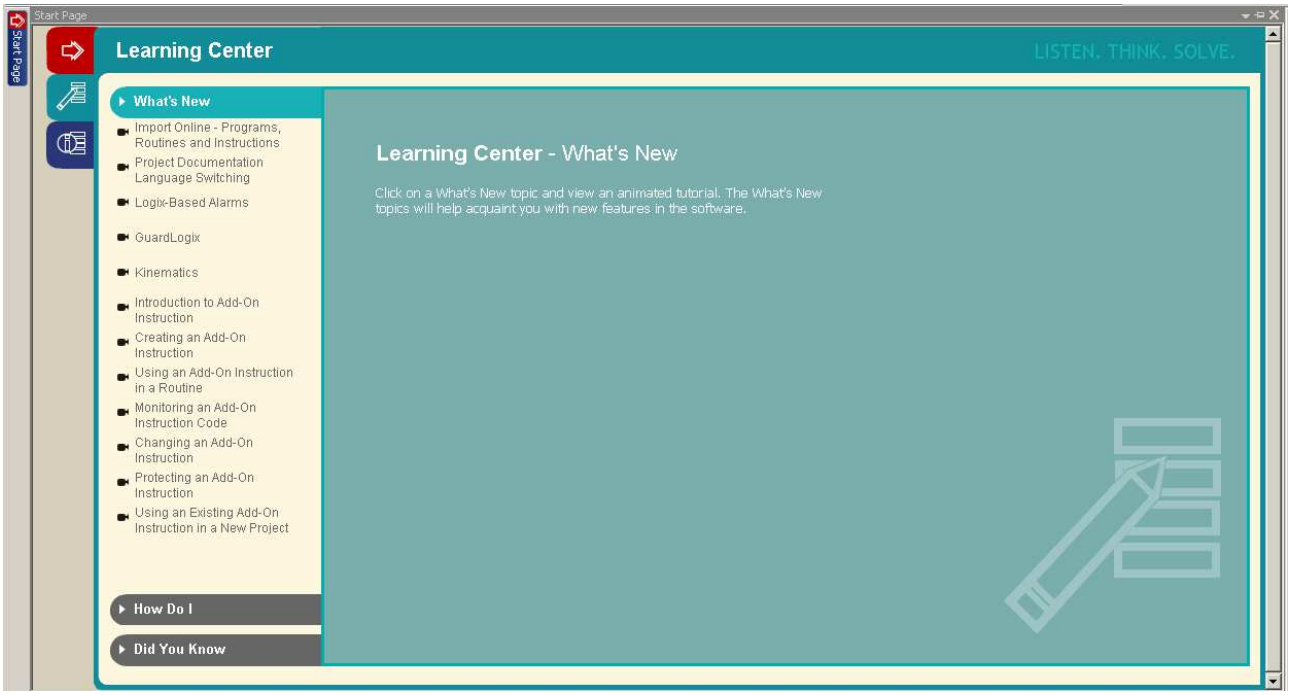
1. From the ***Help*** pull down menu select ***Quick Start***  which is one of the three tabs available from the ***Start Page***.



- Organizes various resources intended to accelerate the customer's ability to use the software and to locate relevant information
- Provides Getting Started and My First Project media clips and tutorials to assist new users
- Provides easy navigation to Studio 5000 sample projects Rockwell Automation specific and those involving other vendors

Learning Center Tab

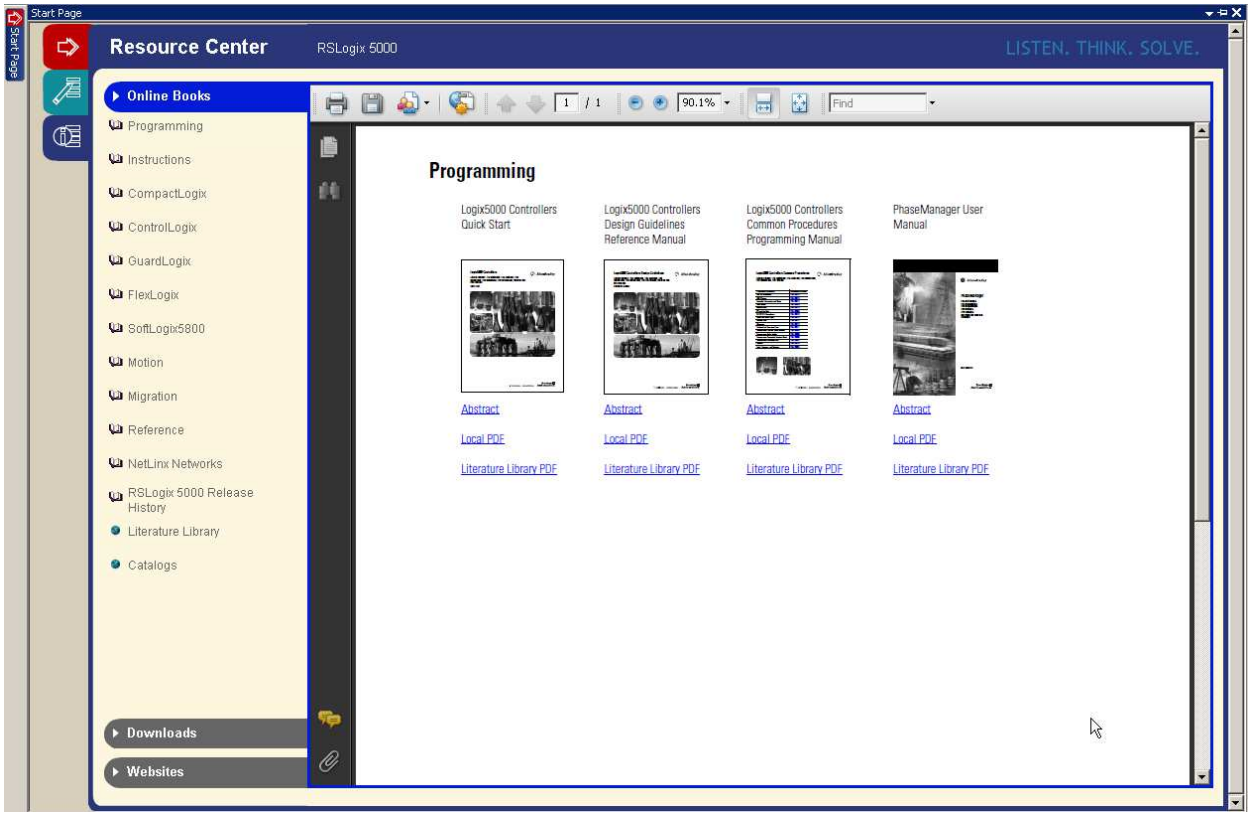
- Targets customers wanting to learn or explore how to use the software beyond just getting started reduces learning curve and helps increase productivity.



- ***What's New*** media clips or tutorials previewing new features
- ***How Do I*** media clips or tutorials organized under various topics to show the user how to use the software to complete common tasks
- ***Did You Know*** tips / tricks for using the software, e.g. Keyboard Shortcuts

Resource Center Tab

- Targets a customer looking for additional information or support
- Provides links to Download sites for software, firmware, EDS files, etc
- Provides links to Support sites Knowledgebase, Technical Bulletins, Sample Code
- Provides links to Online books installed to the PC with Studio 5000.



Congratulations! You have completed all sections!

Thank you for attending!

Notes :