L25 - Studio 5000 Logix Designer®: Basics Lab

For Classroom Use Only!

LISTEN. THINK. SOLVE.®

Allen-Bradley • Rockwell Software

Rockwell Automation

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.

Contents

Before you begin

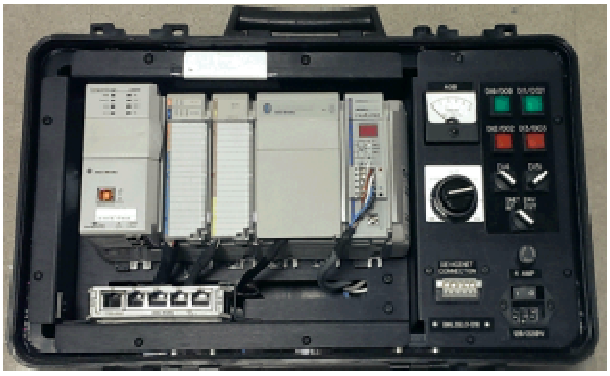
About this lab

This session provides you with an opportunity to explore the Studio 5000 Logix Designer software and the CompactLogix hardware platform. The session steps through creating a new project, programming, and downloading to a controller.

This lab takes approximately 105 minutes to complete.

Tools & prerequisites

- Studio 5000 Logix Designer
- RSLinx Classic
- No files are required for this lab
- CompactLogix L3Y-Lite demo box (2 models shown below)



> ***Tip Text –*** *The text inside this gray box is supplemental information. This text can include FYIs, useful tips, and other related information*

About Studio 5000 Logix Designer

Studio 5000 software includes the Logix Designer application for the programming and configuration of Allen-Bradley ControlLogix and CompactLogix programmable automation controllers. Logix Designer is the progression of RSLogix 5000 software, and will continue to be the package you use to program Logix5000 controllers for discrete, process, batch, motion, safety, and drive-based systems. Logix Designer offers an easy-to-use, IEC61131-3 compliant interface, symbolic programming with structures and arrays and a comprehensive instruction set that serves many types of applications. It provides ladder logic, structured text, function block diagram and sequential function chart editors for program development as well as support for the S88 equipment phase state model for batch and machine control applications.

About CompactLogix Controllers

CompactLogix: Perfect for smaller, machine-level control applications

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.

CompactLogix offers the following benefits:

- Ideal for small, to mid-size applications that require low axis motion and I/O point counts
- Offers support for Integrated Motion over EtherNet/IP™ for maximized scalability
- Provides support for Device Level Ring (DLR) network topologies to help increase network resiliency
- Removes the need for lithium batteries with built-in energy storage
- Includes up to a 2-GB secure digital (SD) card for fast program save and restore
- Offers a smaller form factor for maximized cabinet space
- Supports up to 2 axes Kinematics for simple articulated robotics
- Open socket capability allows support for Modbus TCP as well as devices such as printers, barcode readers and servers

GuardLogix Safety Controllers Features:

- Provides integrated safety and integrated motion in a single controller
- Supports integrated safety up to SIL 3, PLe CAT 4

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.

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, and 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
- Wide range of communication options and analog, digital, and specialty I/O
- Select ControlLogix products are TUV-certified for use in SIL 2 applications

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.

GuardLogix Safety System

A GuardLogix controller is a ControlLogix controller that also provides safety control. The GuardLogix controller is used with a safety partner to achieve SIL 3/PLe/Cat. 4. A major benefit of this system is that it is still one project, safety, and standard together. The safety partner controller is a part of the system, is automatically configured, and requires no user setup.

Section 1: Creating a Project

> ***Tip Text –*** *The text inside this gray box is supplemental information. This text can include FYIs, useful tips, and other related information*

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. Double-click on the ***Studio 5000 icon*** on the Desktop to launch Studio 5000 software

The Studio 5000 Splash Screen appears

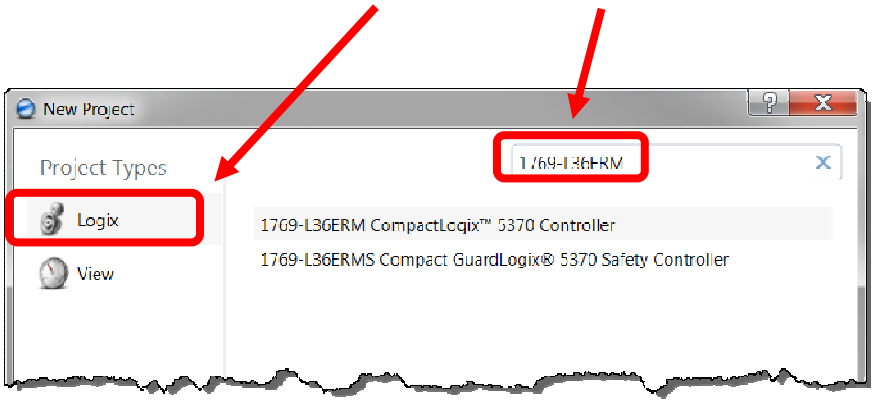
> ***Tip -*** To see what versions of Studio 5000 you have installed on your computer, select ***About*** under the **Explore** section.

Creating a New Controller Project

In this portion of the lab, you will create an offline project using a CompactLogix controller.

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

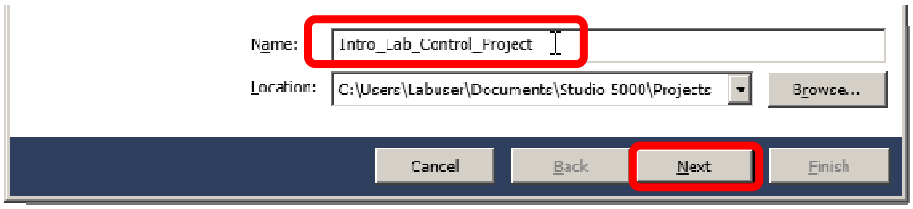
3. When the **New Project** pop-up is displayed, select ***Logix*** and type ***'1769-L36ERM'*** in the Search field.



> Be sure to choose the correct controller type and revision that matches the equipment at your lab station. Not choosing correctly will prevent the application from downloading later in the lab

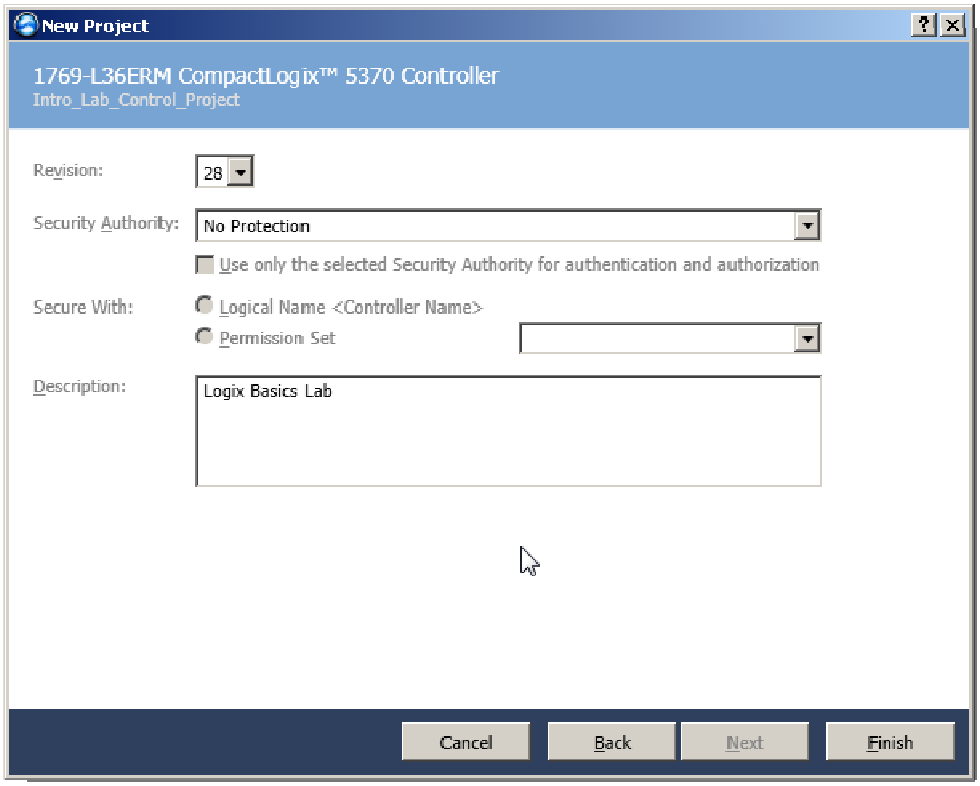
4. Type ***'Intro_Lab_Control_Project'*** into the name field.

5. Press the ***Next*** button.



> FYI – A name is required. This names both the controller and the default ACD file name. An ACD file is the file the project is stored in. In this case, the file name is "Intro_Lab_Control_Project.ACD"

6. When the **Project Configuration** window appears, fill it in as follows:



- Select ***V28***
- Select ***No Protection***
- Add a project description ***'Logix Basics Lab'***
- Click ***Finish***

The Logix family of controllers in this lab all use Studio 5000 Logix Designer software to configure the system, but each controller type is set up slightly differently. For example, ControlLogix has more settings than CompactLogix.

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

Controller 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.

Name: The name of the controller and project.

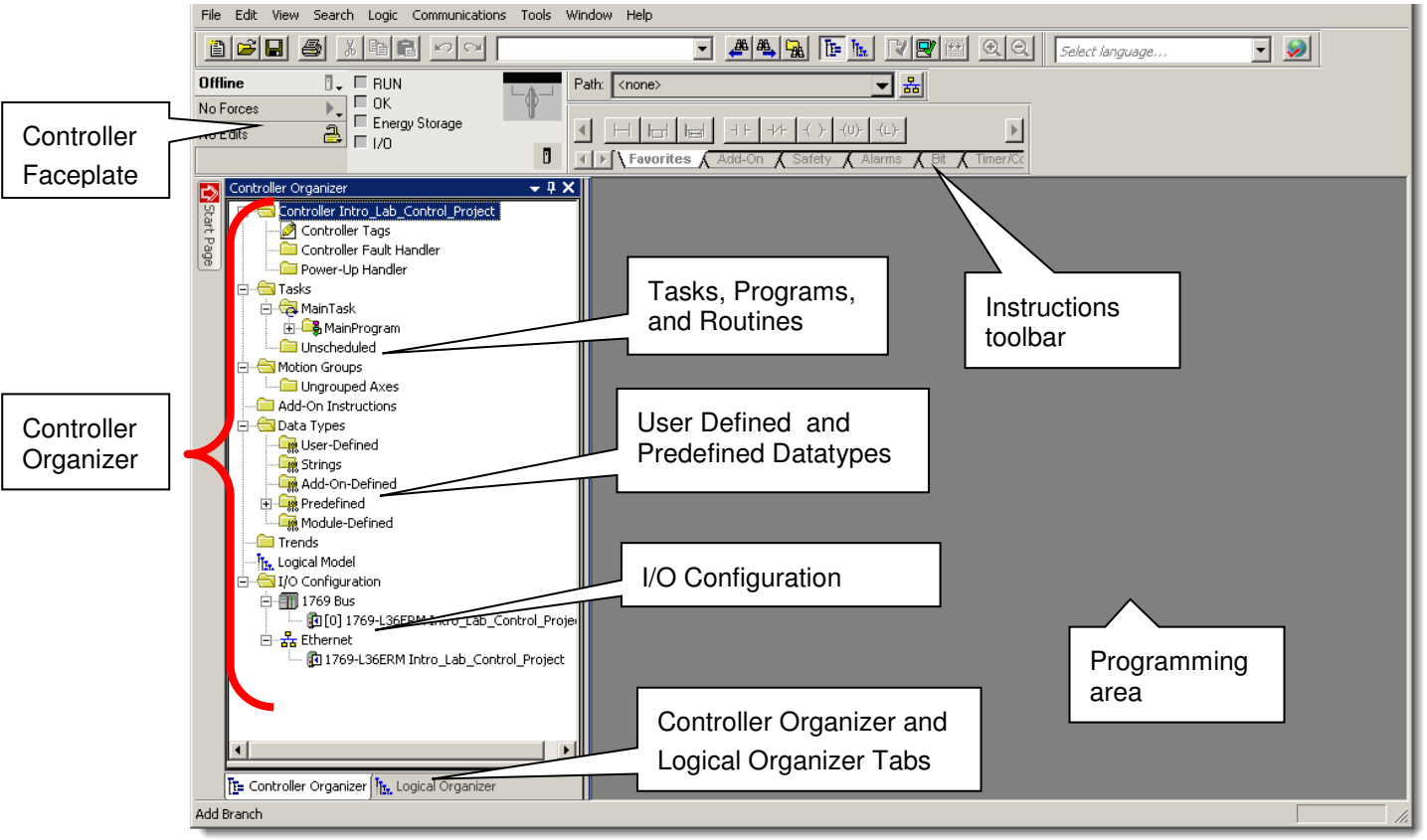
Revision: Here you are selecting the firmware revision of the project that will be created.

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.

Security Authority: Optional, the security server to use for project security.

The **Organizer Window** appears on the left side of the Studio 5000 window, with a folder called Controller Intro_Lab_Control_Project. At this time, there is no I/O, tag database, or logic associated with the controller. We will be adding these later. The following picture points out the various areas.



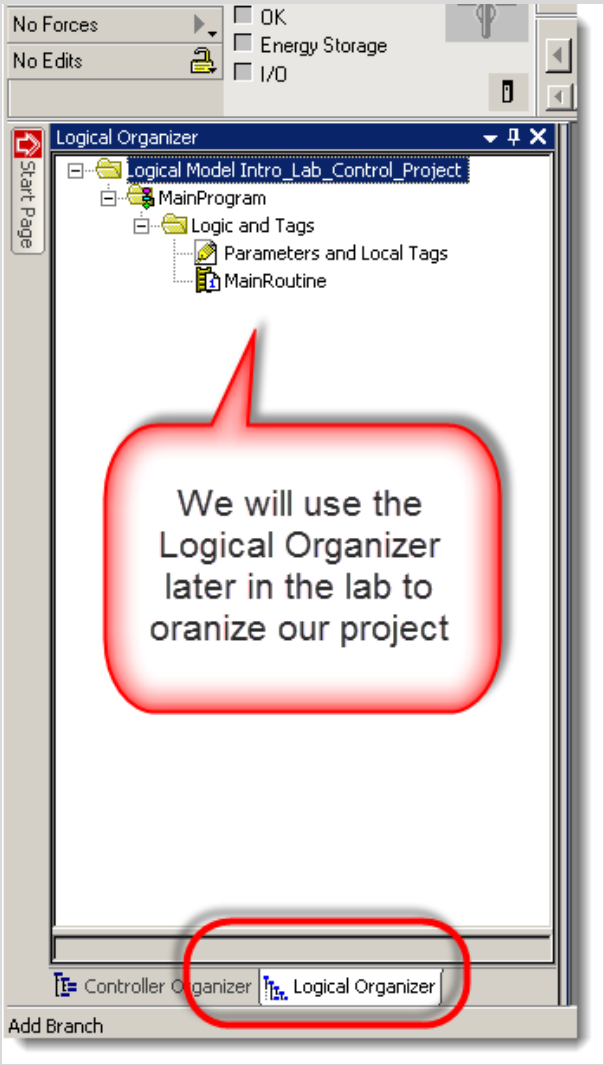
You have now created your first controller project!

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 or collapse items in the tree display.

The **Logical Organizer** – This a recent feature of Logix Designer. The Logical Organizer allows code to be grouped and organized by purpose, instead of by task execution. The code can be organize in a way to make understanding and troubleshooting the end machine or application easier and faster! This also gives greater documentation directly inside the Logix Designer application! We'll cover this more later in the lab.

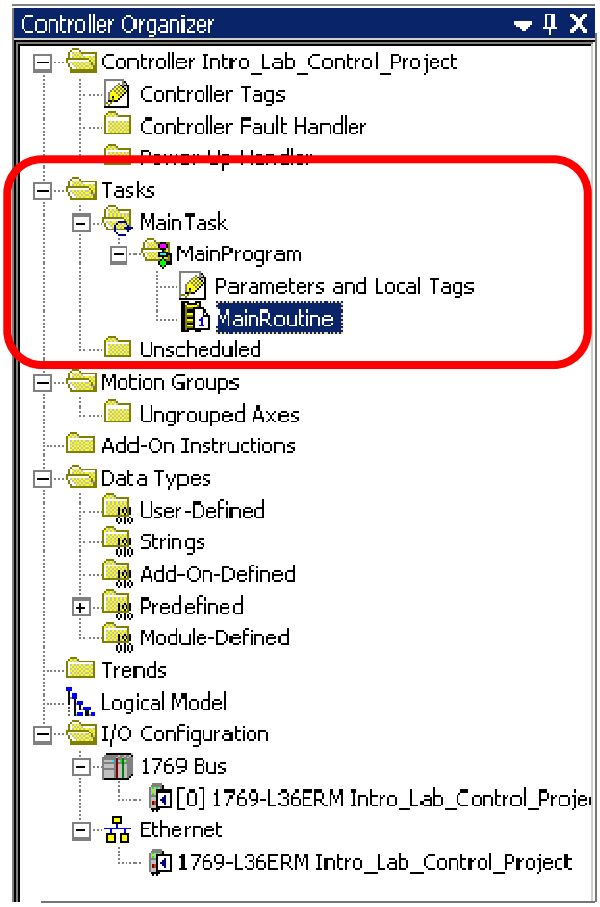


Adding Ladder Logic to the Main Routine

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.

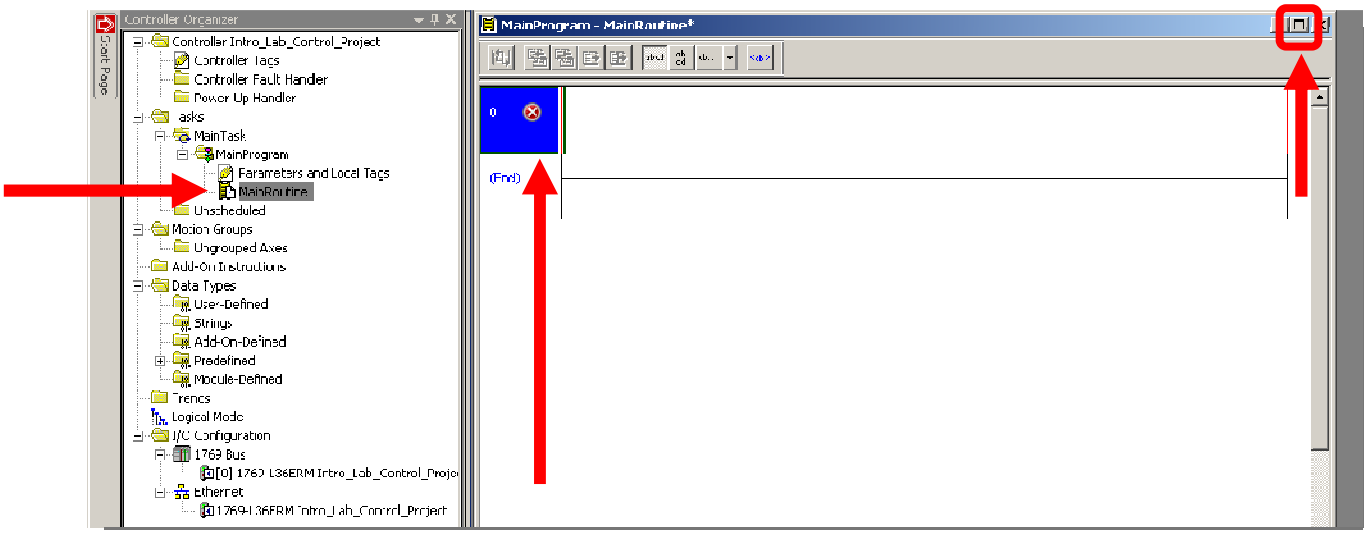
You will continue to use the project already open.

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

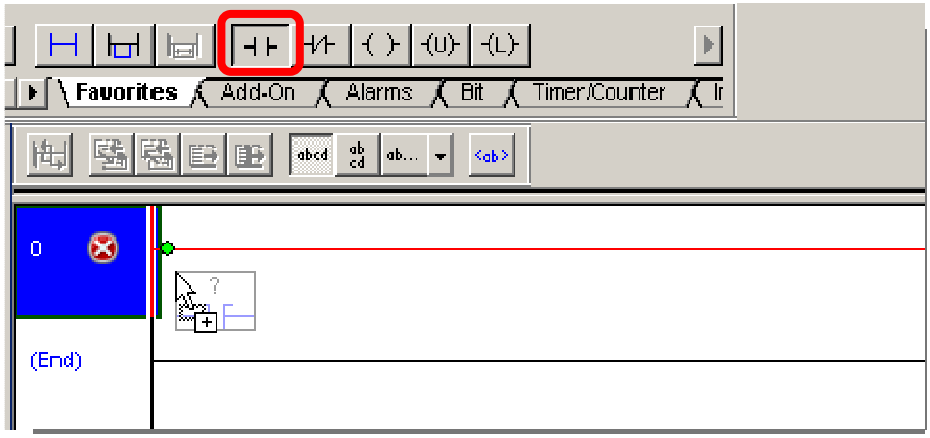


2. Double-click the ***MainRoutine*** icon and ***maximize the ladder window*** if it is not maximized.

This will open the routine editor. An empty rung will already exist as shown below: The red color of the rung and the circled "x" next to the rung indicate the rung is incomplete.

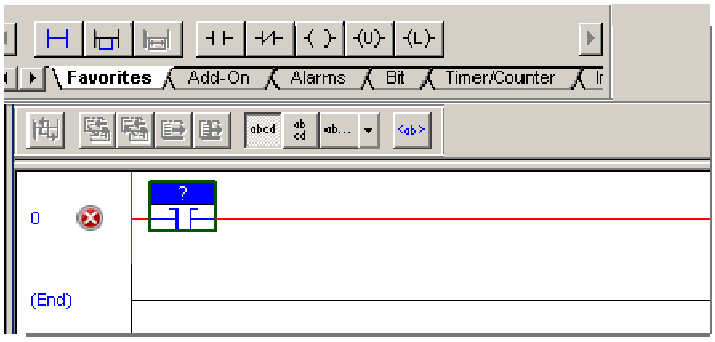


3. From the instruction toolbar, left click and hold on the ***Examine On*** (***XIC***) instruction. 

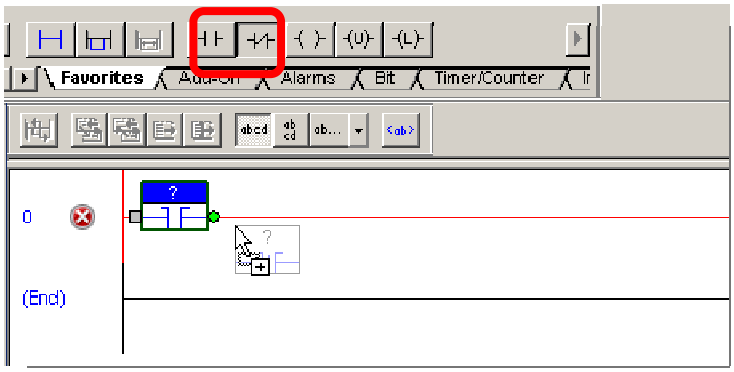


4. Drag the ***XIC*** onto **rung 0** until the **green dot** appears as shown above. Release the mouse button at the location you wish to place your instruction.

Verify your rung appears like the figure below:

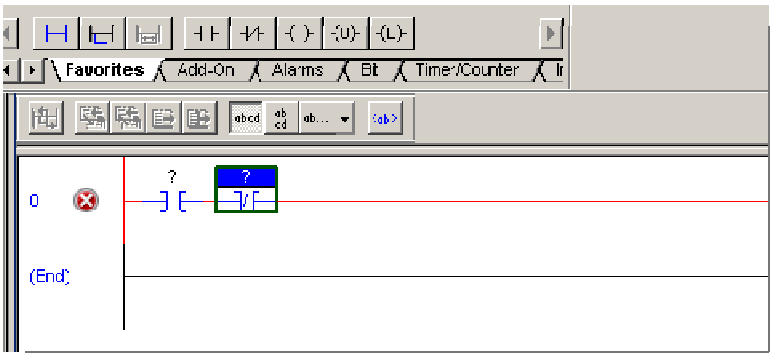


5. From the instruction toolbar left click and hold on the ***Examine Off*** (***XIO***) instruction. 



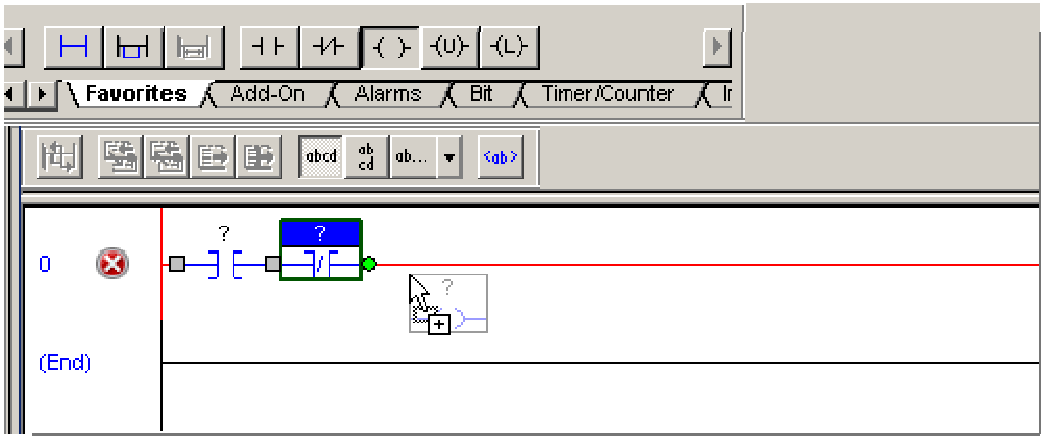
6. Drag the ***XIO*** onto **rung 0** to the right of the **XIC** instruction as shown above. Again a green dot will appear to the right of the XIC instruction indicating where your new instruction will be inserted. Release the mouse button at the location you wish to place your instruction.

Verify your rung appears like the figure below:



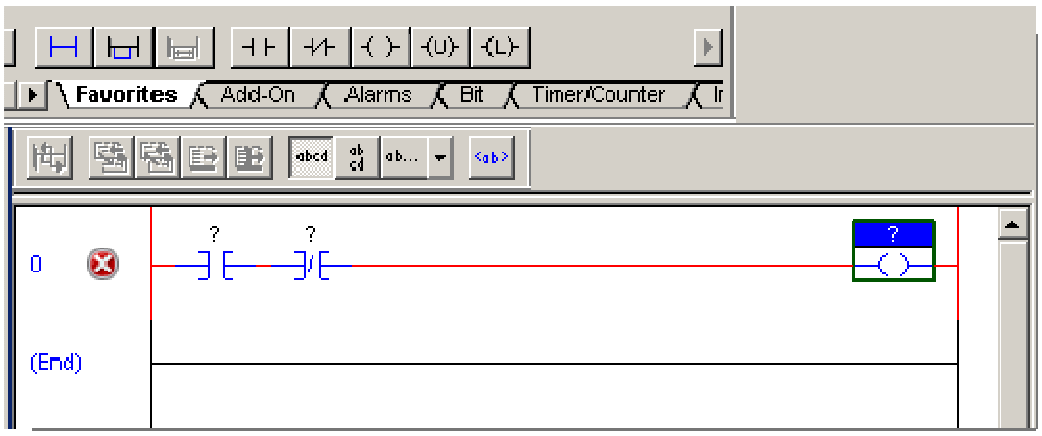
FYI - If you place an instruction in the wrong location on a rung, simply click and hold on the instruction and drag it to the correct location.

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



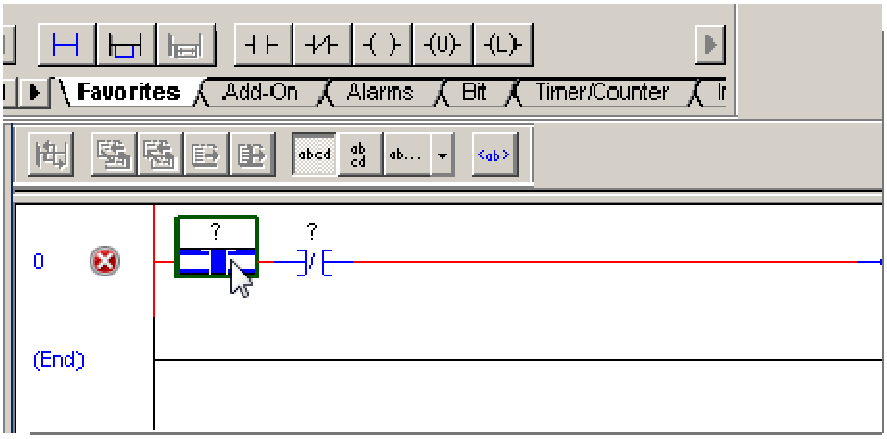
8. Drag the ***OTE*** onto **rung 0** to the right of the **XIO** instruction as shown above. Again a green dot will appear to the right of the XIO instruction indicating where the OTE instruction will be inserted. Release the mouse button at the location you wish to insert the instruction.

Verify the rung appears as follows:



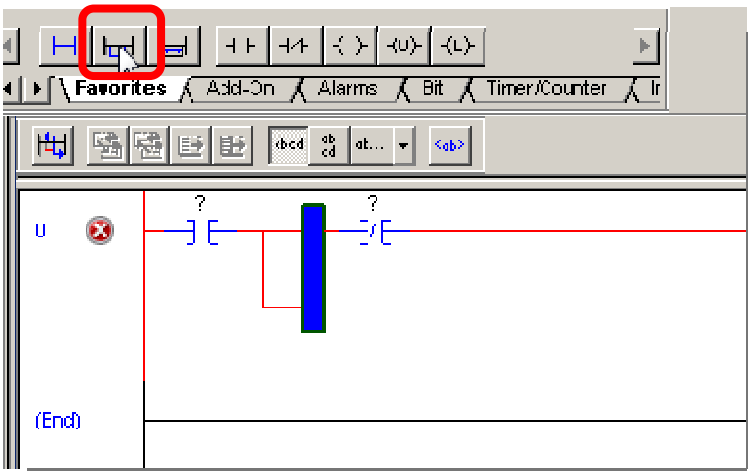
We will now add a branch around the XIC instruction.

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



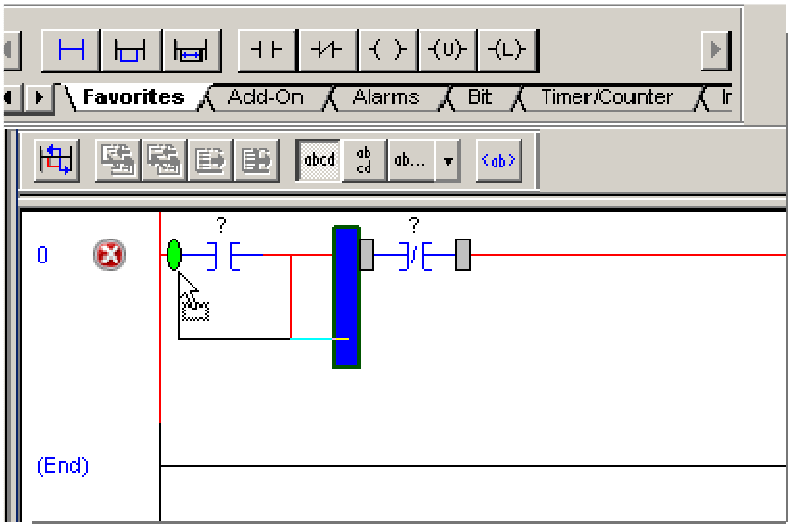
10. From the instruction toolbar click on the ***Branch*** instruction. 

A branch will be inserted on the rung.



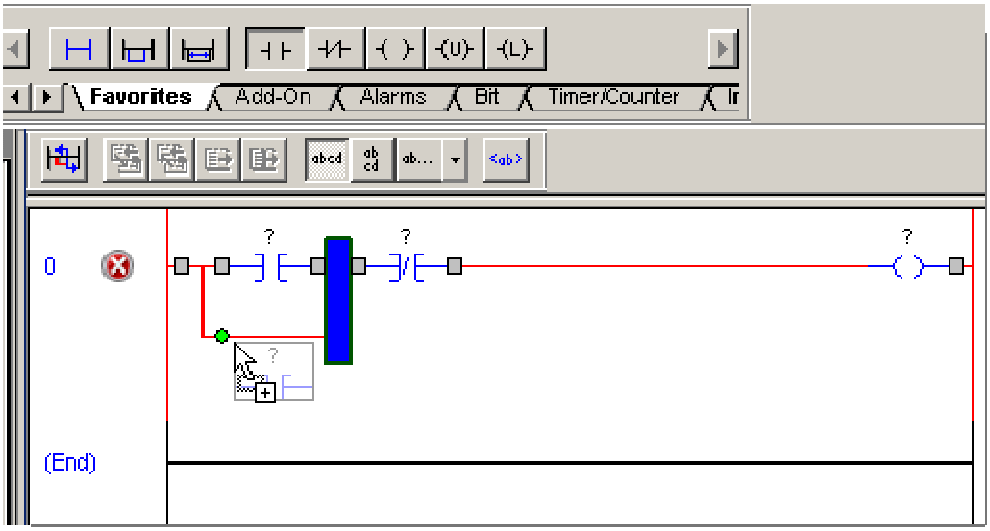
11. Left-click and hold on the ***blue highlighted part of the branch*** and drag your selected leg of the branch to the ***left side of the XIC instruction***.

12. Place the branch ***over the green dot and release*** the mouse button.

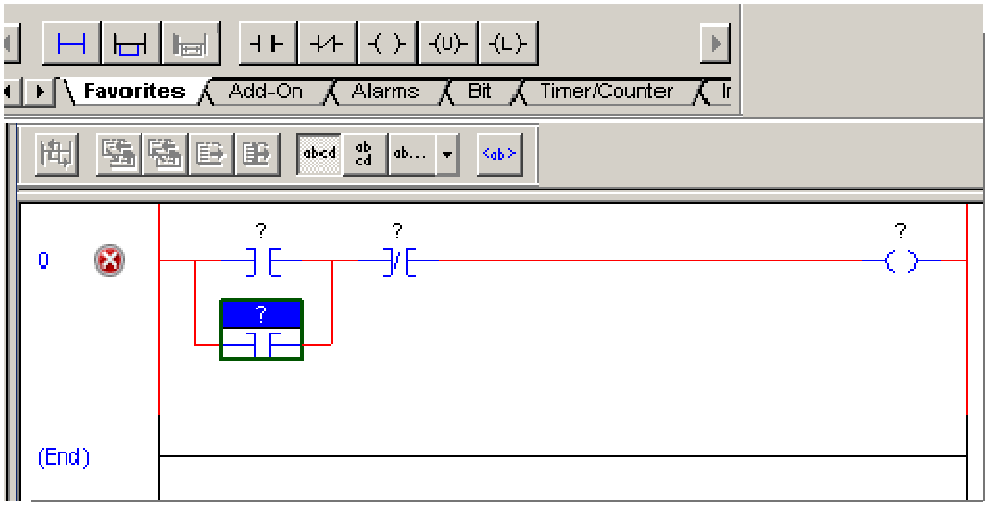


13. From the instruction toolbar, left click and hold on the ***XIC***  instruction.

14. Drag the ***XIC*** onto your newly created branch until the ***green dot appears and release*** the mouse button.



Verify that the entire rung appears like the figure below. If it does not, use what you have learned to make it match.



15. 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\

> 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.

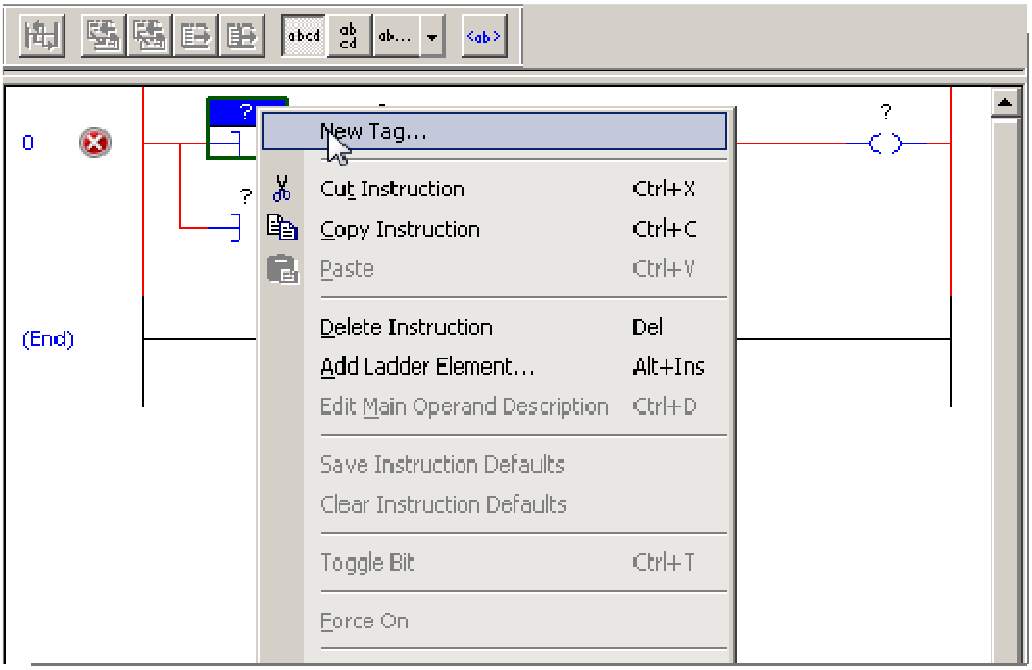
Creating Tags for the Ladder Code

In this section of the lab you will create the tags needed for the program. In older 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.

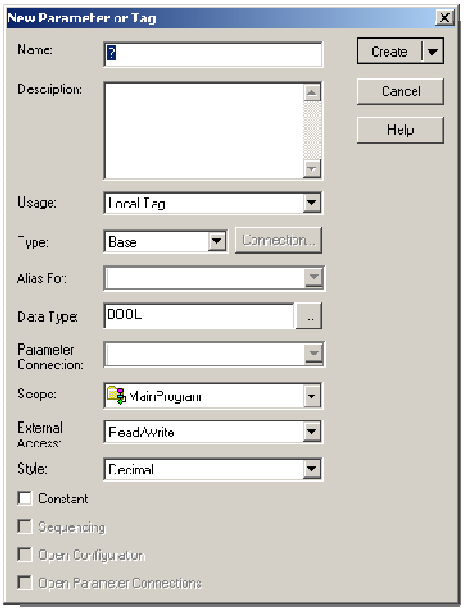
> **What is a tag and why are they better?**
>
> A tag is a text-based name for an area of memory. By using a text-based system you can use the name of the tag to document your ladder code and organize your data to mirror your machinery. For example you could create a tag named North_Tank_Pressure. This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



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.

Parameter Connection: Shows and allows selection of the parameter connected to this tag.

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

Style: Display the tag value has Binary, Octal, Decimal, or Hex.

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

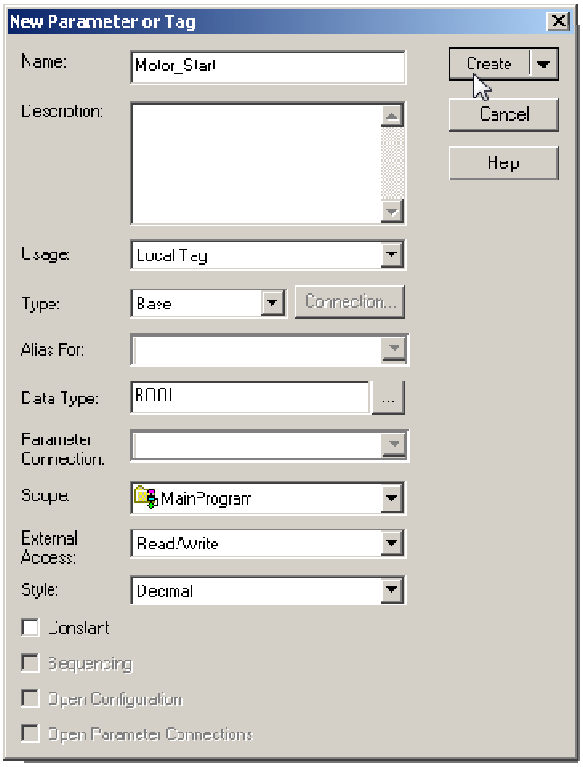
Sequence: Allows Equipment Phase input/output tags to be used with FactoryTalk Batch Server.

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

Open Parameter Connection: Opens the Connection Configuration window.

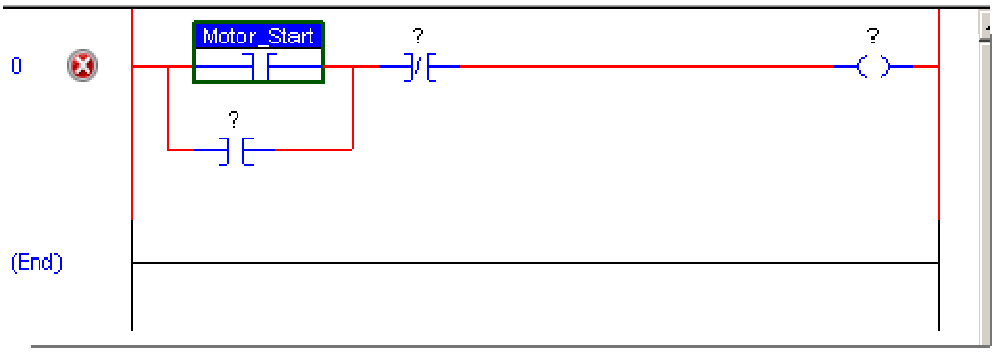
2. Enter the name ***"Motor_Start"*** and fill in the remaining fields as shown below.

Make sure the scope of the tag is MainProgram.



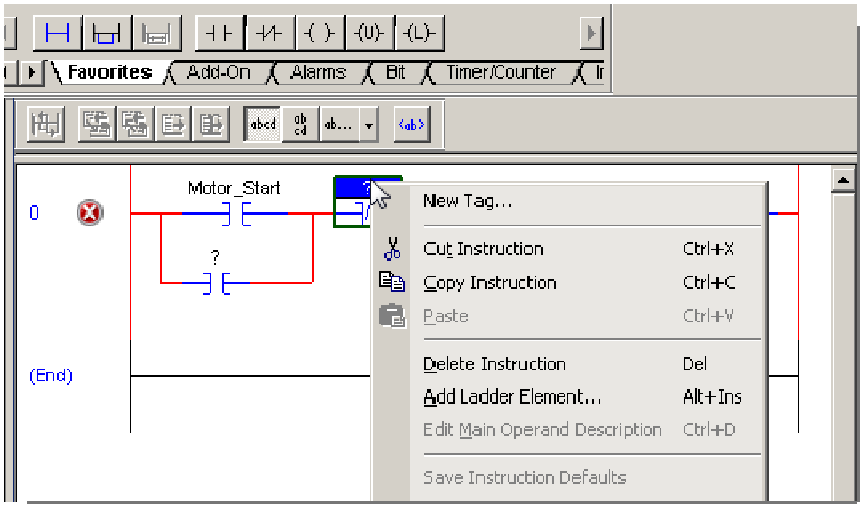
3. Click ***Create*** to accept and create the tag.

The rung will now look like the figure below.



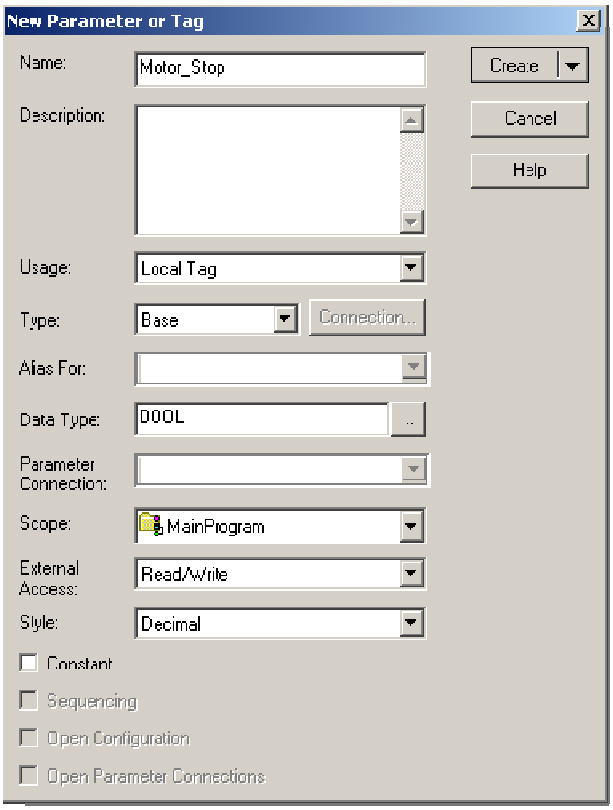
Next you will create the tag Motor_Stop.

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



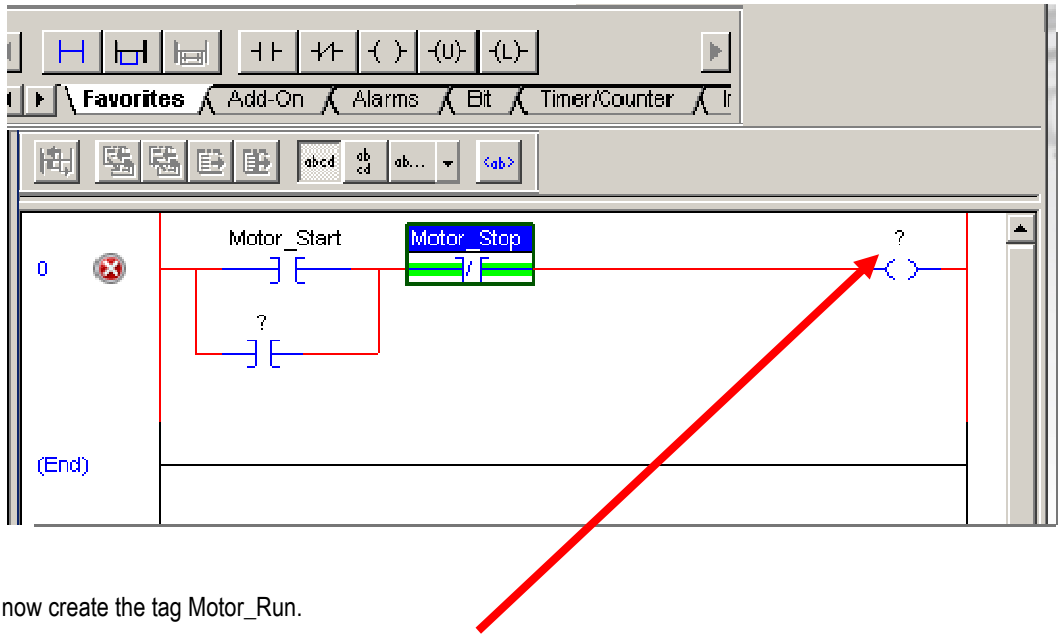
Again, the New Tag window will appear:

5. Enter the name ***"Motor_Stop" and other fields*** as shown below:



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

Verify the rung appears like the figure below:

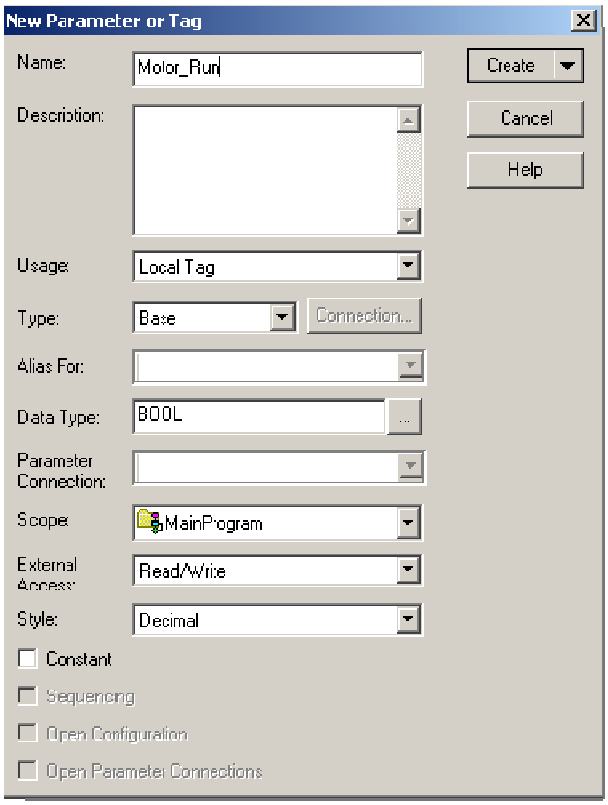


You will now create the tag Motor_Run.

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

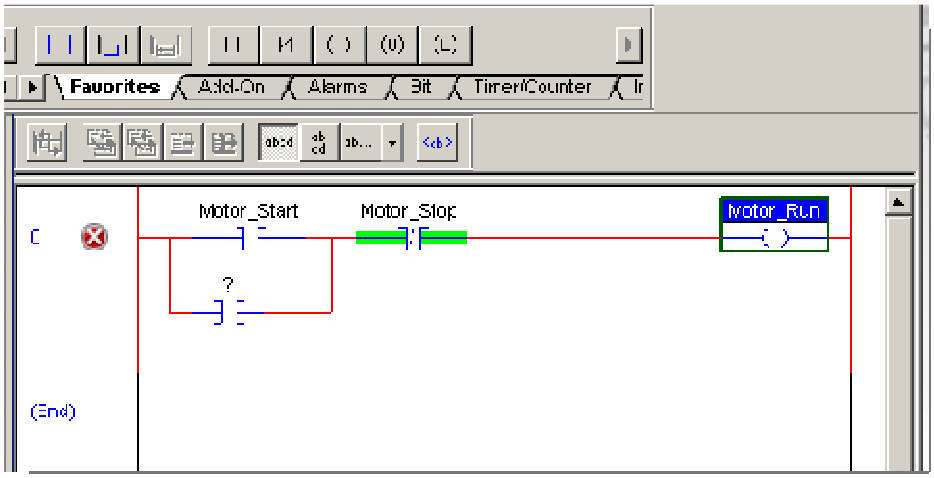
The New Tag window will appear.

8. Enter the fields as shown below for ***"Motor_Run"***:



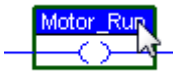
9. Click ***Create*** to accept and create the tag.

Your rung should now appear as shown below:

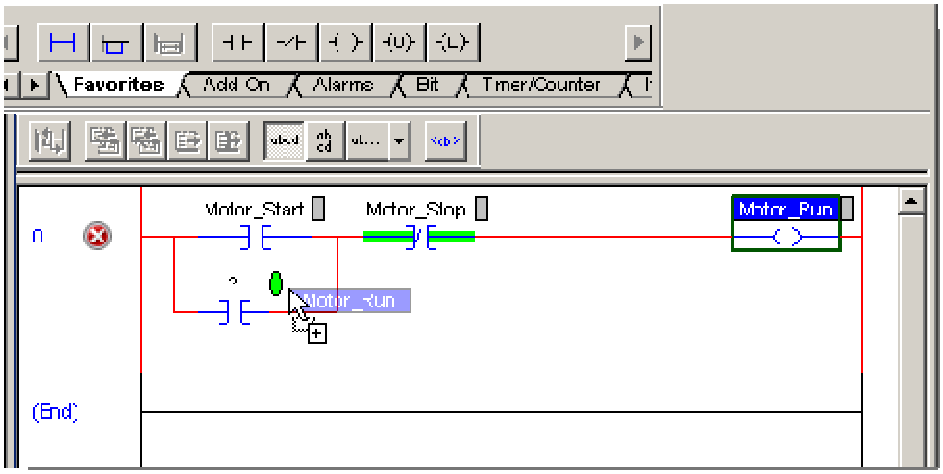


For the XIC instruction in the branch we do not have to create a tag. You will use the tag Motor_Run.

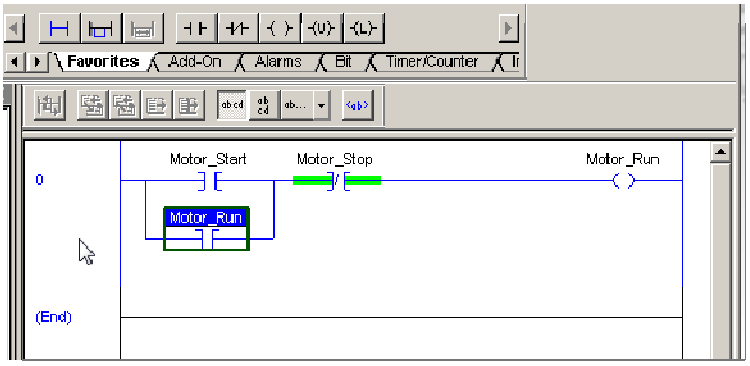
10. Left click and hold the mouse button over the tag ***Motor_Run*** on the OTE instruction.



11. Drag the ***Motor_Run*** tag to the ***XIC*** instruction until a green dot appears next to the "***?***" and release the mouse button.

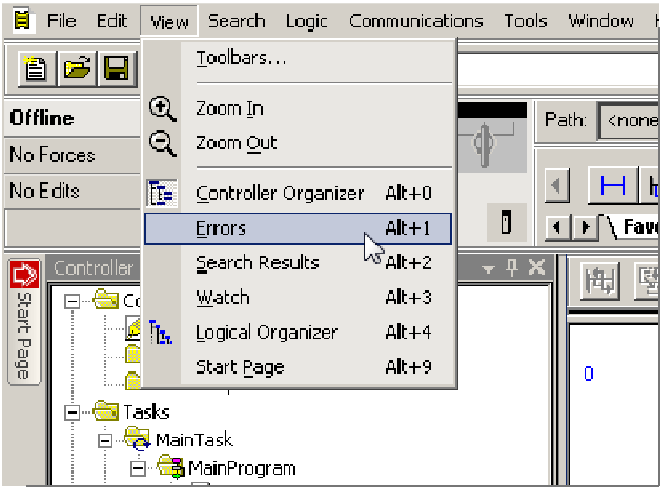


The rung should now appear as shown below. Notice the “X” next to rung zero has disappeared and that the rung color is now blue. This indicates that the rung passes auto verification and no errors are present.

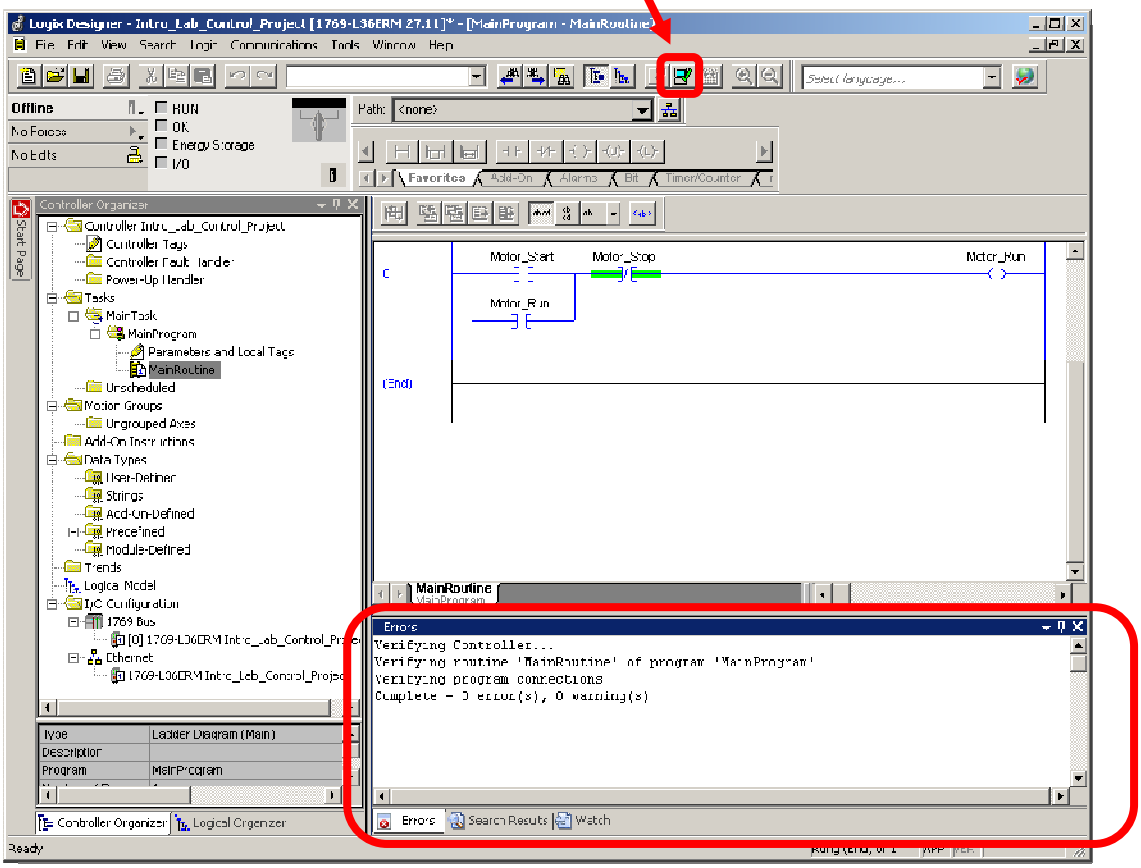


Studio 5000 software verifies each rung automatically to make programming easier!

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



13. Verify the program by clicking on the ***Verify Controller icon***  on the toolbar.

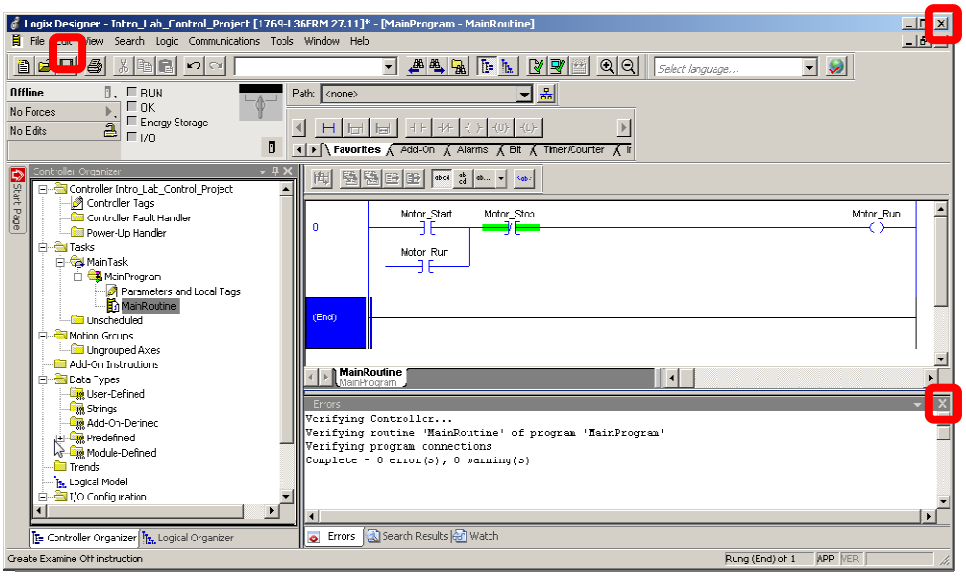


You will see if there are any errors in the status window.

> This is useful to locate errors or incomplete rungs in larger projects that may have hundreds or thousands of rungs!

14. Close the **MainRoutine** by clicking the ***"X"*** located at the top right corner of the screen.

15. Close the **Errors window** by clicking its ***"X"***



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

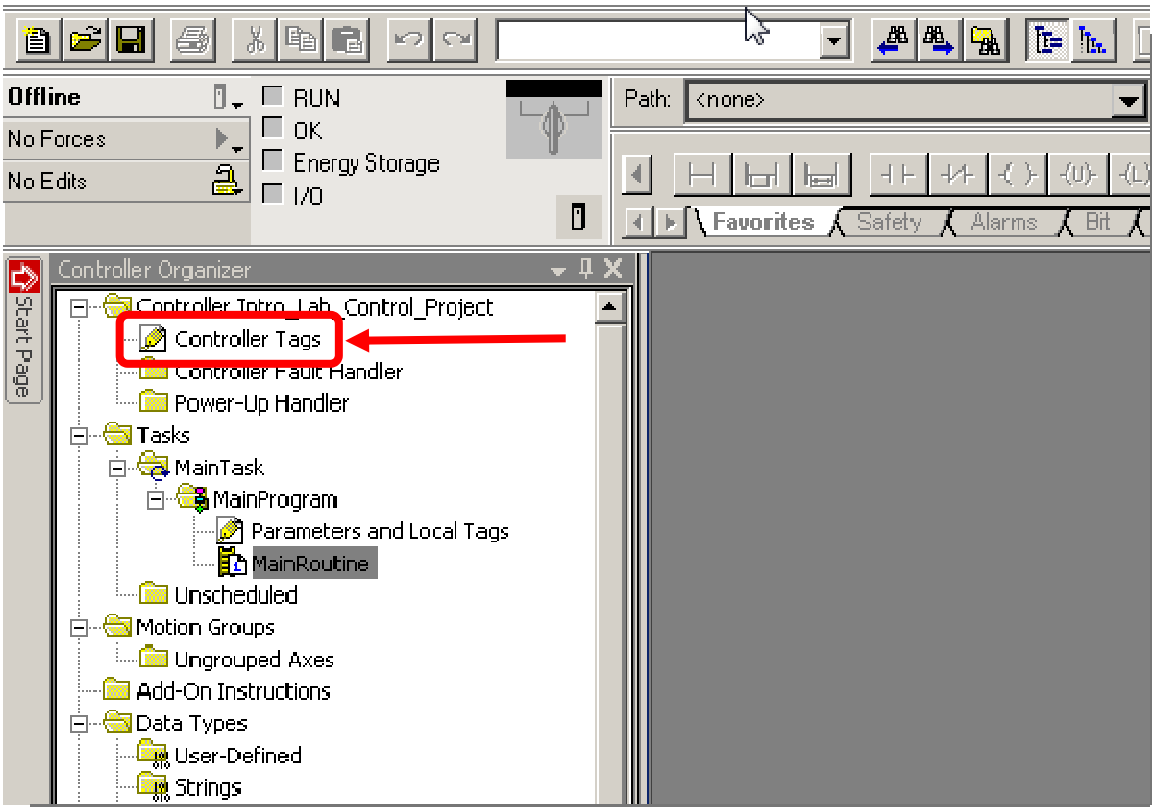
> The tag database of Logix versus a traditional PLC's fixed memory addresses help you create self-documenting code. This means you do not have to use address descriptions or symbols to make code easy to read.

Monitoring/Editing Tags

In this section of the lab, we will review the Tag Monitor/Editor in Studio 5000. The concept of Controller, Parameter, and Program Local tags will be covered.

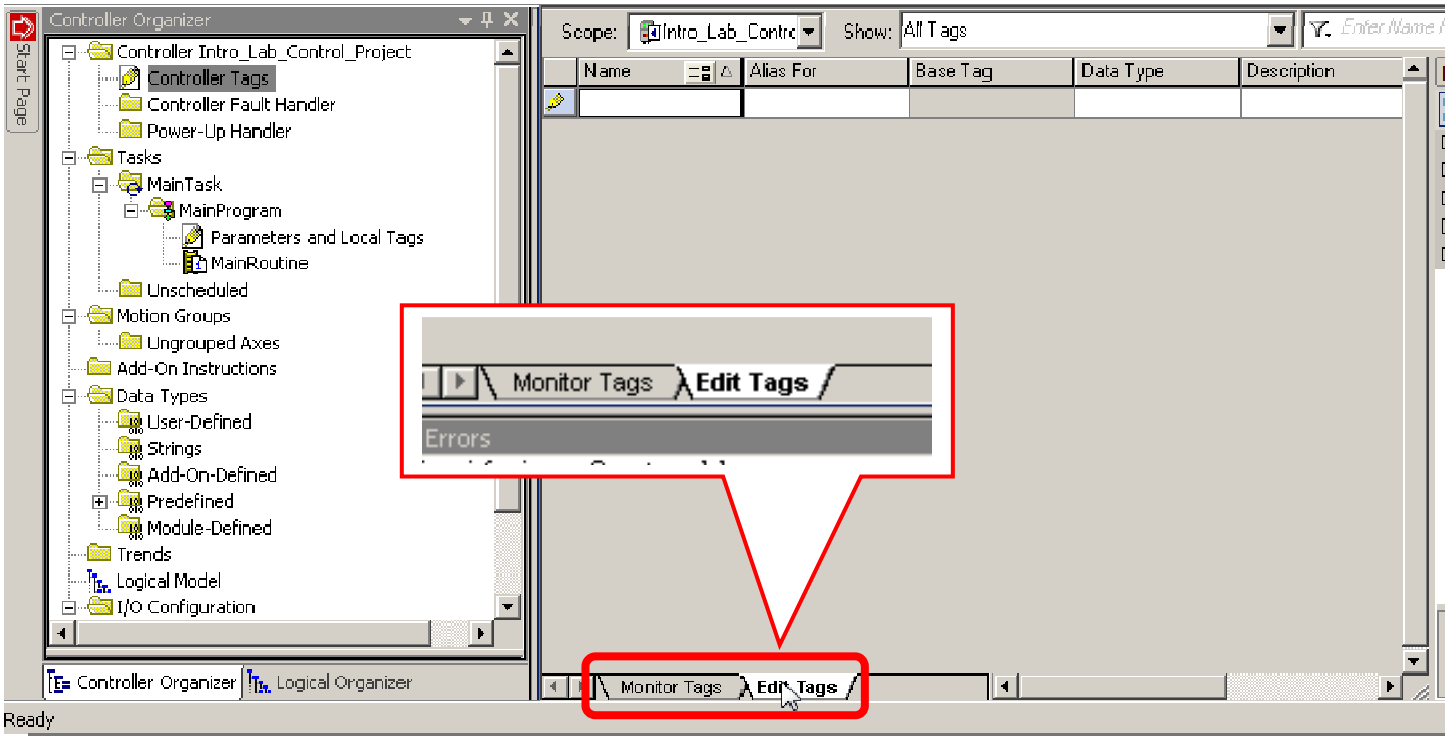
You will continue to use the project already open.

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



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

2. Click on the tab ***Edit Tags***.



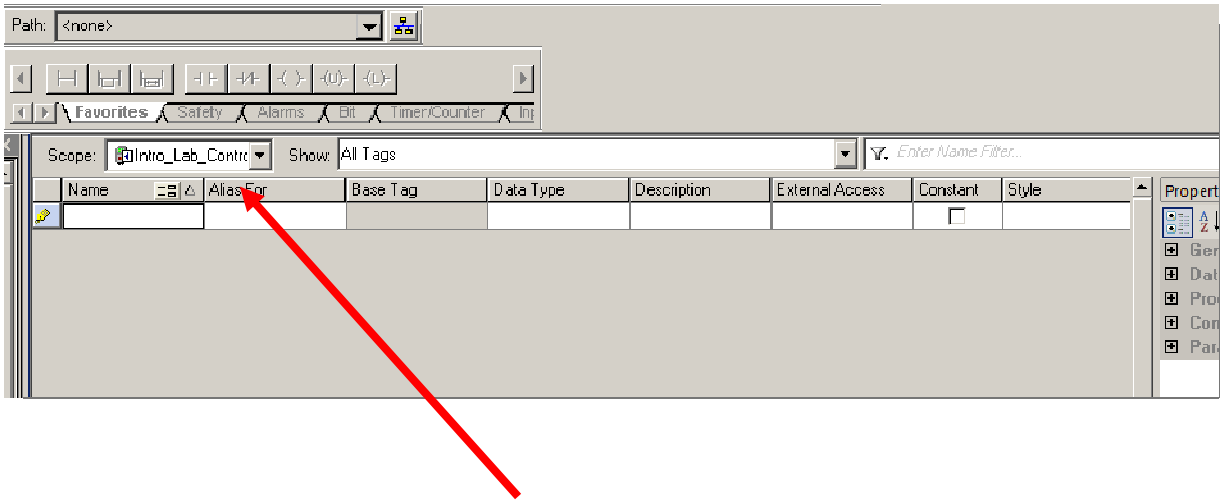
Monitor/Edit Tags Tabs

When the 'Monitor Tags' tab is selected the tag values are shown and new values 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 tags or modifying tag properties, verify that the 'Edit Tags' tab is selected.

Notice that there are no tags present even though you just created three tags. These tags were created at the 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 ***Intro_Lab_Control_Project,*** which will show controller scoped tags.

Data scope defines where you can access tags.

Controller-scoped tags are accessible by all programs. **Parameters and Local Tags** are accessible only by the code within a specific program; Isolate portions of a machine or different stations into separate programs. This lets you do the following:

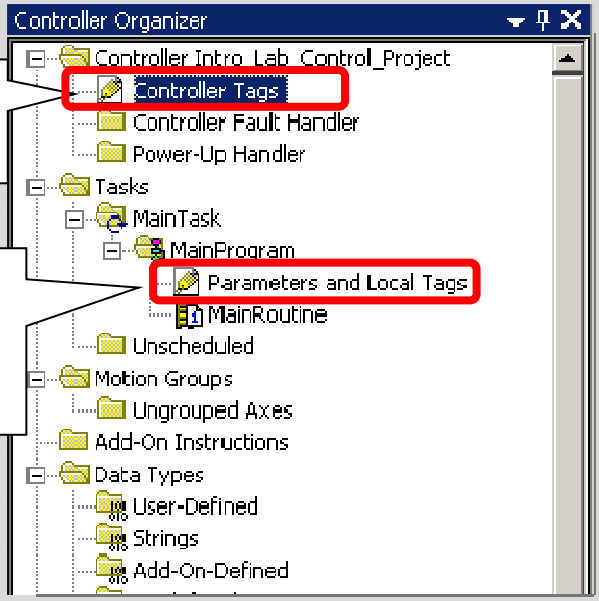
- Provide isolation between programs and equipment phases
- Prevent tag name collisions
- Improve the ability to reuse code

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).

Any program in the controller can access these tags

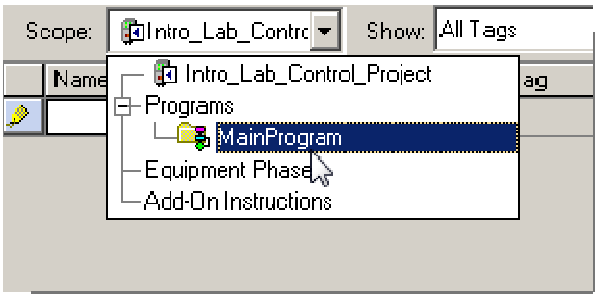
Only the program containing the Local tags can access them. Local tags are also called program scoped tags.



Local tags are isolated from other programs. Routines cannot access the Local tags of another program. Thus you can re-use Local tag names across multiple programs.

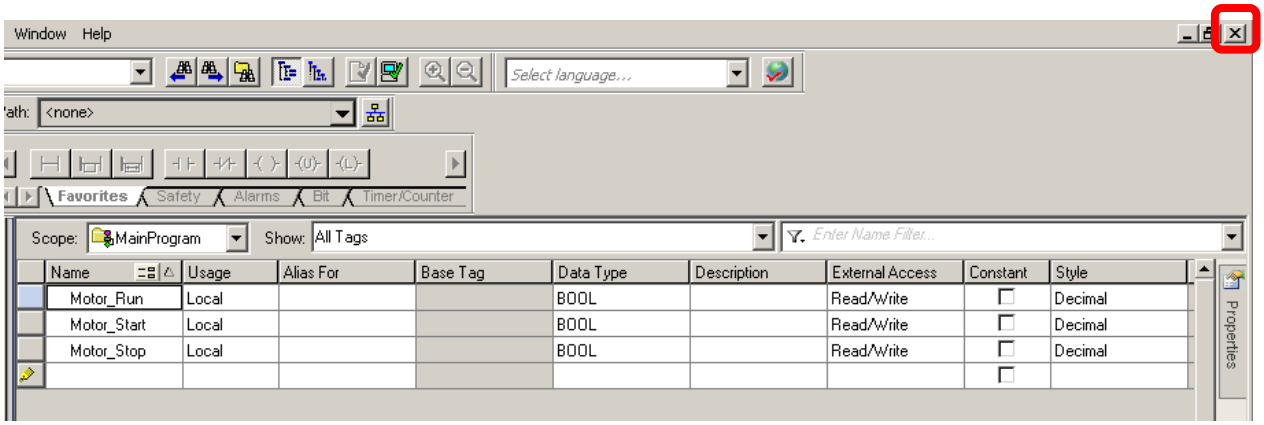
3. Click on the ***down arrow*** for the ***Scope*** selection box.

4. Select ***Programs*** → ***MainProgram***



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

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



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

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

Section 2: Configuring I/O

We will now configure I/O for the project. To communicate with I/O modules you must add modules to the I/O Configuration folder (also referred to as the I/O tree).

This lab section should take roughly 10 minutes to complete.

Objective:

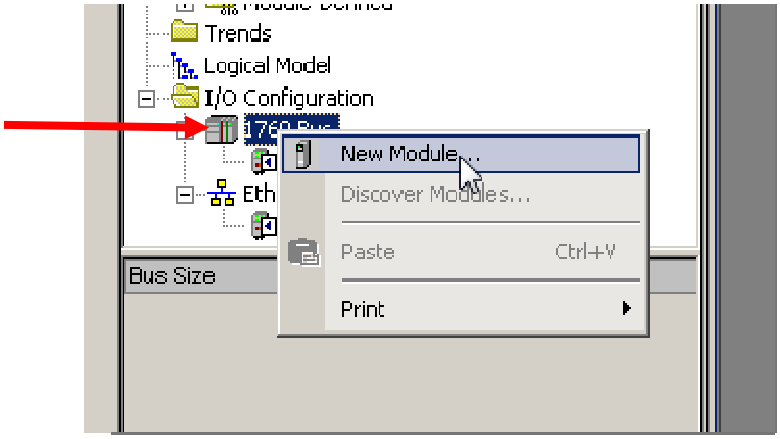
This part of the lab covers adding 1769 I/O using the equipment at your lab station. For this lab we will add the following I/O modules for your lab station.

- 1769-IQ6XOW4 - Combination Digital Input/Output Module
- 1769-IF4XOF2 - Combination Analog Input/Output Module

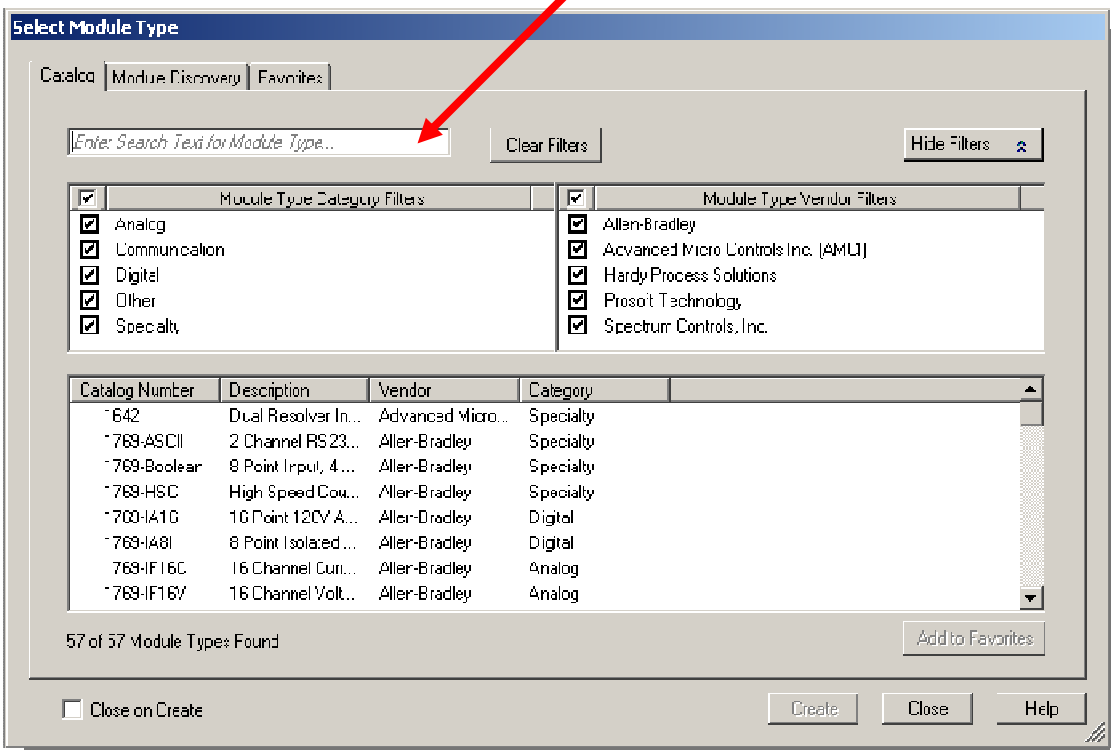
Adding CompactLogix I/O

You will continue to use the project already open.

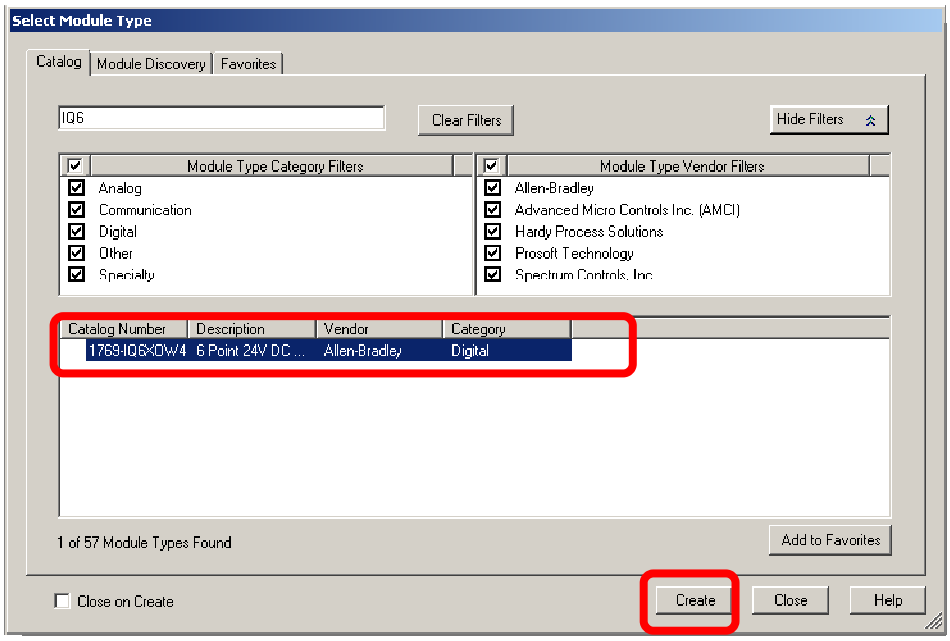
7. In the ***I/O Configuration Folder***, right click on ***1769 Bus*** and select ***New Module***.



8. The ***Select Module Type*** window appears. Type ***"IQ6"*** in the search box.



9. Select the ***1769-IQ6XOW4*** module and click ***Create***

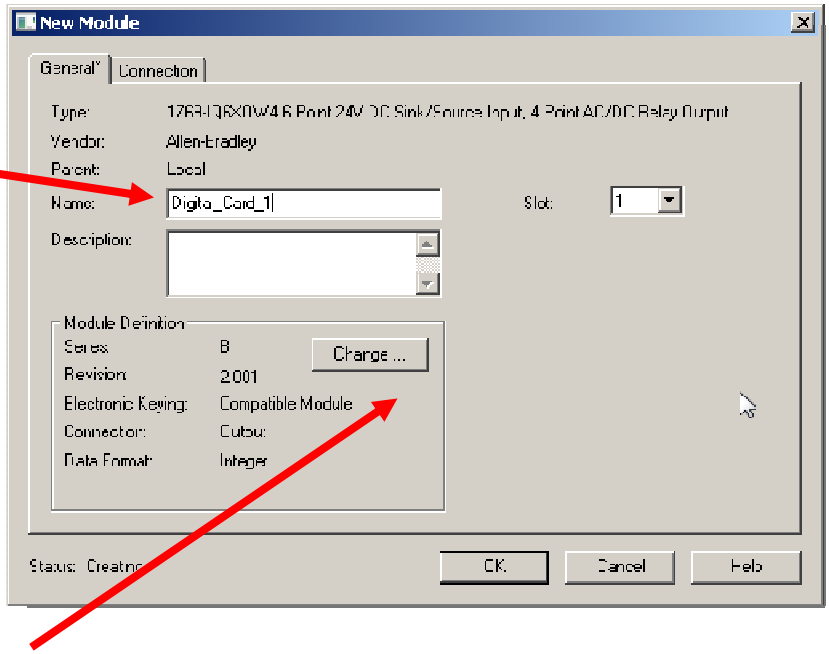


Module Configuration Wizard

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.

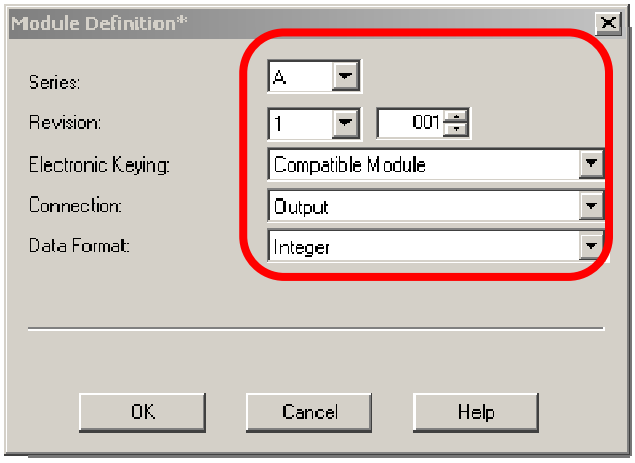
With the Logix family, there are no more dip switches or jumpers needed to configure I/O modules. I/O modules are software configured. This saves time when setting up a system. The configuration for all modules is part of the controller's program and is downloaded to the module from the controller. This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

10. The new module window appears. Enter the Name ***Digital_Card_1*** and ***Slot 1*** parameters as shown below.

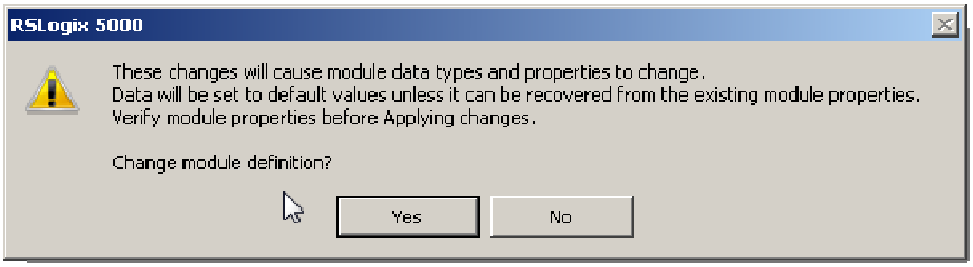


11. Click on the ***Change...*** Button and change the ***Series to A***. Notice the **revision** changes to ***1***. Click **OK.**

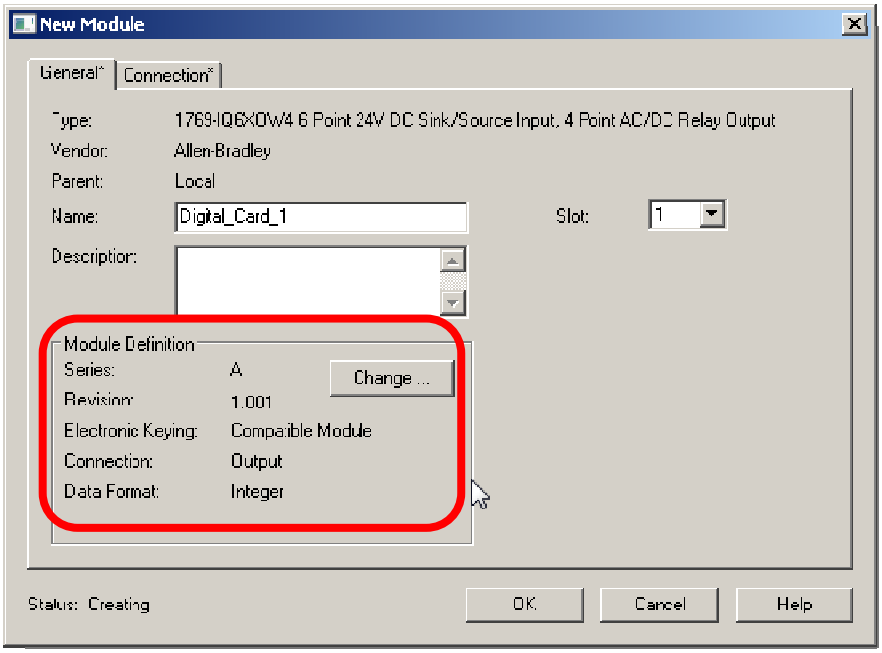
Note: Some of the demos have series B modules. Compatible module keying will allow earlier profile version to work with newer modules.



12. Click ***YES*** on the Change Module Definition Warning that pops up.



13. The properties window should now appear as follows.



Electronic Keying **–** Keying determines what checks are performed between the controller configured I/O tree and the module before an I/O connection is made. This helps guard against improper operation by verifying the hardware matches with what is configured.

The following data is read and compared:

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

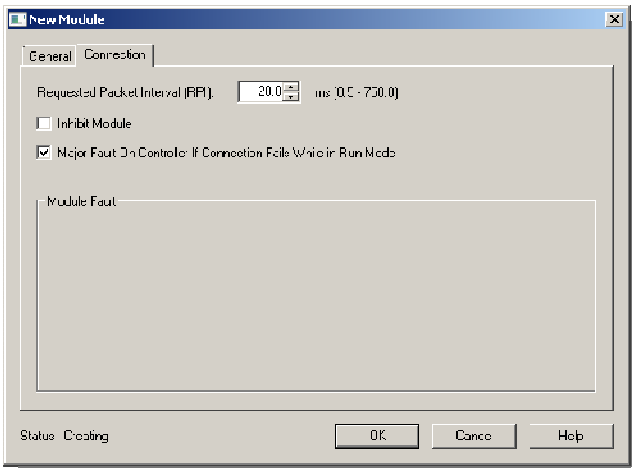
The user may select one of the following module keying options during the initial module configuration:

- *Exact Match* – All of the parameters described above must match or the inserted module will reject the connection.
- *Compatible Module* – The module determines if the settings are compatible. Generally, the IO module checks the Module Type, Catalog Number, and verifies the revision of the hardware is equal to or greater than that configured.
- *Disable Keying* – No keying used at all. This is not typically used.

Connection **--** Input only modules use “Data”. Modules that include outputs use “Output”.

Data Format **--** Determines the data structure for the tags that are associated with the module. With the modules in this lab, the format is integer.

14. Click on the ***Connection*** tab to observe the **Requested Packet Interval** data. We will leave the default at 20 milliseconds.

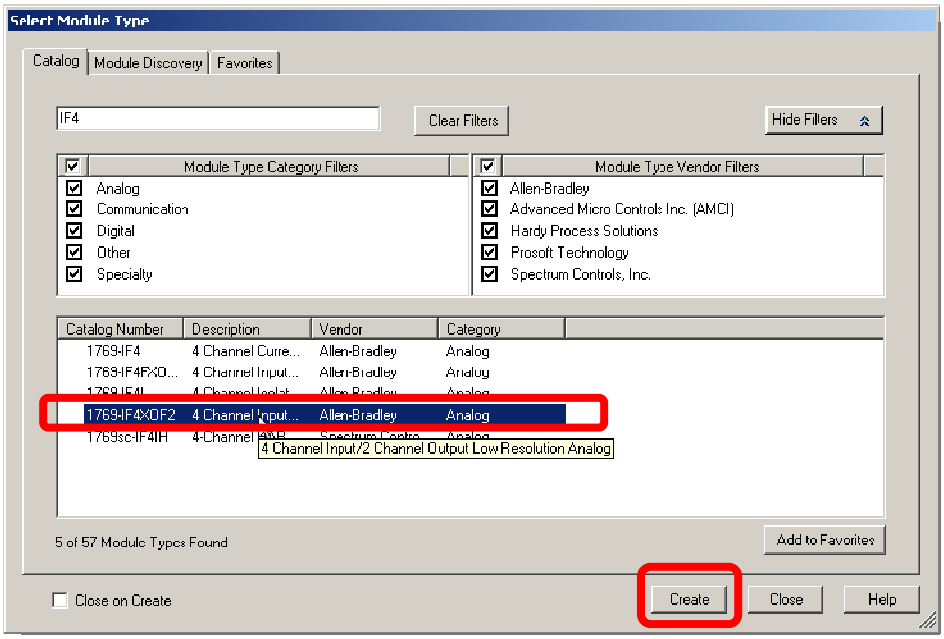


Requested Packet Interval (RPI)

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 .5ms to 750ms.

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

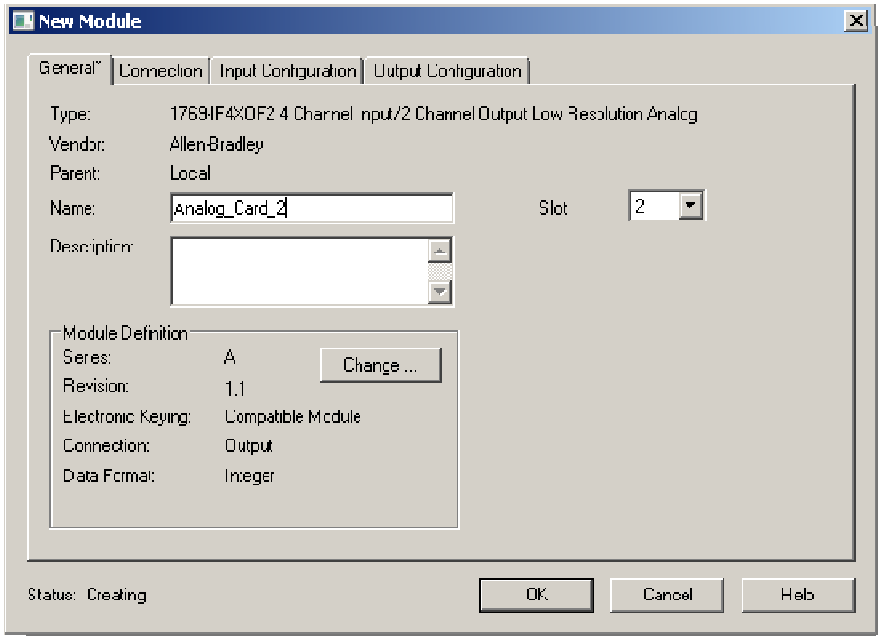
16. In the **Select Module Type** window, type in **"*IF4*"** into the filter box and select the ***1769-IF4XOF2*** module.



Note: Do not select the module with "F" at the end, I.E. do not select 1769-IF4XOF2F (which is shown above as 1769-IF4FXO....). This is a different module then shown in the rack and will give a mismatch error if selected. If this module is selected, it will need to be deleted to put in correct module.

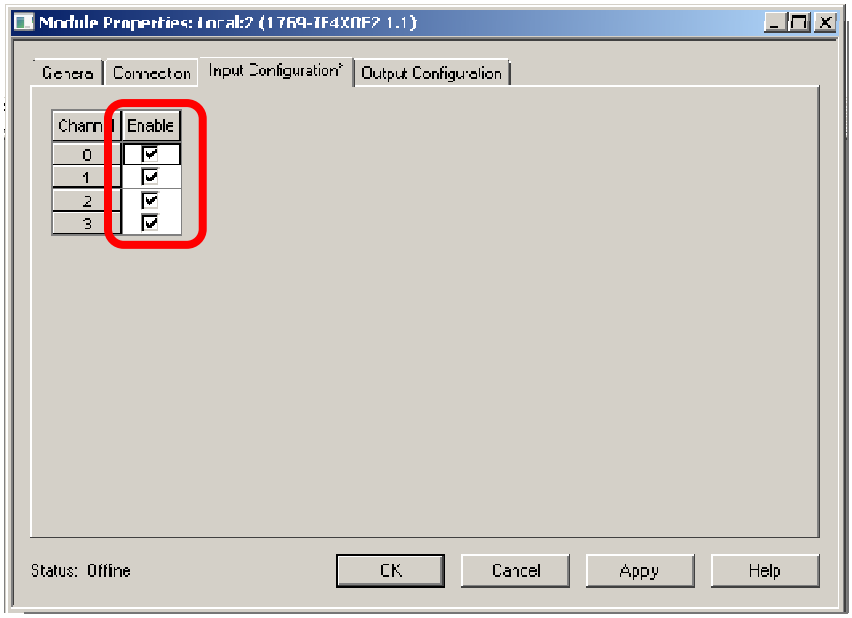
17. Click on ***Create***. The new module window appears.

18. Fill in the name **"*Analog_Card_2*"**



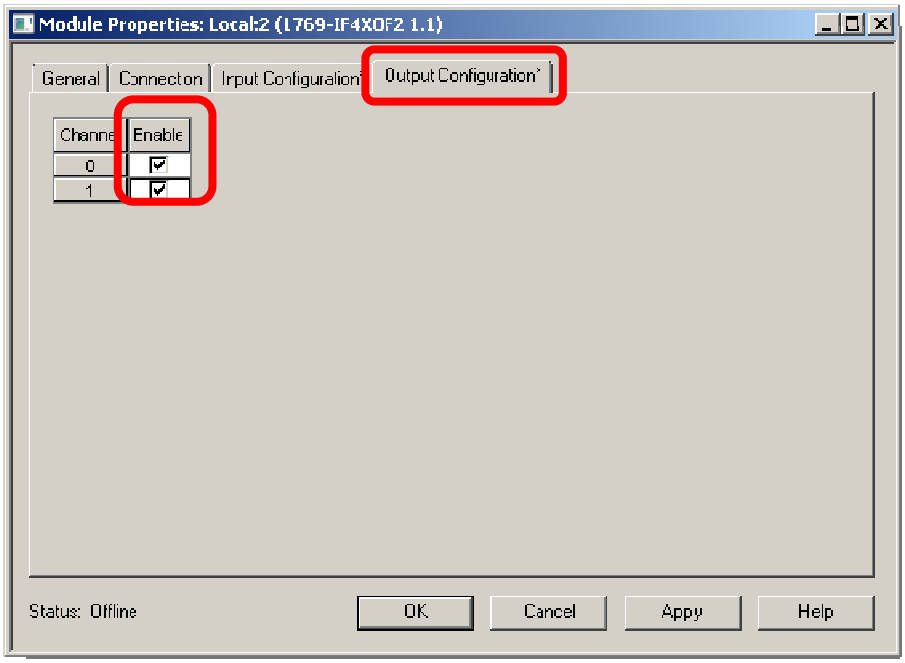
19. Click on the ***Input Configuration*** tab.

20. Enable all 4 channels by clicking a ***check*** **in each box**.



21. Click on the ***Output Configuration*** tab.

22. Enable both channels by clicking a ***check*** in each box**.**



23. Click ***OK***

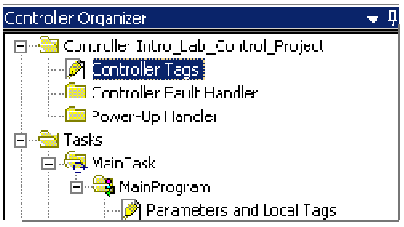
24. *Close* the **Select Module Type** dialogue.

Viewing the CompactLogix 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.

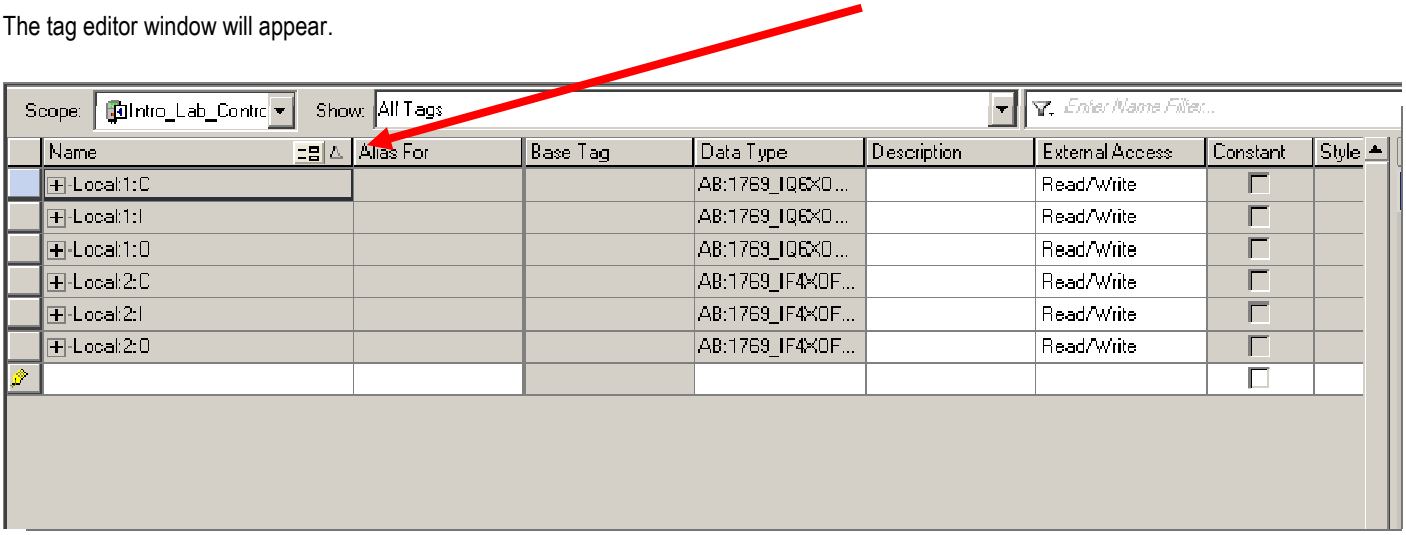
You will continue to use the project already opened.

25. From the Controller Organizer double click on ***Controller Tags***.

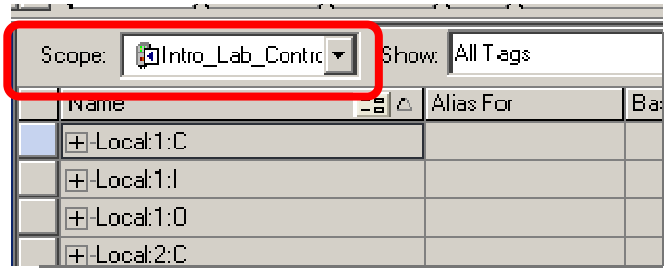


If necessary, drag to the right to increase the size of the Tag Name field. This will allow you to view the entire Tag Name.

The tag editor window will appear.



Notice by looking in the upper left corner of the tag editor that Controller Scope is selected. All I/O module tags are created in the Controller Scope. Modules that reside within the controller chassis are called "Local".



FYI - I/O Address Format

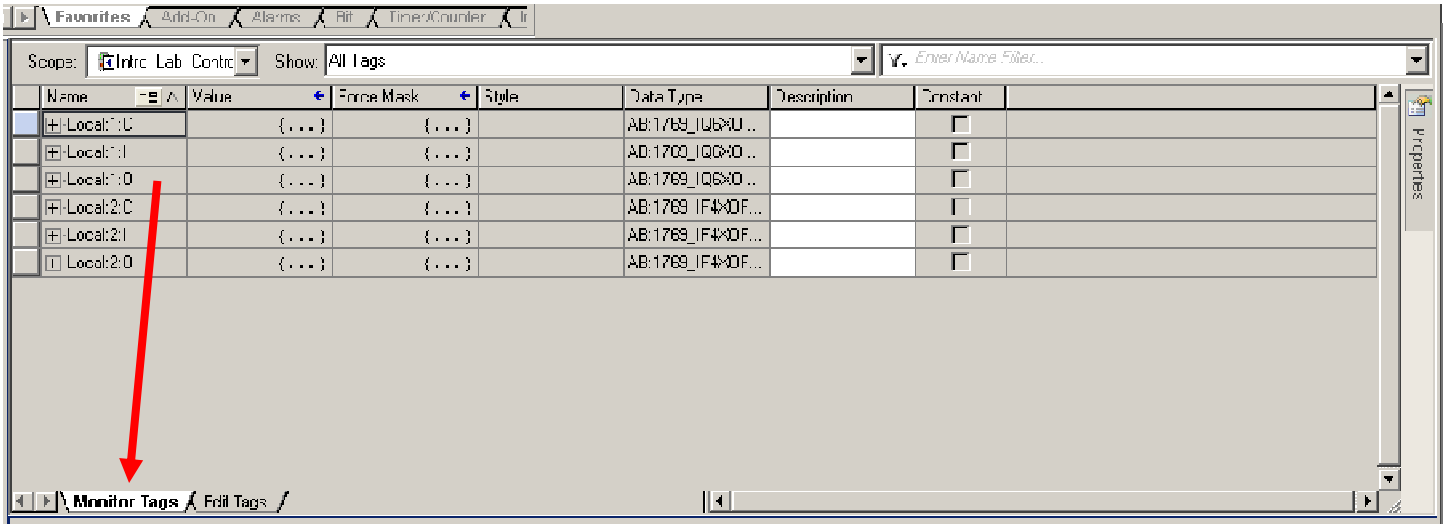
An I/O address follows this format:

`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)

26. 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.

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

- **:C** - Configuration tags hold the module configuration and are designated by a **":C"** in the tag name.
- **:I** - Input tags have **":I"** in the tag name.
- **:O** - Output tags have a **":O"** in the name**.**

28. 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.

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:

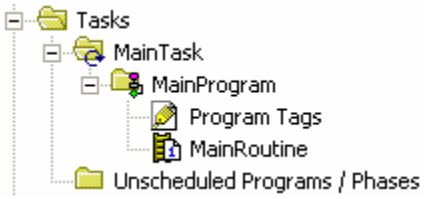
-Program logic in advance of wiring diagrams

-Assign a descriptive name to an I/O device

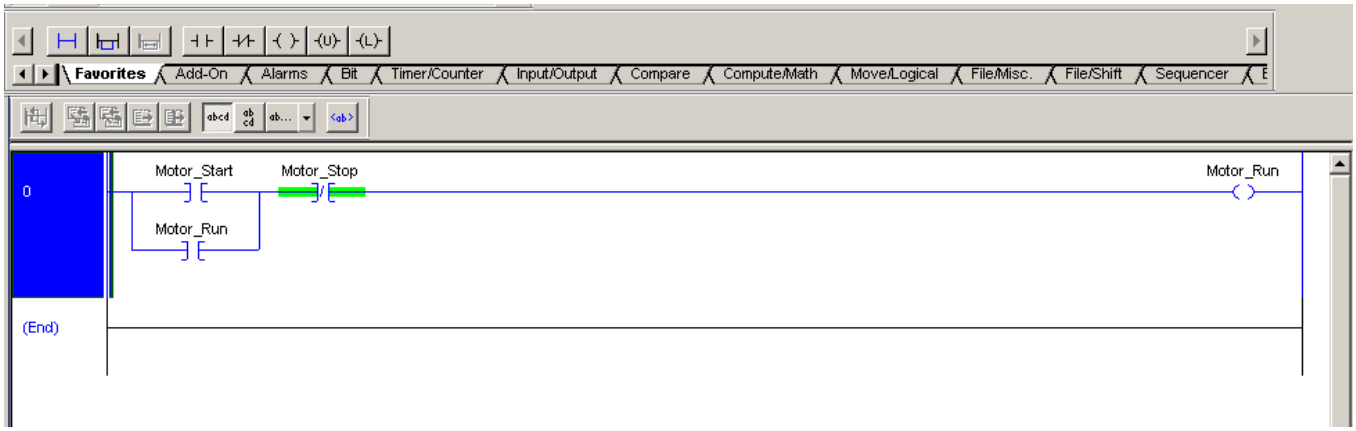
-Provide a simpler name for a complex tag

-Use a descriptive name for an element of an array

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

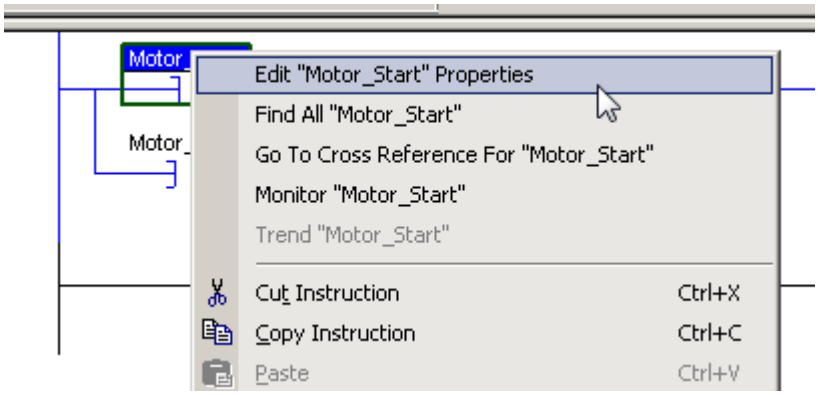


The ladder editor appears as shown below:



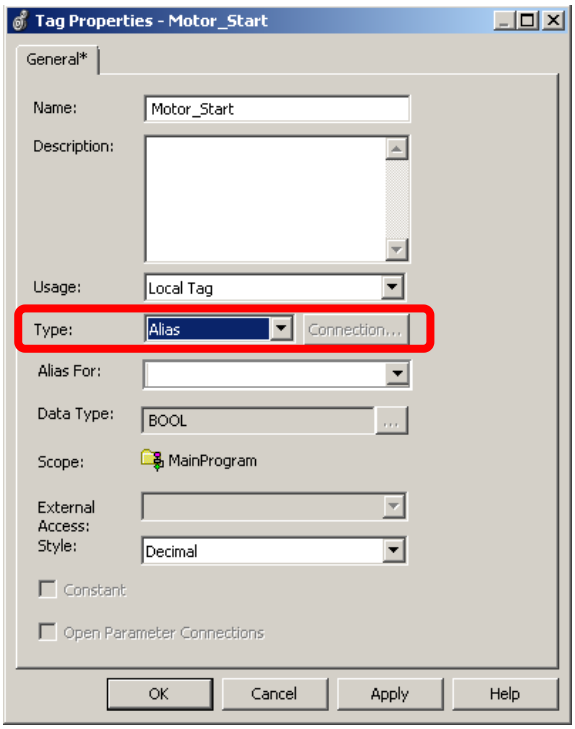
In the last part of the lab we added I/O modules to the project. Now it's time to Alias the tags in the program to the I/O Modules. This will connect the ladder logic to real world I/O points. All three Motor tags will be aliased to points on the 1769-IQ6XOW4 module.

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



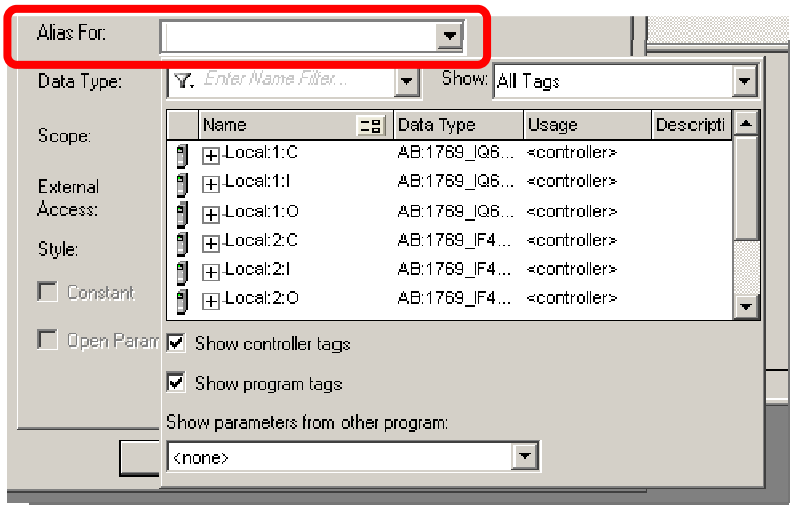
The **Tag Properties** window for Motor_Start will appear. Currently the tag is defined as a Base tag.

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



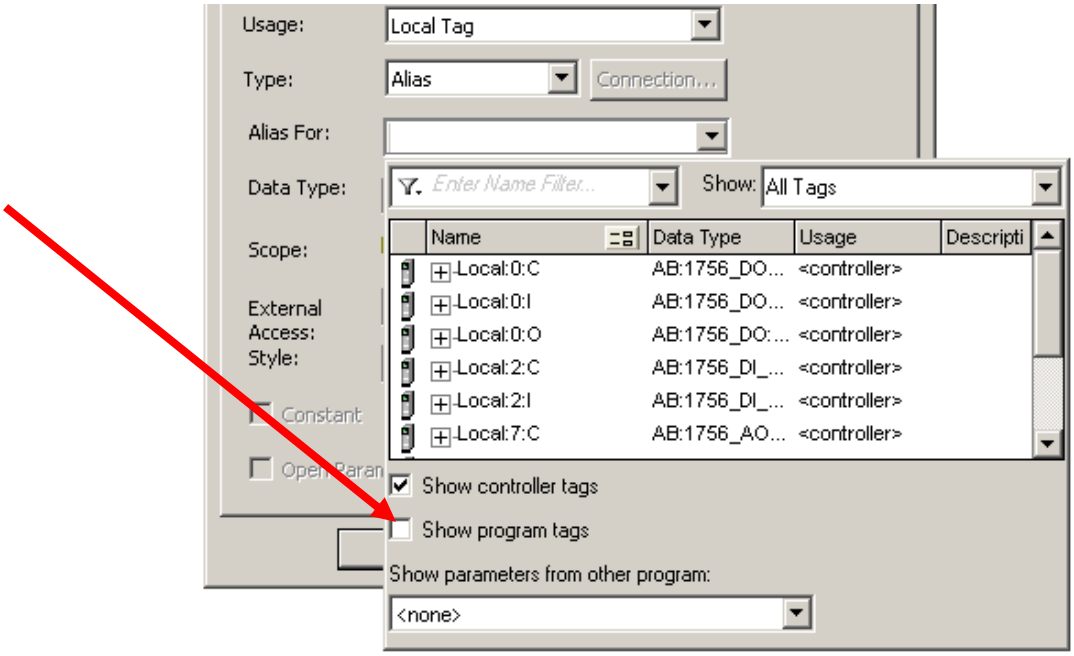
32. Click on the ***down arrow*** for **Alias For**.

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



33. 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

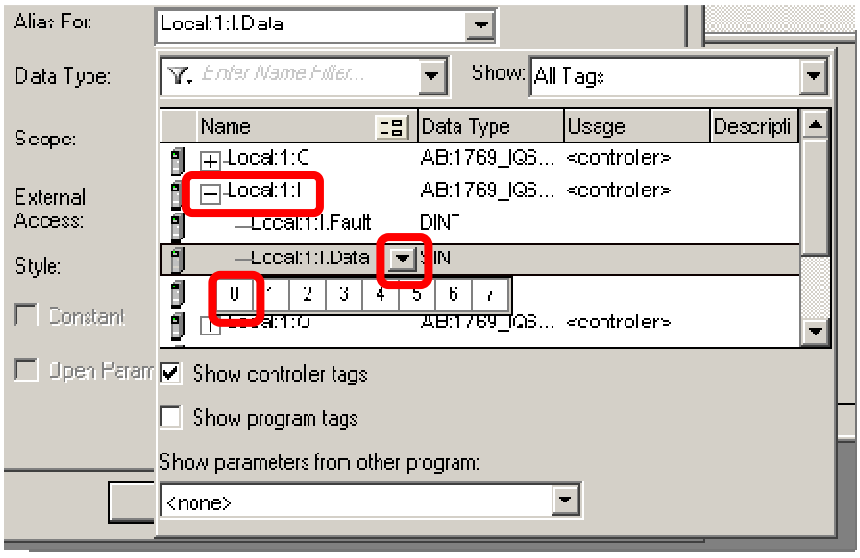


34. Expand ***Local:1:I*** by clicking on the **+** sign and select ***Local:1:I.Data.***

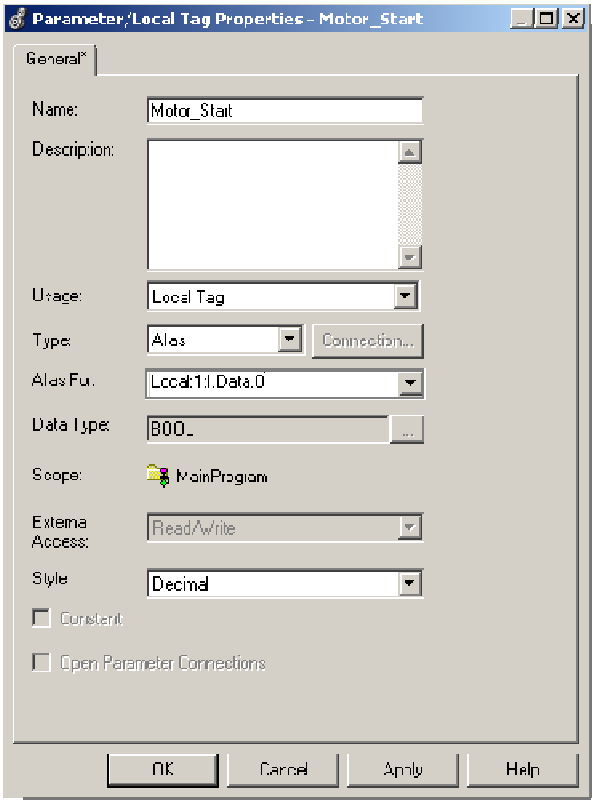
35. Click the ***down arrow*** for ***Local:1:I.Data*** as shown below.

This will open the table of data points for the 1769-IQ6XOW4 module.

36. Select ***0*** from the table.



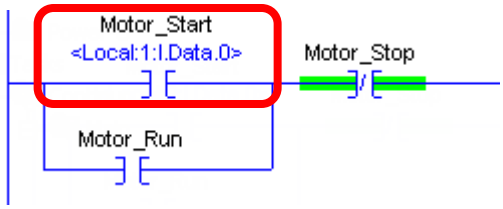
When you select ***0*** from the tag browser the window will close. **Tag Properties** will now appear as follows:



Motor_Start will now be aliased to **Local:1:I.Data.0**, which is the first input point on the 1769-IQ6XOW4 module.

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

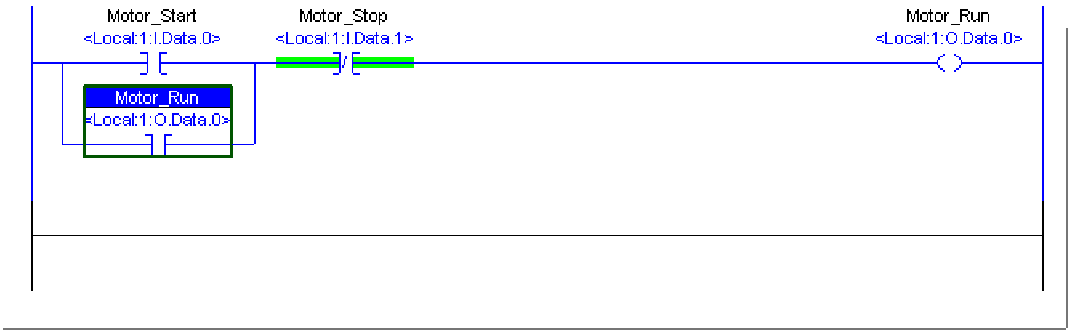
Motor_Start has been Aliased to Local:1:I.Data.0. This means that the tags are equivalent to one another in code. It is much easier to understand Motor_Start than Local:1:I.Data.0.



38. Using the previous steps, alias the remaining two tags.

- ***Motor_Stop = Local:1:I.Data.1***
- ***Motor_Run = Local:1:O.Data.0***

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



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

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

Section 3: Connecting Your Computer to the Controller

This lab section should take roughly 5 minutes to complete.

Objective:

In this lab, we will learn to configure a driver in RSLinx Classic communications software. We will complete the following steps:

- Launch RSLinx Classic communications software
- Configure a communications driver

Launching RSLinx Software

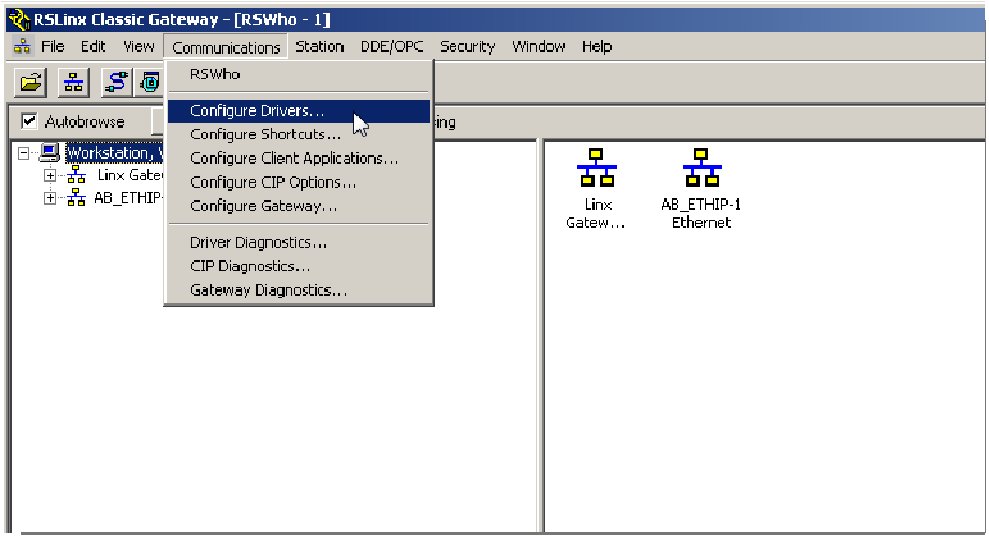
In this section of the lab, you will launch the RSLinx software, which will enable you to configure the driver you will use to communicate with the CompactLogix processor in the Demo Box. .

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

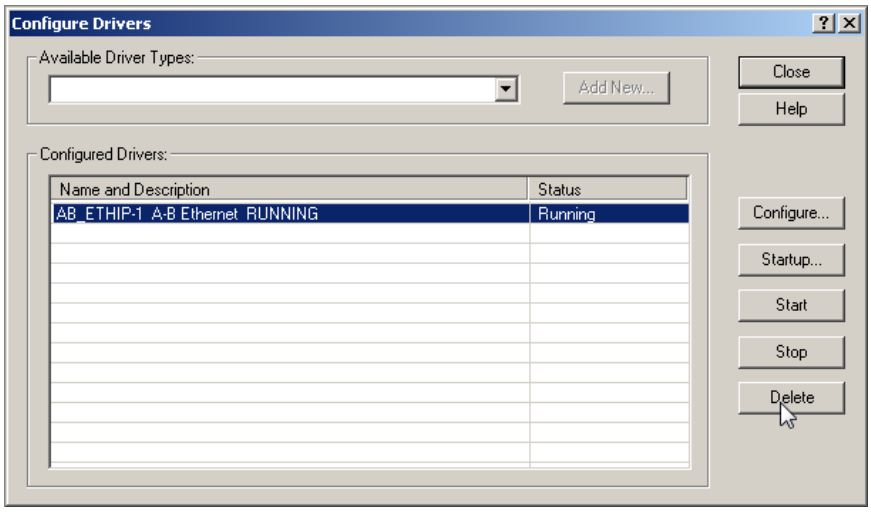
Adding the AB_ETHIP (Ethernet/IP) Driver

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

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

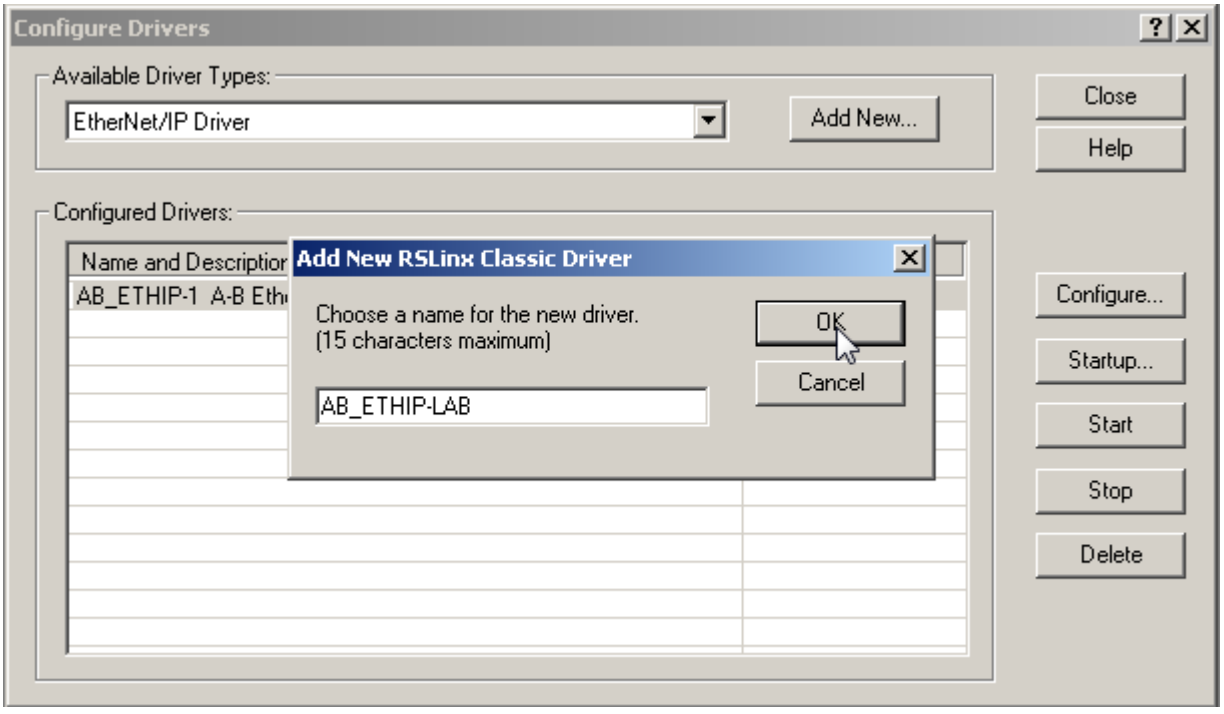


The Configure Drivers dialog appears. There may already be a driver configured on this lab. However, we are going to create a new driver. RSLinx Classic allows multiple drivers to be used.

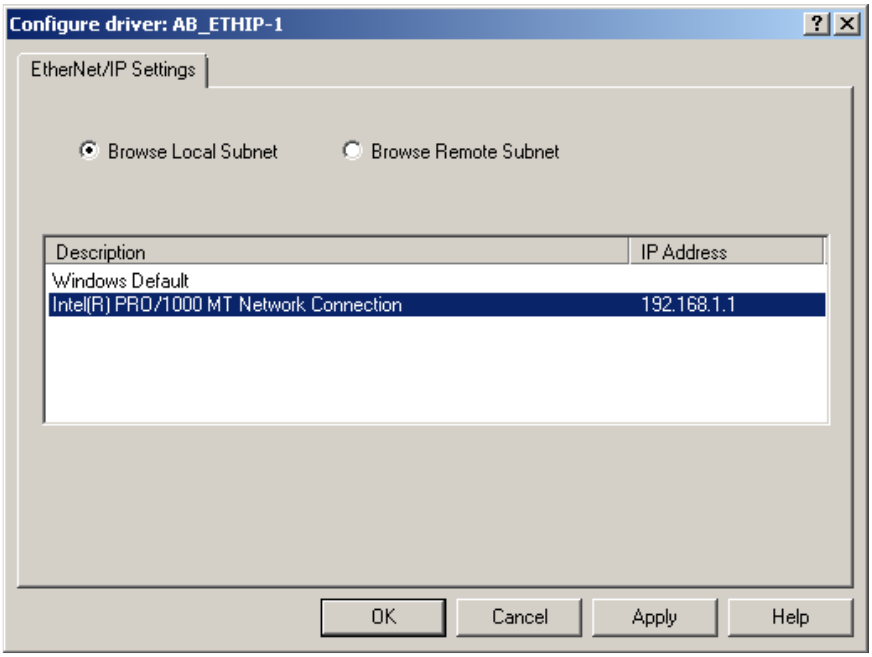


3. From the ***Available Driver Types*** pull-down menu, choose ***EtherNet/IP Driver*** then click on the ***Add New*** button.

4. Change the name of the driver from ***AB_ETHIP-1*** *(or* ***AB_ETHIP-2****)* to ***AB_ETHIP-LAB*** as shown and click ***OK***



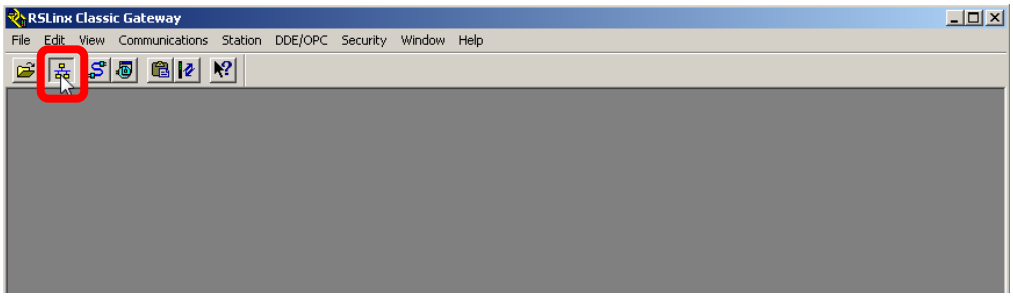
5. Choose ***"Browse Local Subnet"*** and the ***"Intel"*** network driver as shown. Click ***OK***.



6. 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 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 addresses 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.

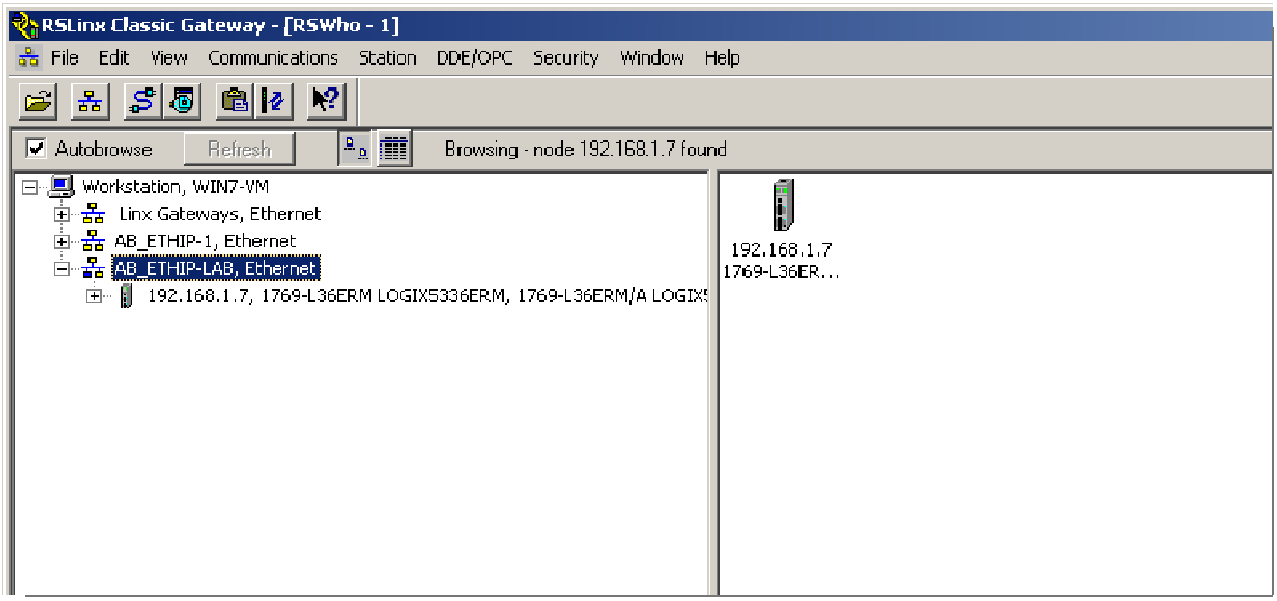
7. If no drivers are shown the software background is grey, click the ***RSWho*** icon  in the toolbar. Otherwise move on to the next step.



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

8. Expand the ***AB_ETHIP-LAB, Ethernet*** driver to see the Ethernet module with IP 192.168.1.7

This is the RSLinx driver we will use in Studio 5000 Logix Designer to download to the Logix controller in the next section.



> ***FYI*** **- RSWho**
>
> The RSWho screen is actually a RSLinx 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 3. Please move on to Section 4.

Section 4: Downloading the Project from the Computer to the Controller

This lab section should take roughly 10 minutes to complete.

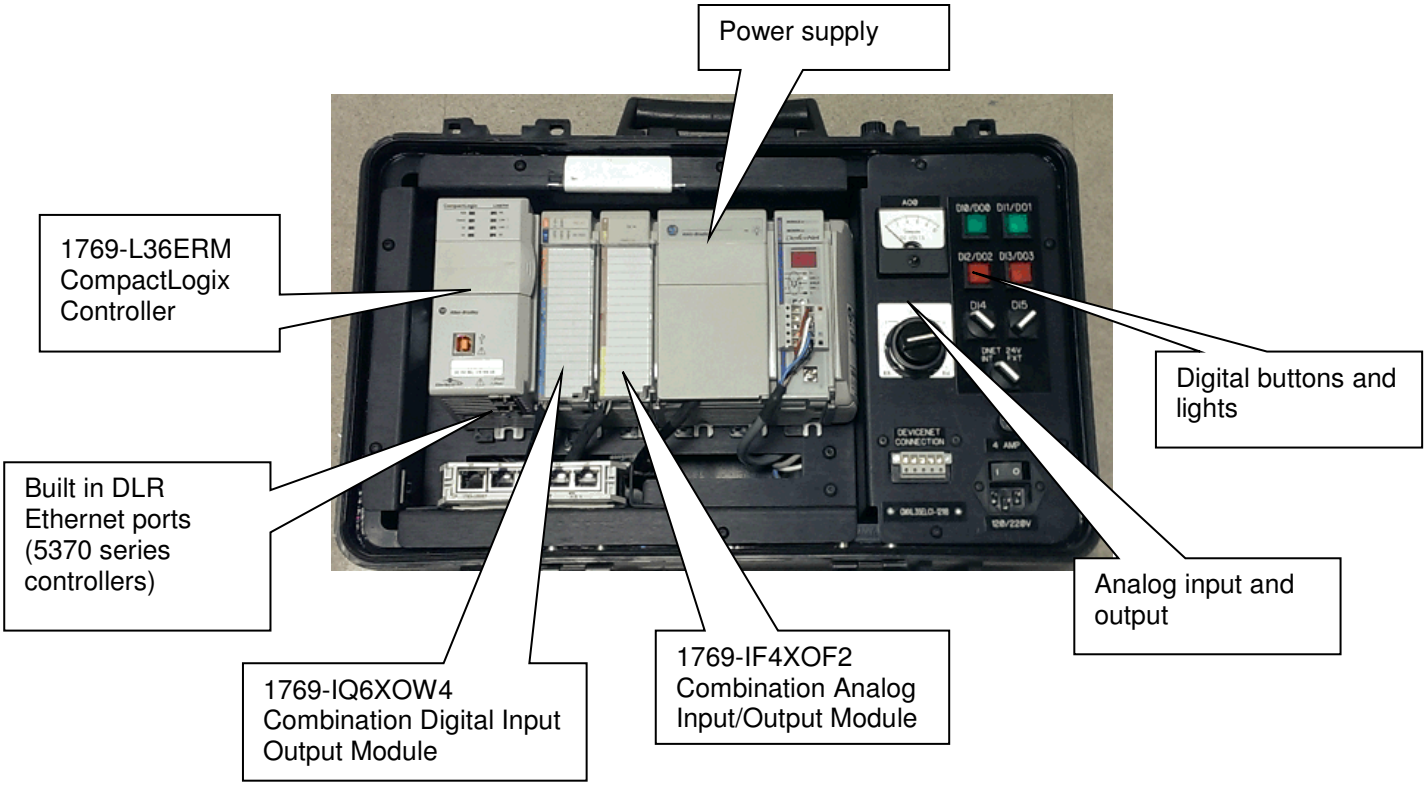
Objective:

In this lab you will:

- Download the program to the controller

You will continue to use the currently open program.

The image below describes the parts of the lab station demo.



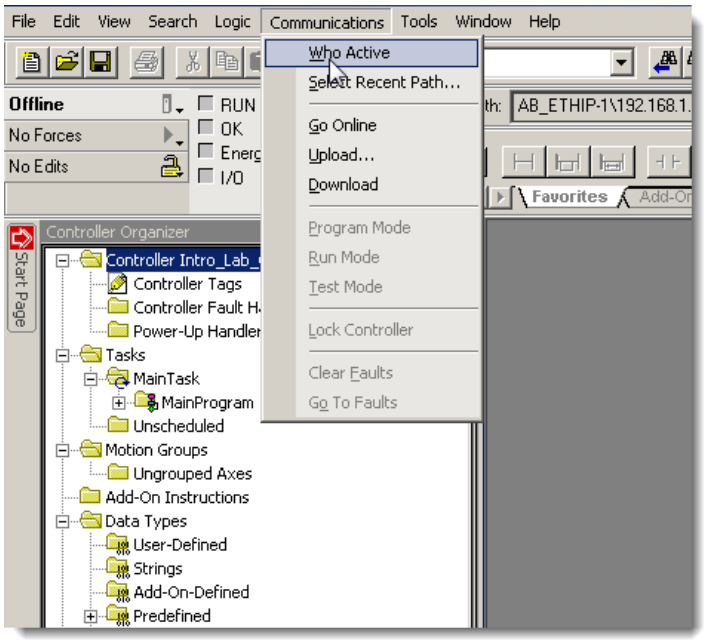
NOTE: Demo boxes may vary

Downloading the Project to the Controller

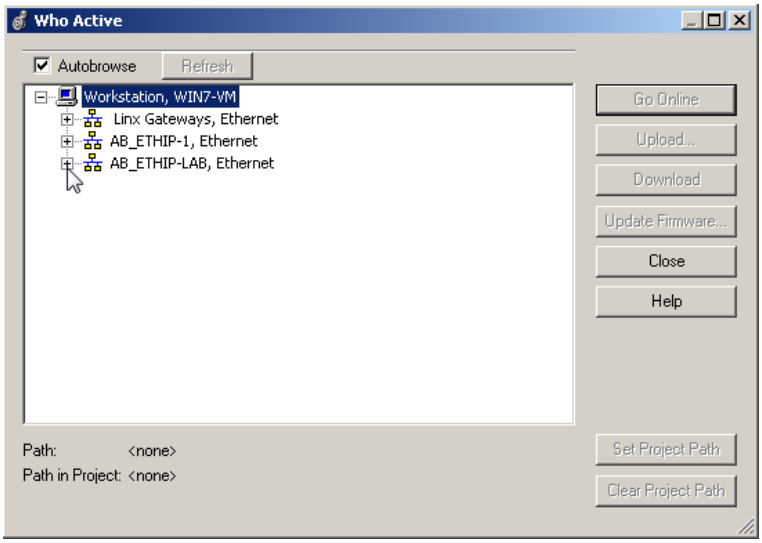
In this section of the lab you will download the project.

1. ***Maximize*** Logix Designer.

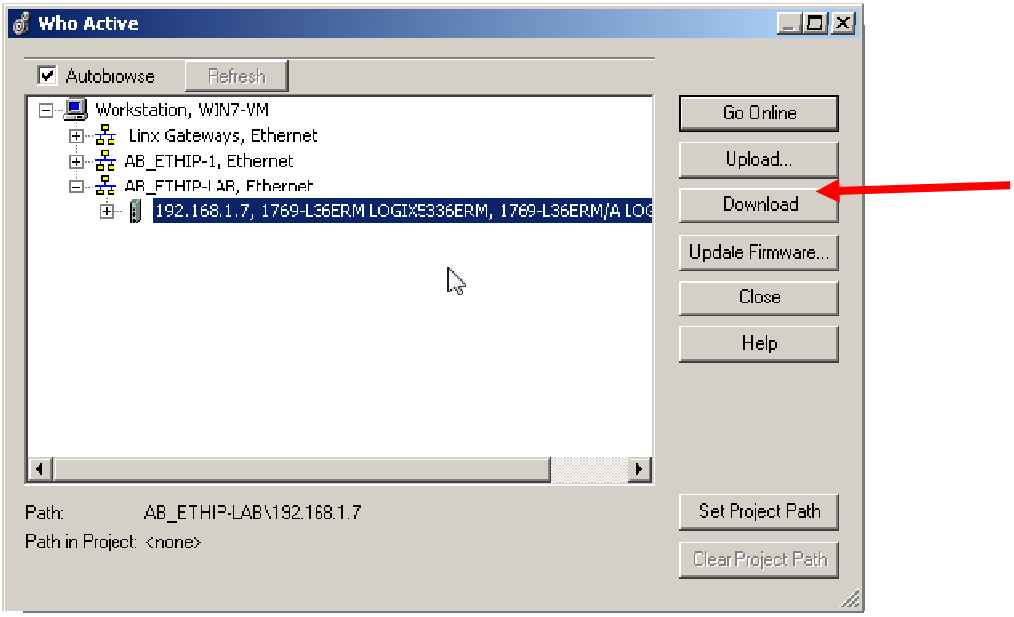
2. From the ***Communications*** menu, choose ***Who Active***.



The **Who Active** Screen appears.

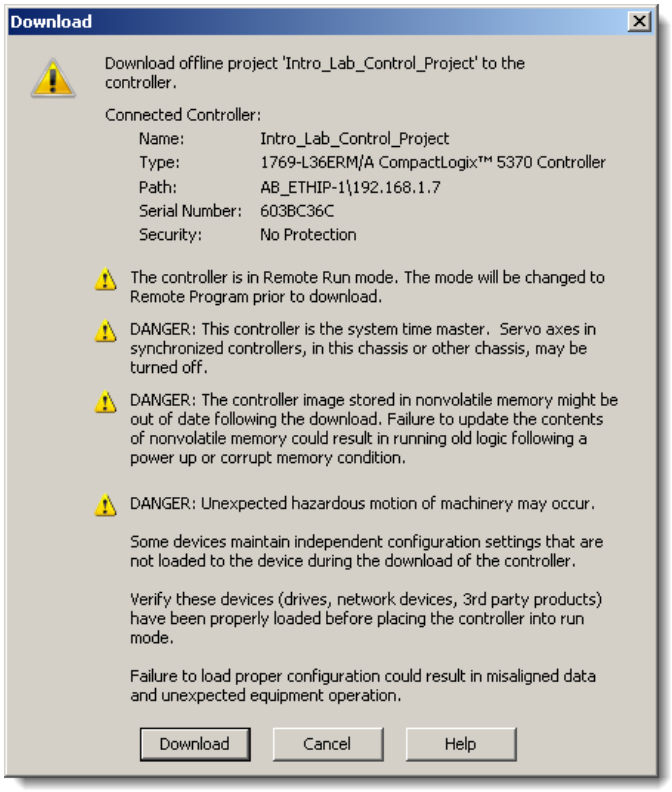


3. Expand the ***AB_ETHIP-LAB*** driver and select the ***1769-L36ERM*** controller by clicking on it.



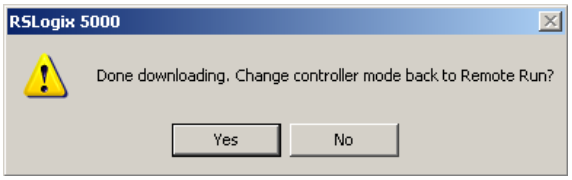
4. Click ***Download***. You will be asked to verify the download. Click ***Download*** again.

The project will then begin to download to your controller.



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. In this case the green color represent run mode. Blue would signify program mode. Grey means not connected to a controller.

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.

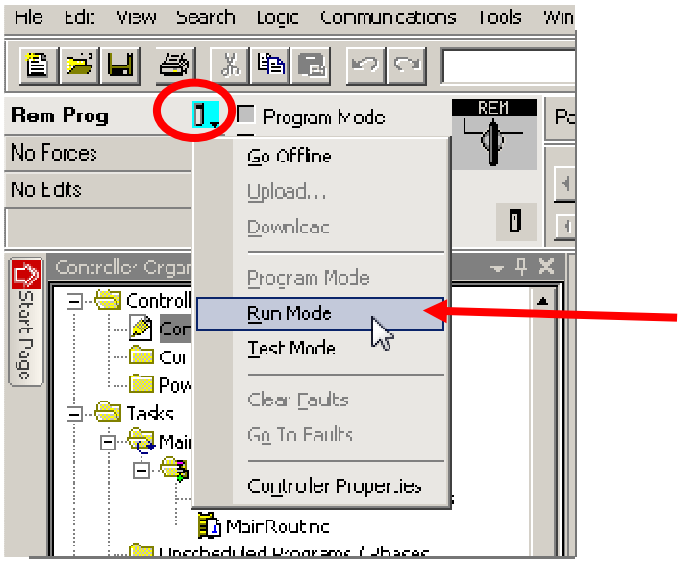
Objective:

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

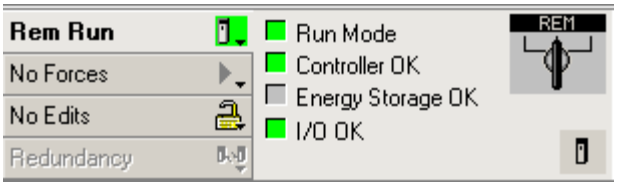
> **I/O Mapping**
>
> For the lab there are a group of push buttons on the Demo Box. The push buttons are mapped as follows:
>
> Motor_Start = DI0
>
> Motor_Stop = DI1
>
> Motor_Run = DO0

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***.

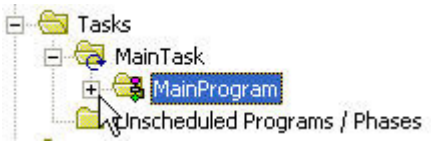


The controller will go into run mode. This can be verified by looking at the Run LED on the controller. It should now illuminate green. It can also be verified through Studio 5000 by viewing the controller faceplate.

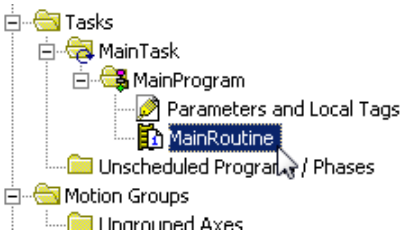


Notice that the faceplate shows four controller status LEDs.

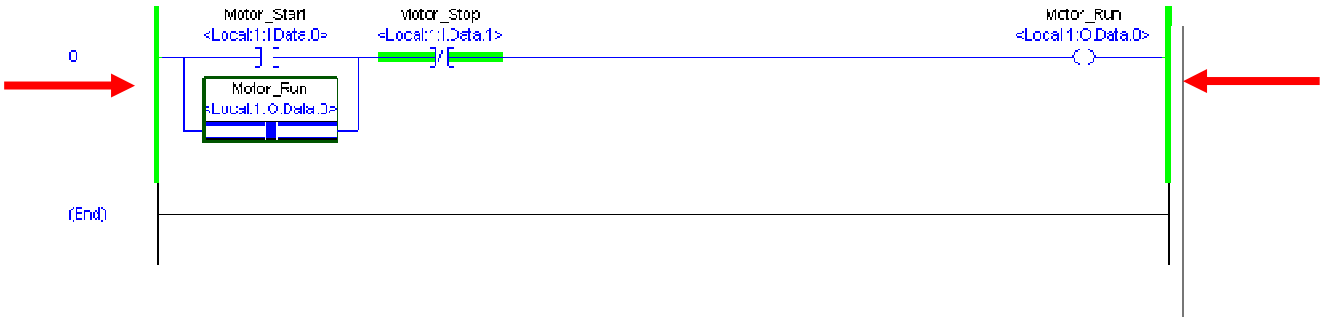
2. From the Controller Organizer expand the ***MainProgram*** by clicking on the "**+**"**.**



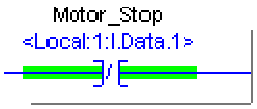
3. Double-click on the ***MainRoutine*** to open the ladder editor.



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.

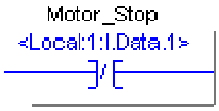


Notice that the XIO instruction Motor_Stop is green. This means that this instruction is in the 'true' or 'on' state. This is because the Motor_Stop Pushbutton is not pressed.



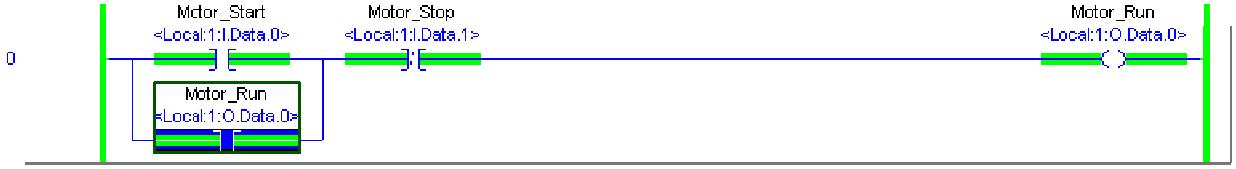
4. Press the ***DI1*** button on the Logix pushbutton panel.

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



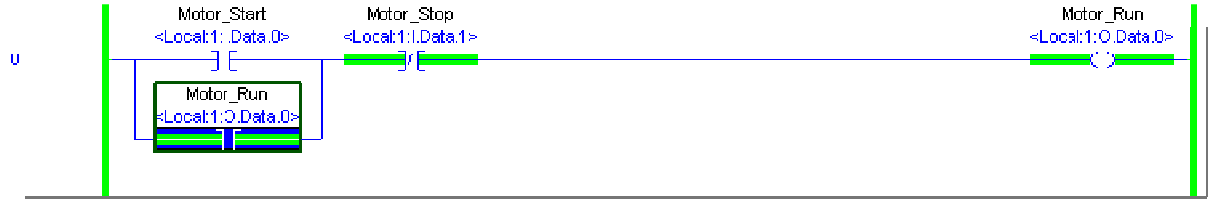
5. Press button ***DI0*** (Motor_Start).

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.

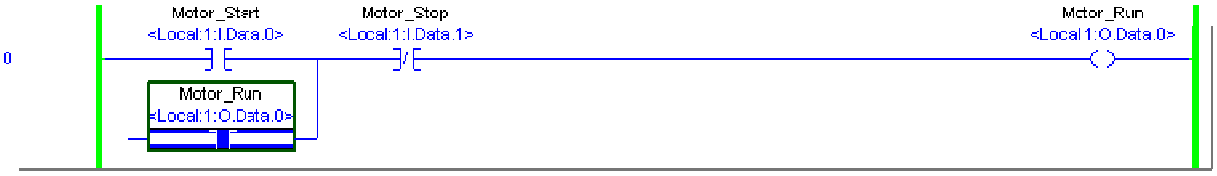


6. Verify that output **DO0** (Motor_Run) stays illuminated when you release pushbutton ***DI0*** (Motor_Start).

The ladder logic you have just written is a simple 3-wire control or motor start/stop seal-in circuit.



7. 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:

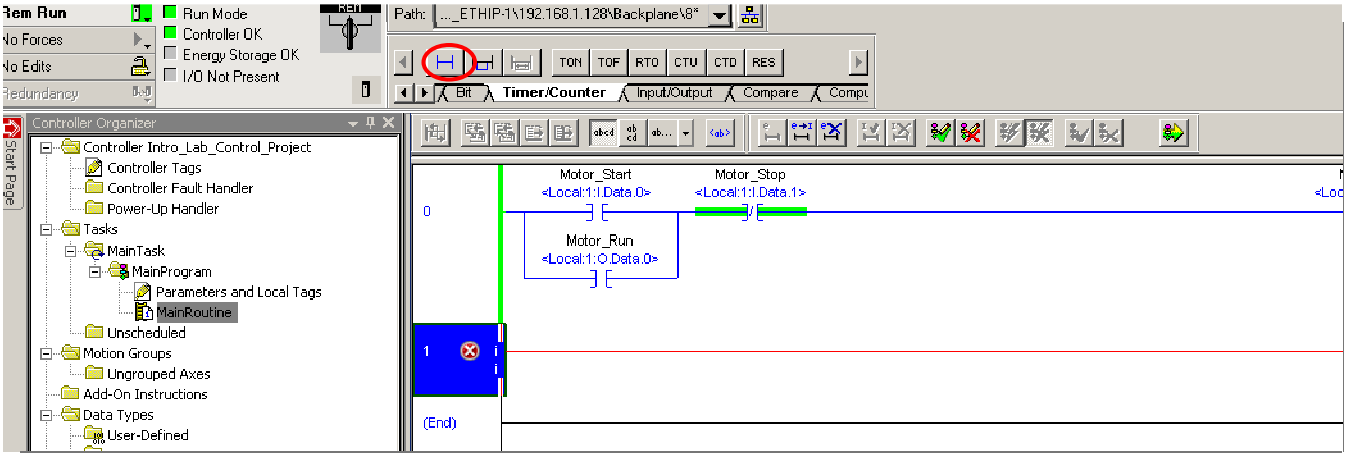
- Add a MOV instruction
- Add a timer to the logic and its execution will be based on the motor running
- Add ladder logic to reset the timer when the motor is stopped.

You will continue to use the project already opened.

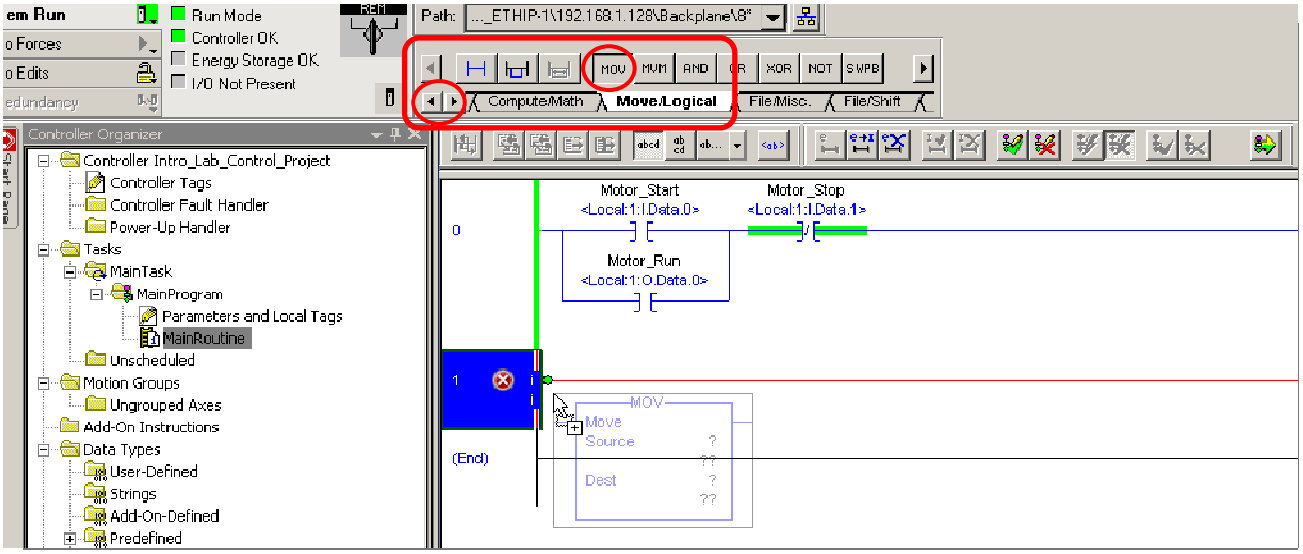
Adding a MOV Instruction to the Logic

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

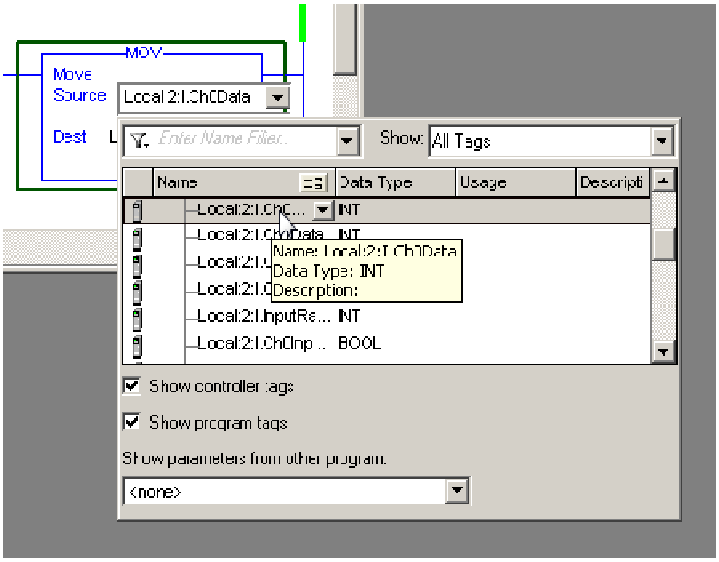
2. Add a rung by clicking the ***rung button***  on the toolbar.



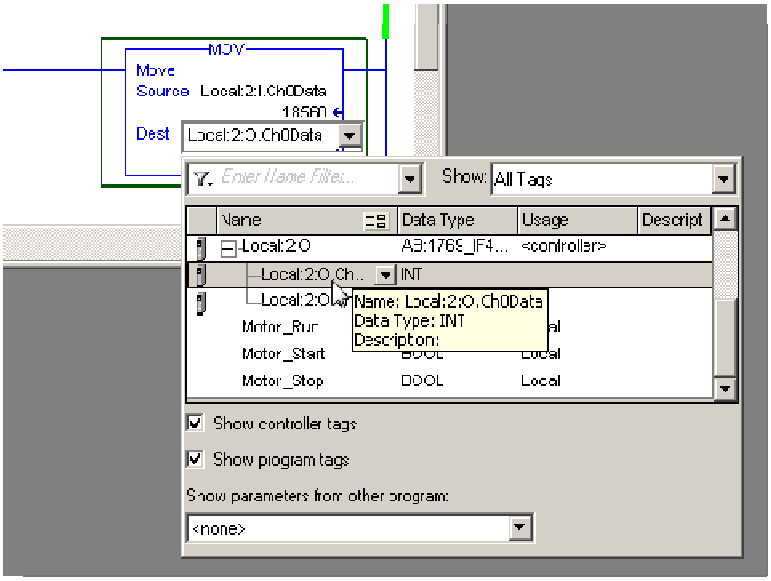
3. Use the ***scroll buttons*** if necessary to scroll to the ***Move/Logical*** instruction group tab in the instruction toolbar. Under the **Move/Logical** category tab, click and drag a ***MOV*** instruction to the new rung.



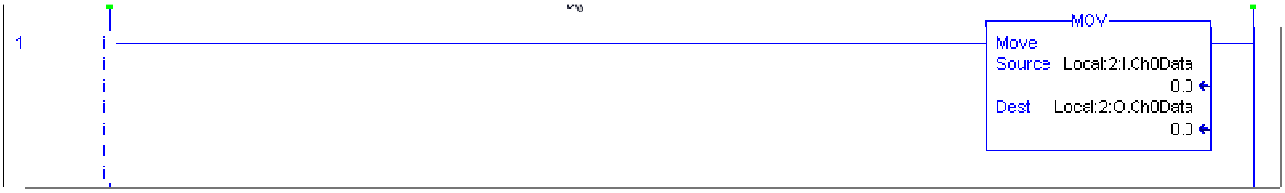
4. Double click the '***?'*** by the source in the MOV instruction and select ***Local:2:I.Ch0Data*** by double-clicking the tag. You might have to ***scroll down*** to find the Channel data tags.



5. Double click the '***?'*** by the destination. Select ***Local:2:O.Ch0Data*** by double-clicking the tag.

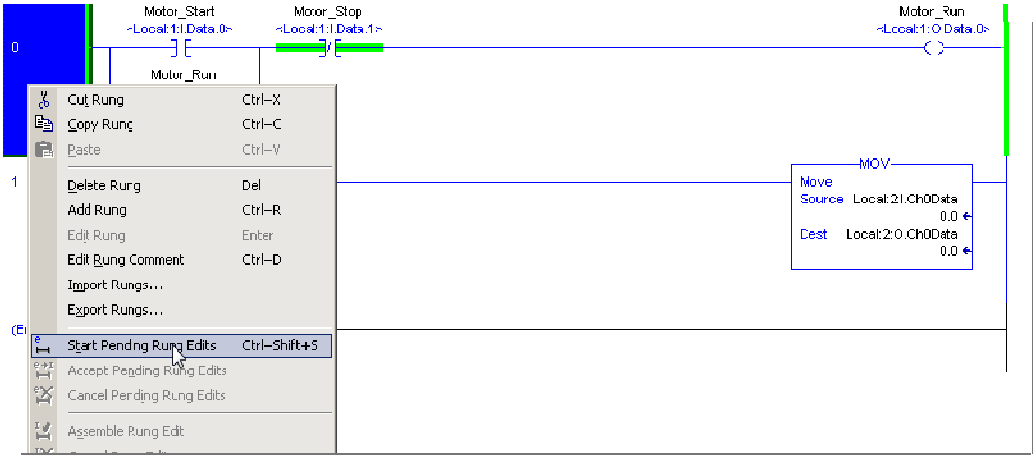


6. The rung should look like the following.

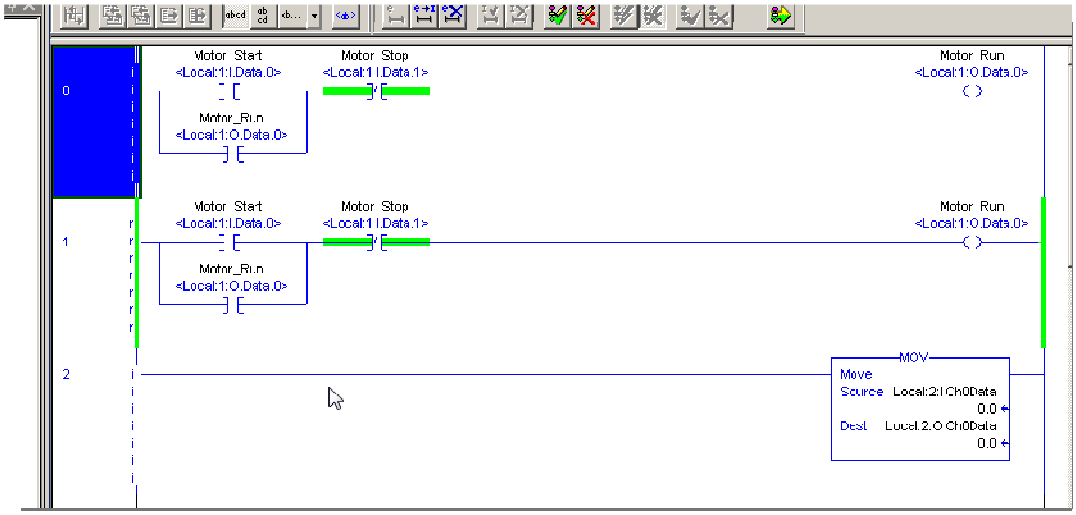


Adding a Timer to the Logic

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

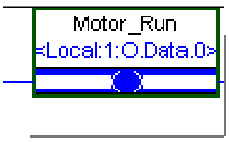


The ladder editor will now look similar to the following:

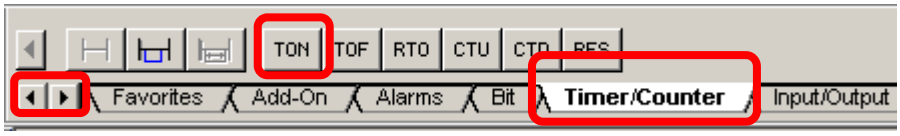


The rung with the lower case 'i's on the power rails is the rung you will perform the edits on.

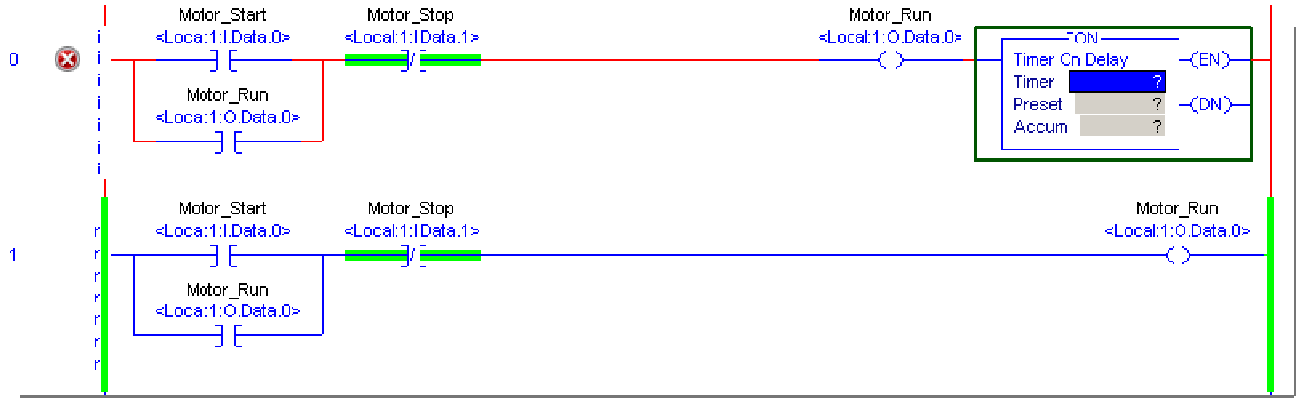
8. Click the ***OTE*** instruction so it becomes highlighted.



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

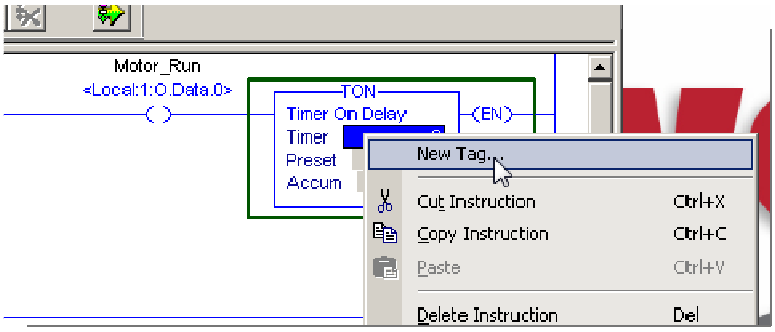


A timer is inserted into the code to the right of the OTE instruction.



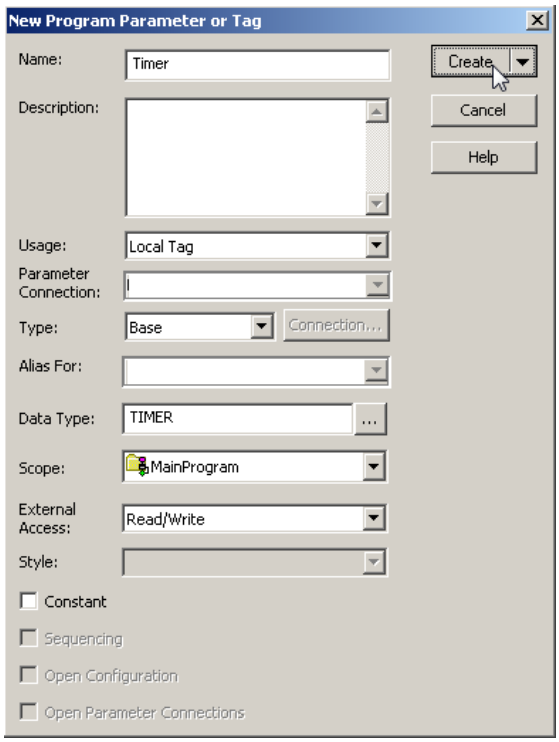
> In ***Studio 5000 Logix Designer*** you can string output instructions together in series. Branches are not required.

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

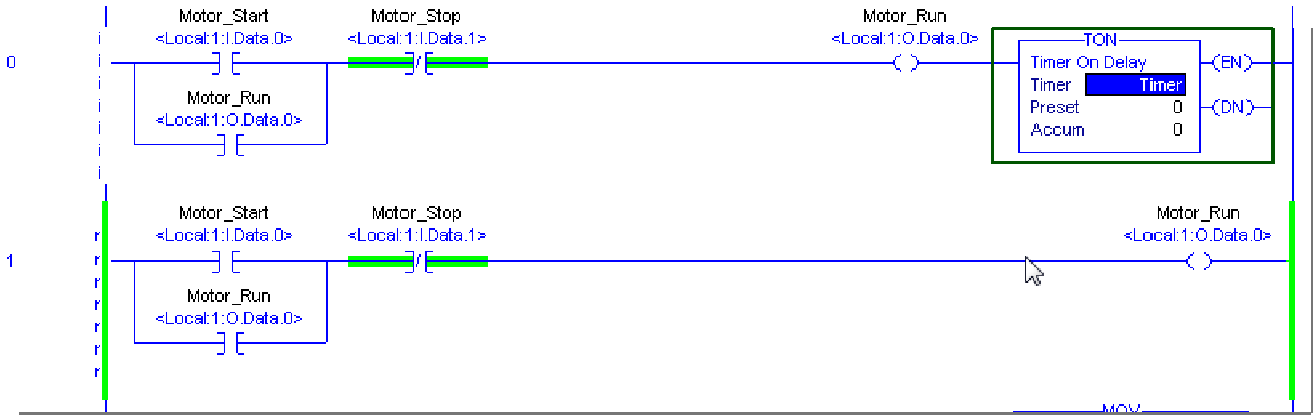


The New Tag window appears. Notice that the Data Type is already set to **TIMER**. This is because you are creating a tag in the timer instruction.

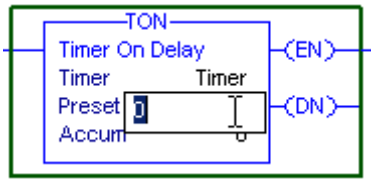
11. In the Name field enter '***Timer***' then click ***Create***



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



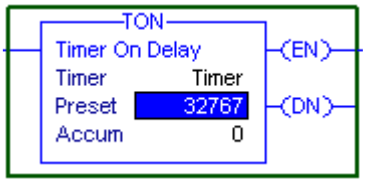
13. Double-click on the ***0*** in the timer instruction next to the word ***Preset***.



14. Enter a value of ***32767***.

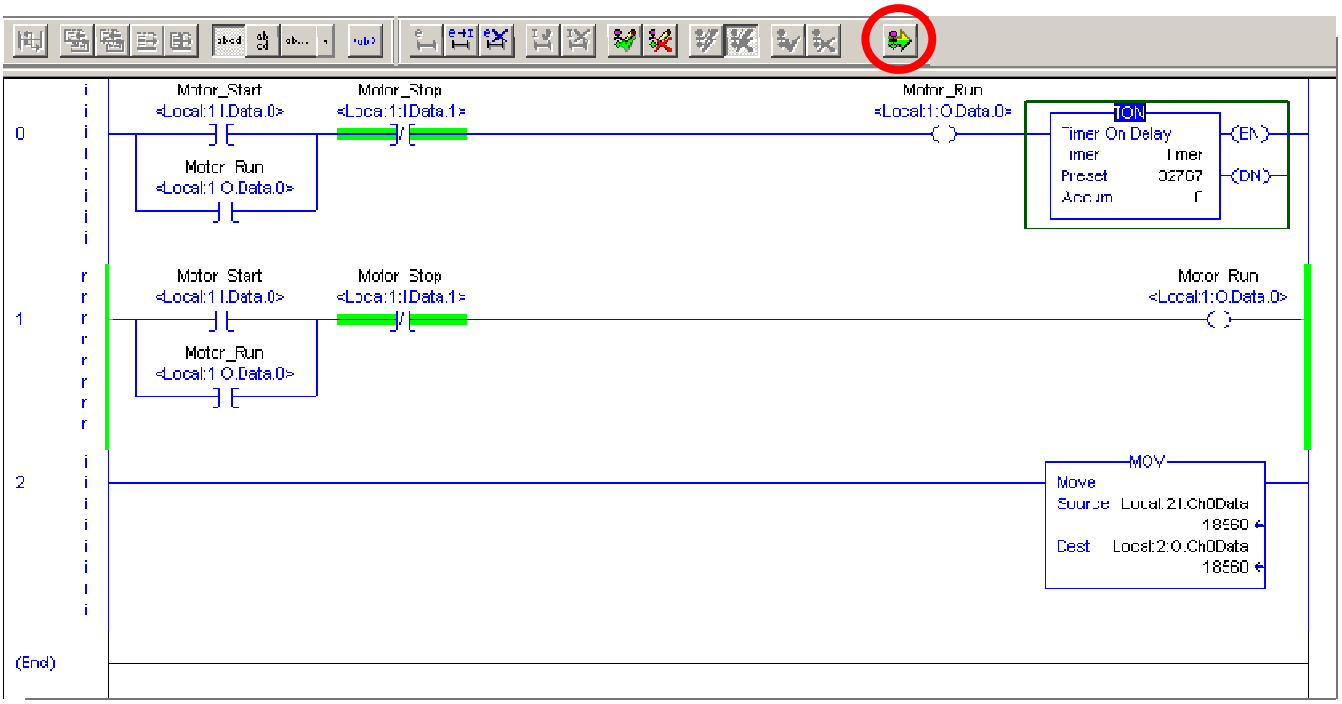
> In Logix the Timer Preset is a 32-bit DINT which means the maximum value for your timers can be: *2,147,483,647*

15. Press ***Enter***. The TON instruction should now appear as shown below.

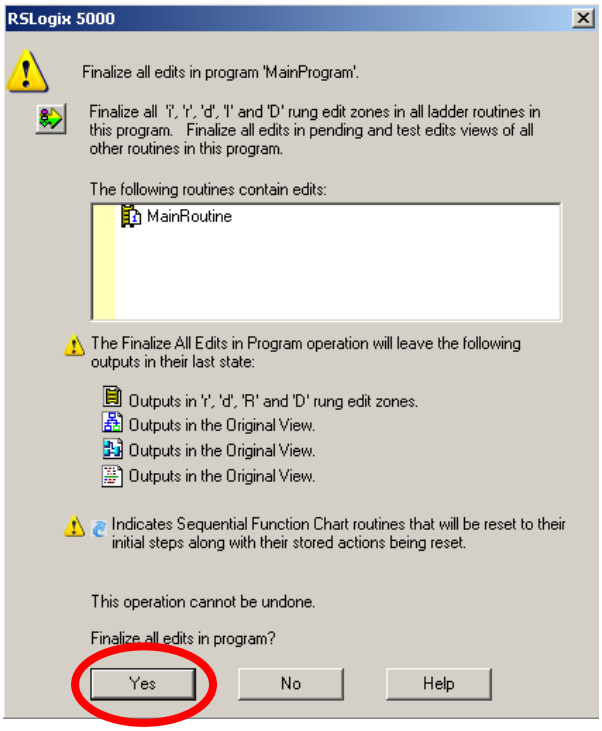


The Preset value is now 32767 milliseconds (= 32.767 seconds). Leave the accumulated value set to zero. You are now ready to verify the edits you made.

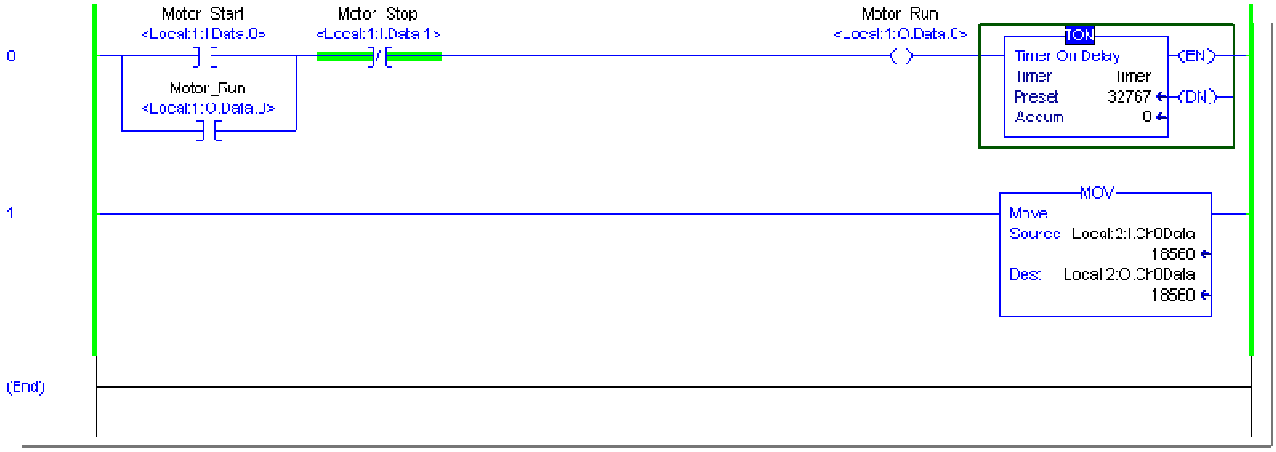
16. Click on the ***Finalize All Edits*** icon



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



The ladder editor will now appear as follows:



Testing Your Logic

18. Press the ***DI0*** (Motor_Start) pushbutton.
19. Verify that ***DO0*** (Motor_Run) illuminates and the Timer instruction starts incrementing.
20. Now, press push button ***DI1*** (Motor_Stop).
21. Verify that ***DO0*** turns off and the Timer resets.
22. Turn the ***AI0*** potentiometer to 5.
23. Verify that the ***AO0*** meter reads 5 Volts.
24. Turn the ***AI0*** potentiometer to MAX.
25. 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:

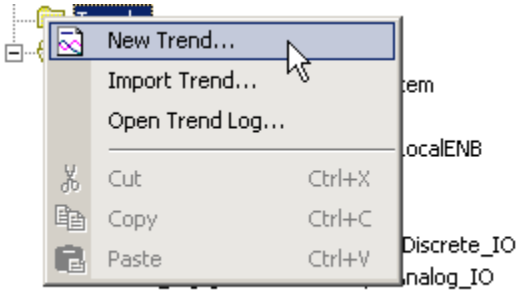
- Create a trend to watch a timer and an input.

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

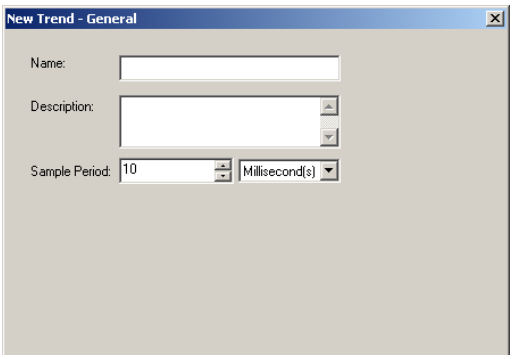
> **Trending**
>
> Basic Trending in Studio 5000 allows you to view data sampled over a time period in a graphical display. Data is sampled at a periodic rate that is configurable from 10 milliseconds to 30 minutes. Studio 5000 will allow you to create a trend and save it as part of your project file.
>
> 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

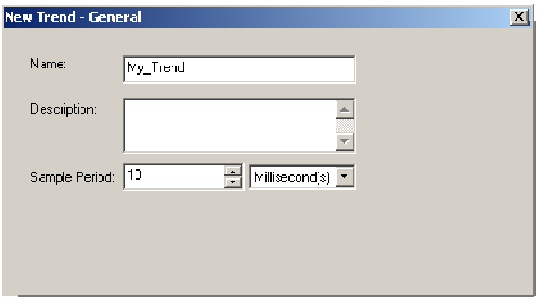
1. From the Controller Organizer, right click on ***Trends*** and select ***New Trend***.



The **New Trend** window appears.

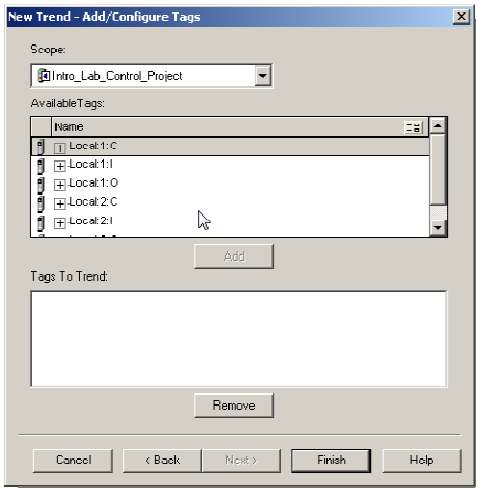


2. In the ***Name*** field enter '***My_Trend***".



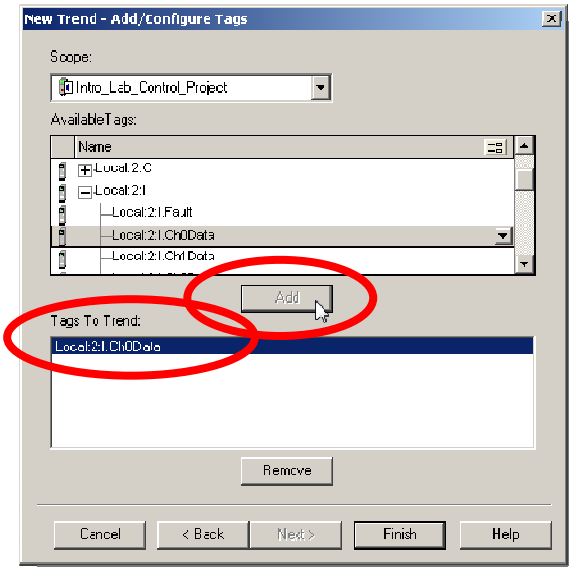
3. Click ***Next***.

   The New Trend Add/Configure Tags window appears.

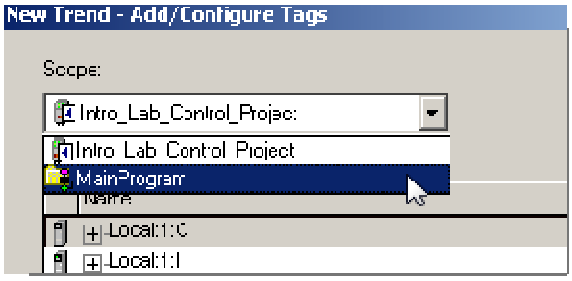


4. Select ***Local:2:I:Ch0Data*** and click Add

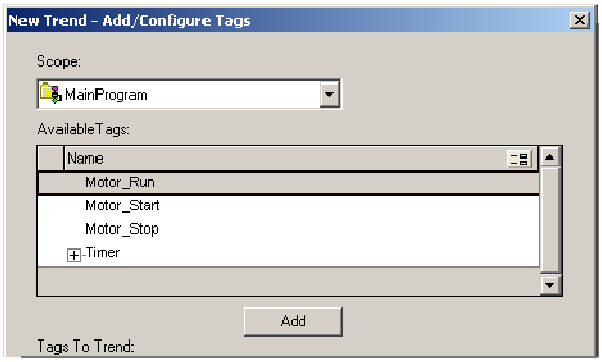
This will allow the trend to monitor the input from AI0



5. Let's also trend the timer accumulator value. The timer the tag was created in the Program Scope, so we must select the ***MainProgram*** tags as shown below:

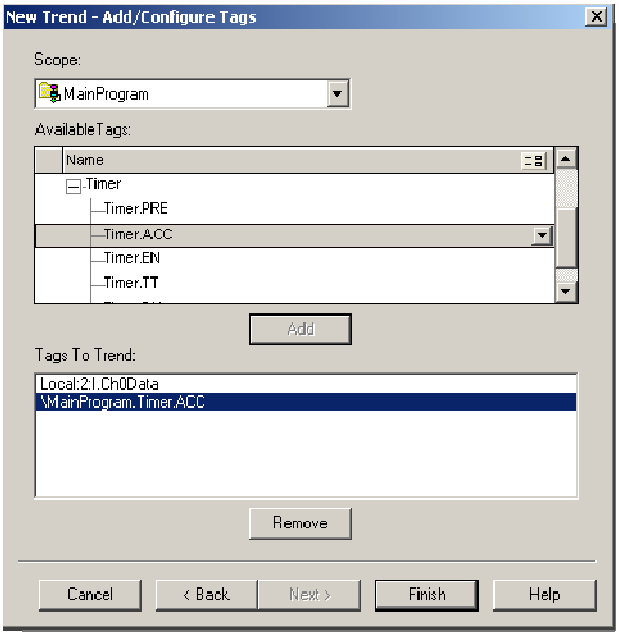


Now <u>only</u> the tags for the ***MainProgram*** are shown.



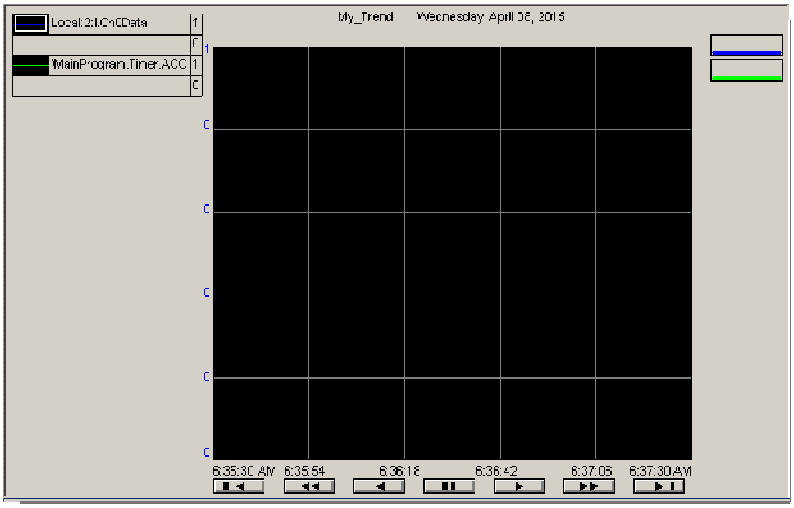
6. Expand the **Timer** tag by clicking on the **+**.

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

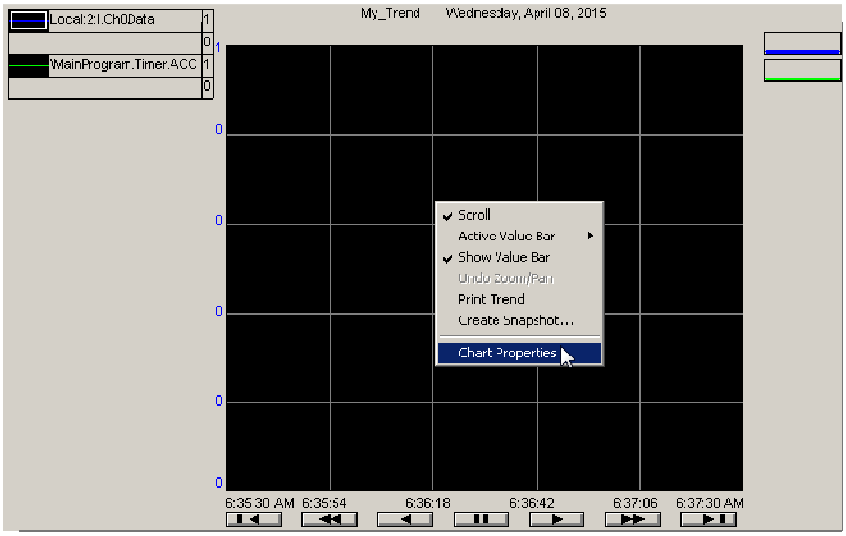


8. Click on ***Finish***.

The Trend window will now appear.



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

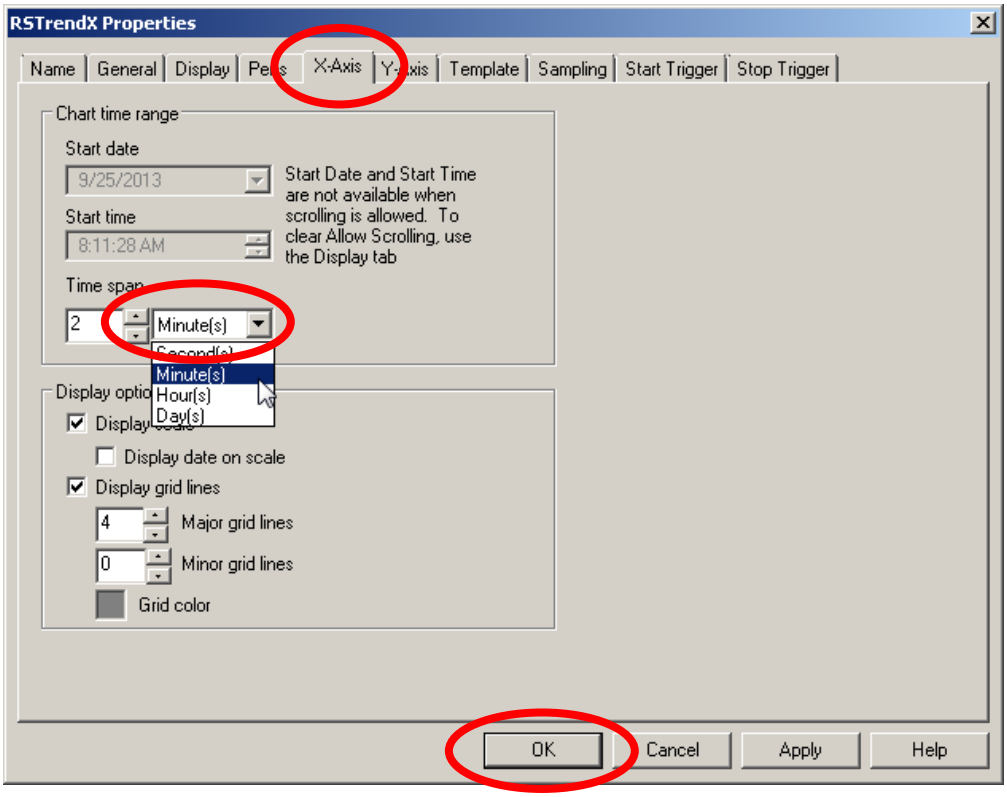


The RSTrendX Properties window will now appear.

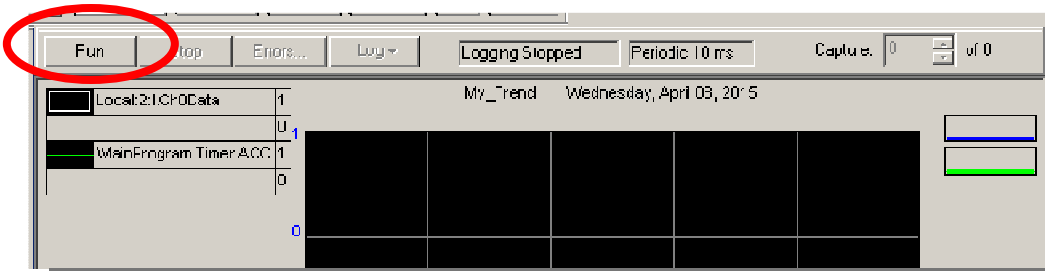
10. Click on the ***X-Axis*** tab.

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

12. Click ***OK.***

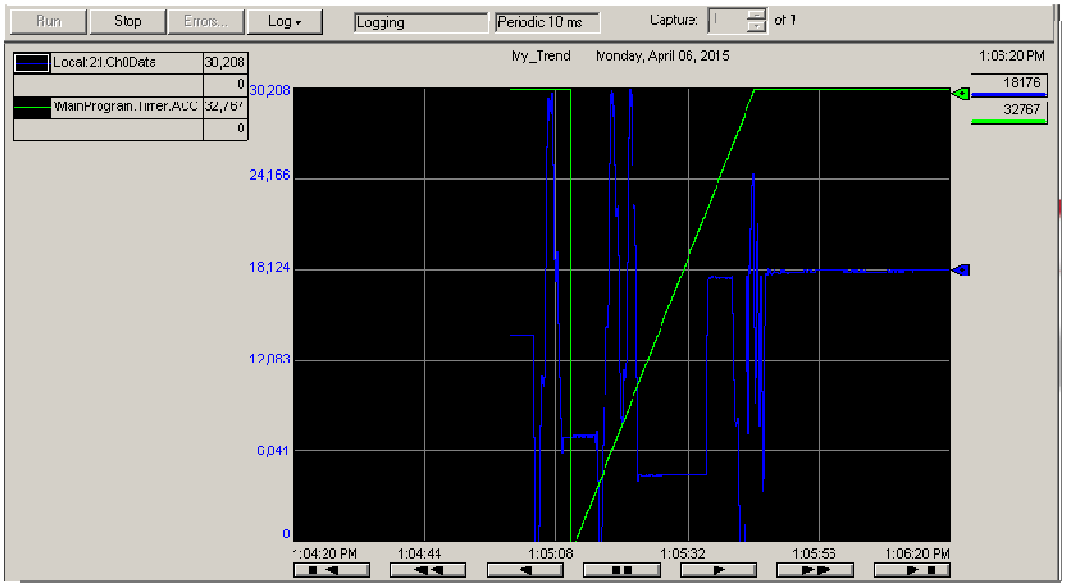


13. Start the trend by clicking on the ***RUN*** button located toward the upper left of the Trend dialog box.



14. Press the ***DI1*** pushbutton then push the ***DI0*** and watch the trend capture the data of the Timer.ACC.

15. Try ***turning the AI0*** input and verify that you see the trend recording the input:



By default, each tag will be independently scaled to its observed min/max values. If desired, the scaling options can be changed under the **chart properties - Y axis tab**.

There are also other options in the trend properties such as a start and stop trigger and pen colors.

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

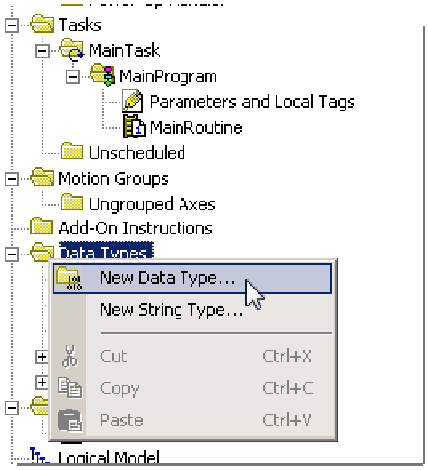
Creating User Defined Types

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

> **What is a UDT and what is it good for?**
>
> A UDT is good for organizing related data into a single structure. A UDT allows a single tag to hold multiple 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.

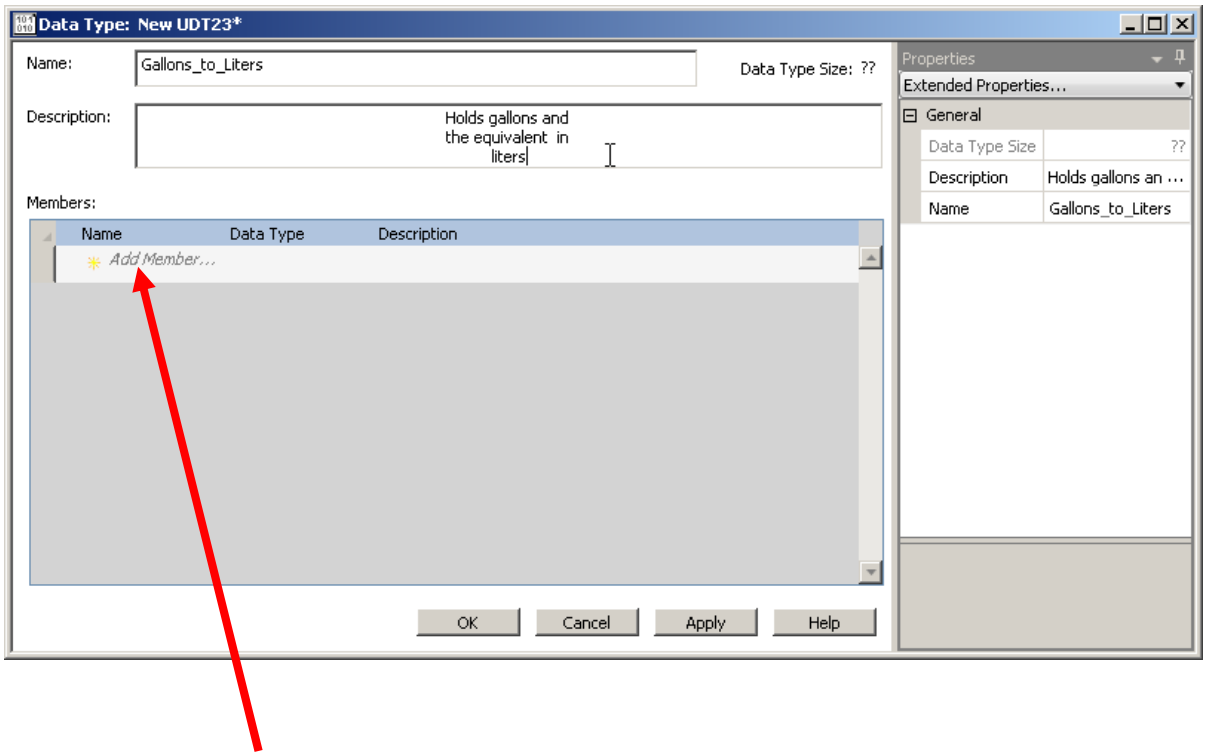
1. *Right click* ***DataTypes*** in the Controller Organizer and select ***New Data Type…..***



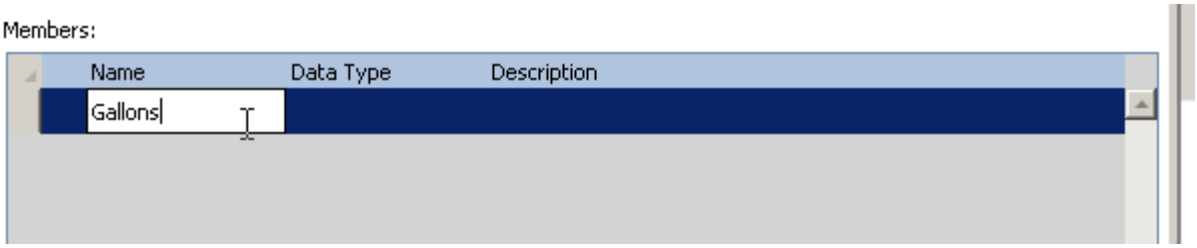
A new **Data Type** window will appear.

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

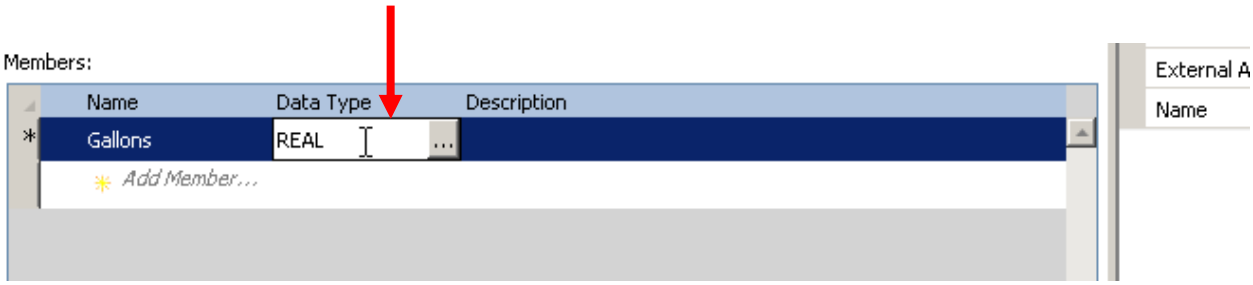
3. Fill in the description field with "***Holds gallons and the equivalent in liters***" as shown.



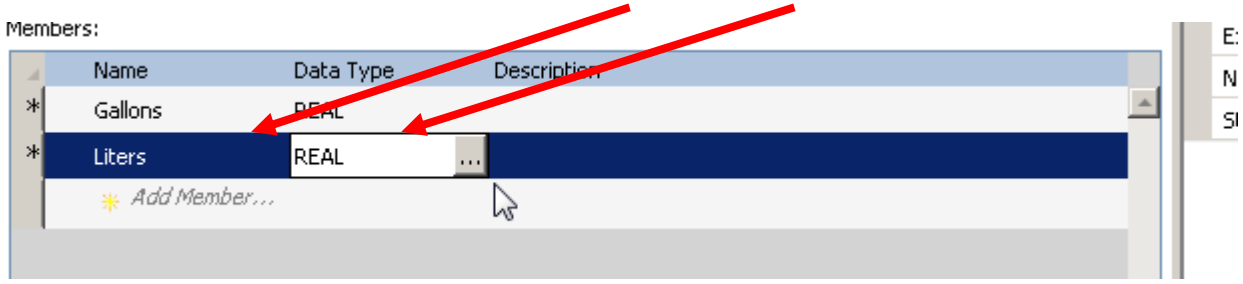
4. Click on ***Add Member…*** and type in '***Gallons'***



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

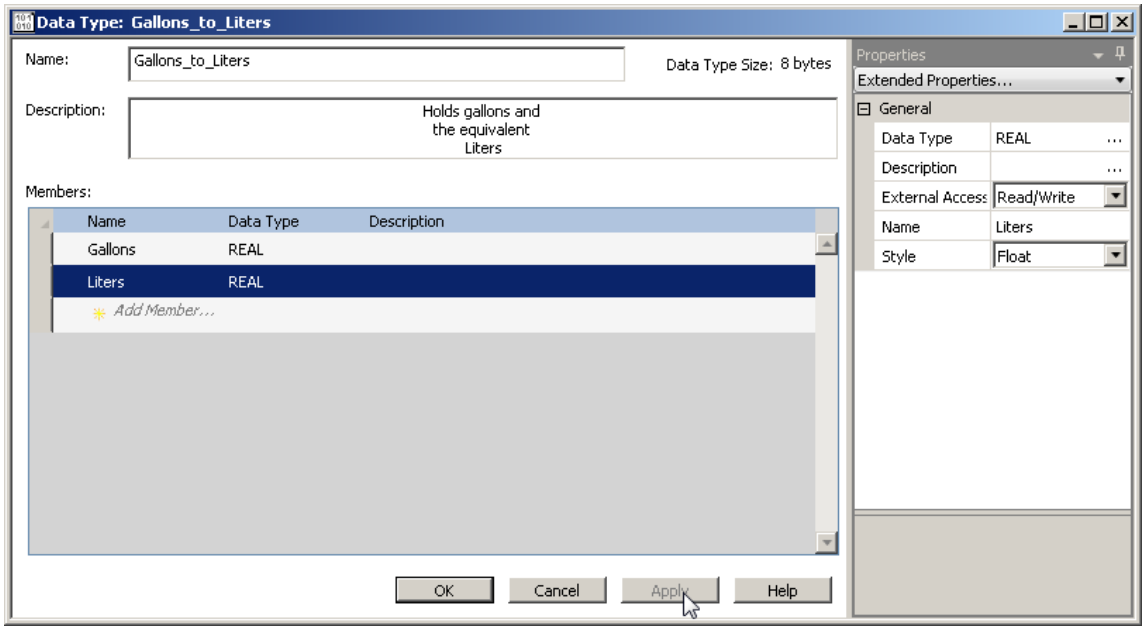


6. Follow the same steps to enter the next row for "***Liters***" and ***REAL*** as shown.



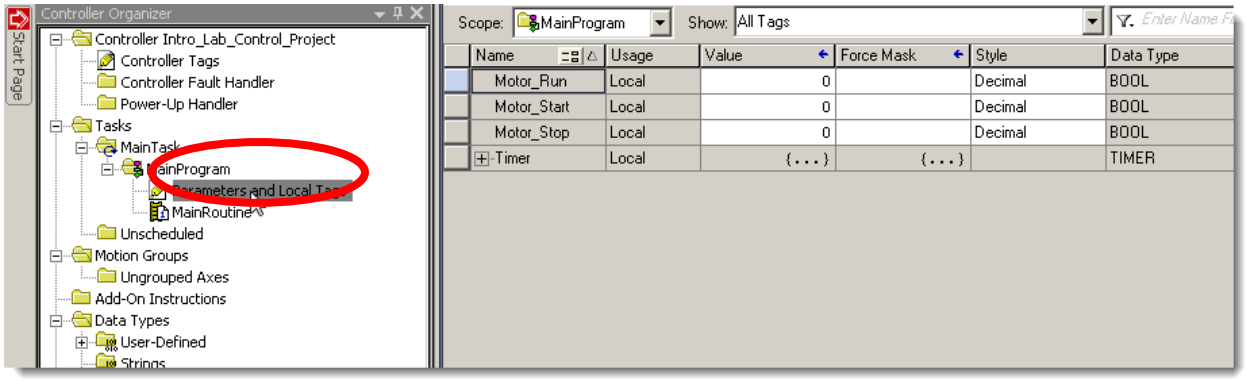
7. Click ***Apply.***

The window should appear as shown.



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

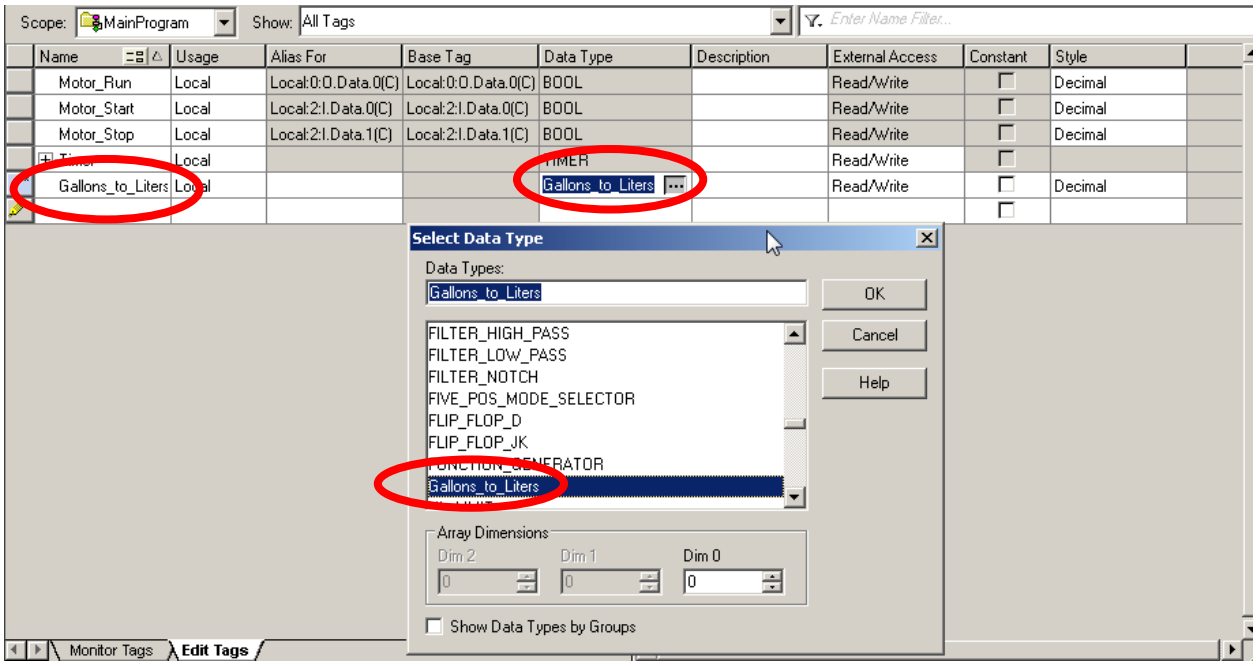
9. Double click on ***Parameters and Local Tags*** under the **MainProgram** as shown to open the tag window.



10. Select ***Edit Tags*** tab on the bottom.

11. On the blank row, type in "***Gallons_to_Liters"*** for the tag name.

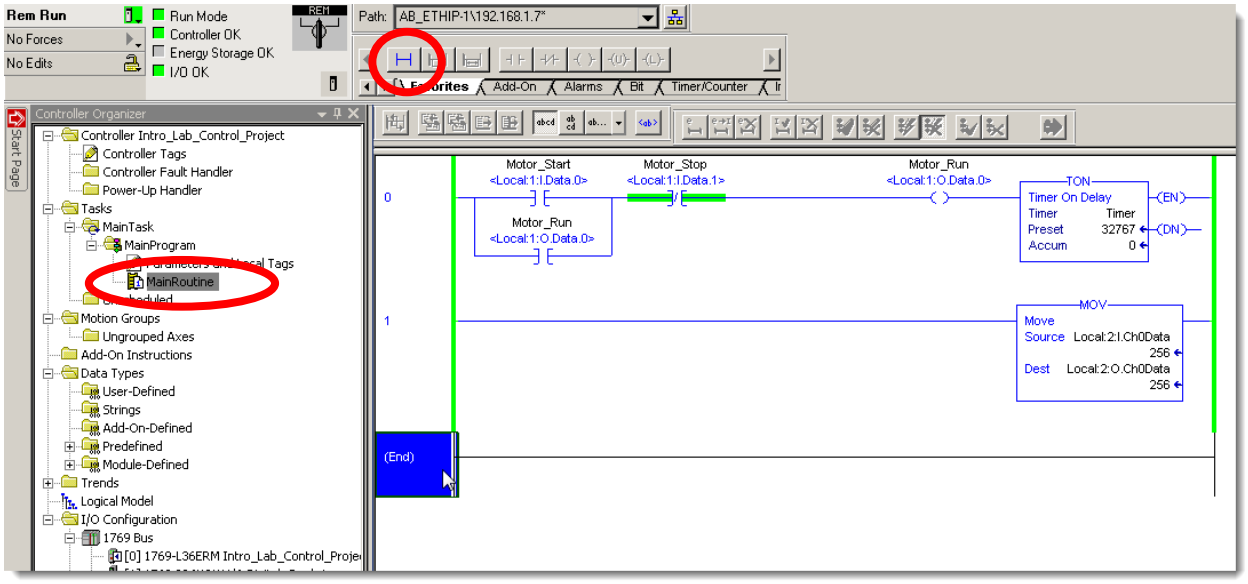
12. On the same row, select "***Gallons_to_Liters"*** for the Data Type as shown and click ***OK.***



13. Click on a ***different row or tag*** to make sure the tag is accepted. The data type column for the Gallons_to_Liters tag will turn grey when it is accepted.

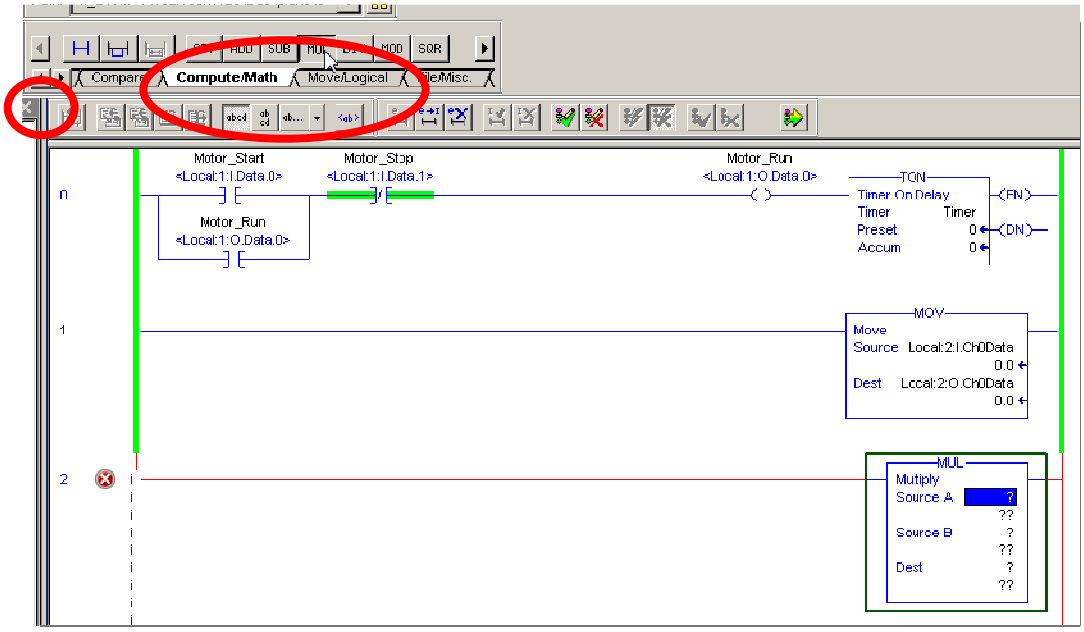
Add the UDT tag to an instruction

14. Double click on the ***MainRoutine***.



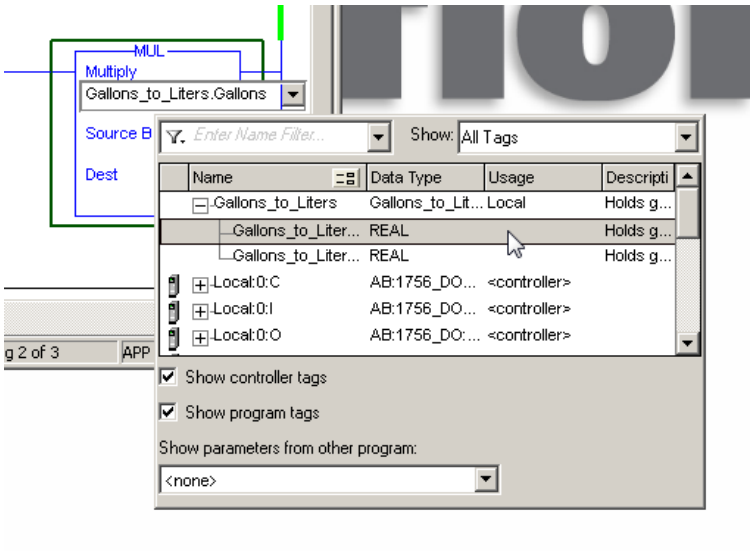
15. Make sure the **End rung** is highlighted. Click on the ***insert rung*** icon to create a new rung.

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



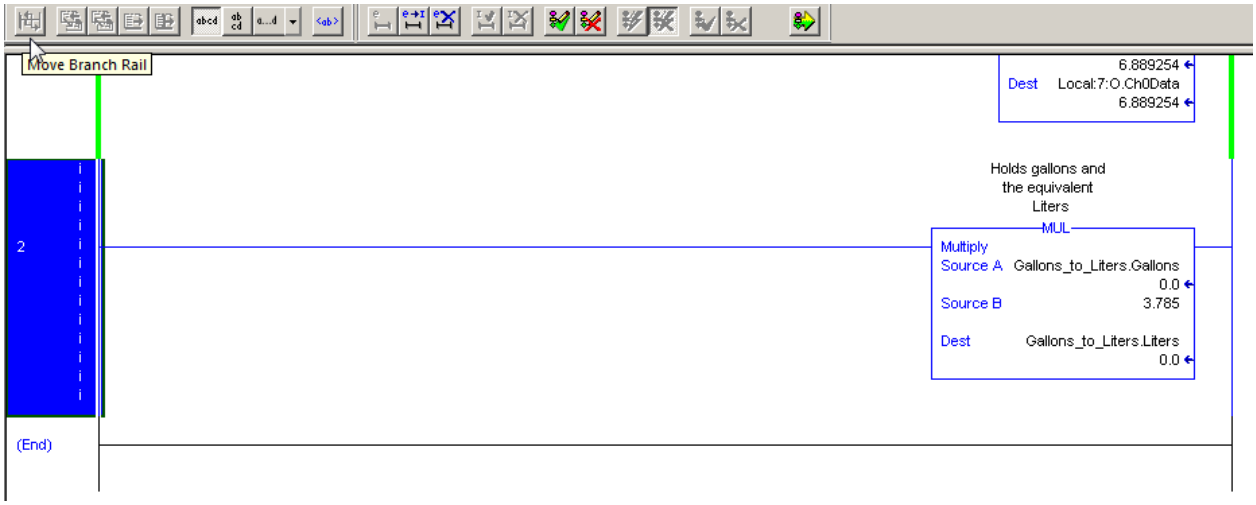
17. Double click on the "***?"*** in the Source A field of the MUL (multiply) instruction and select the ***Gallons_to_Liters.Gallons*** tag.

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



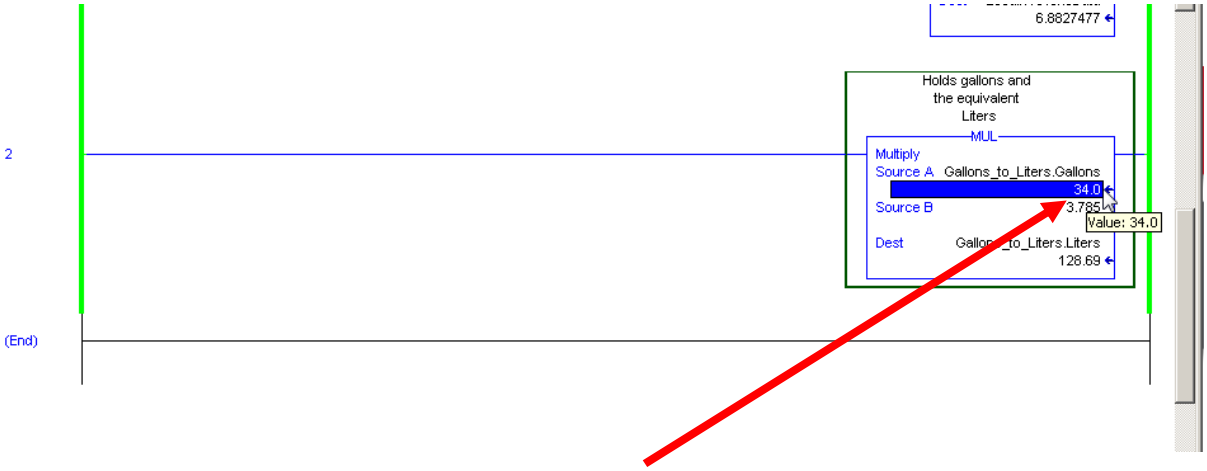
18. Enter '***3.785'*** for source B (the conversion constant to convert gallons to liters).

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



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

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

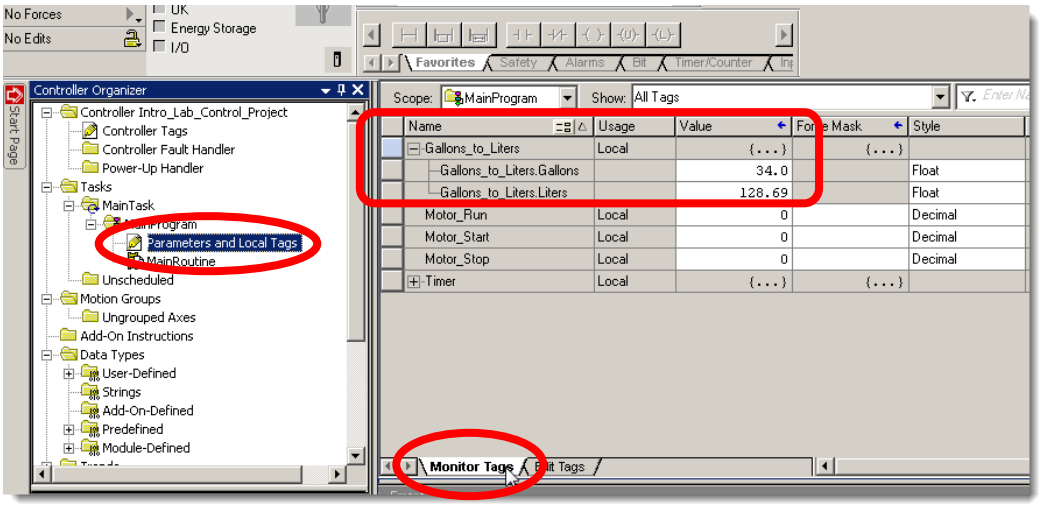


21. Click on the number ***0*** just below gallons, type "***34***" or any desired value, and press enter. Notice that the Liters value updates automatically.

Monitoring UDT Tags

22. 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.

23. 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.



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 Periodic Tasks

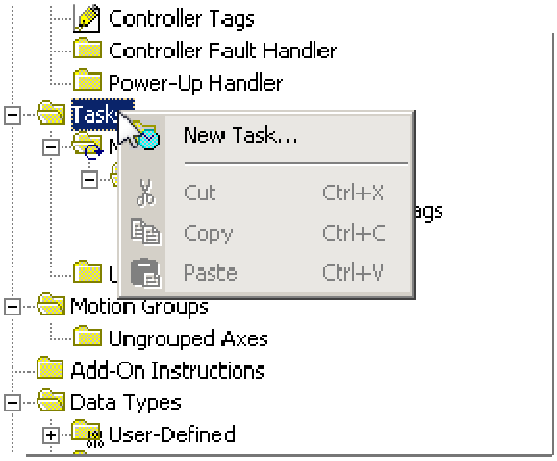
This lab section should take roughly 15 minutes to complete.

Objective:

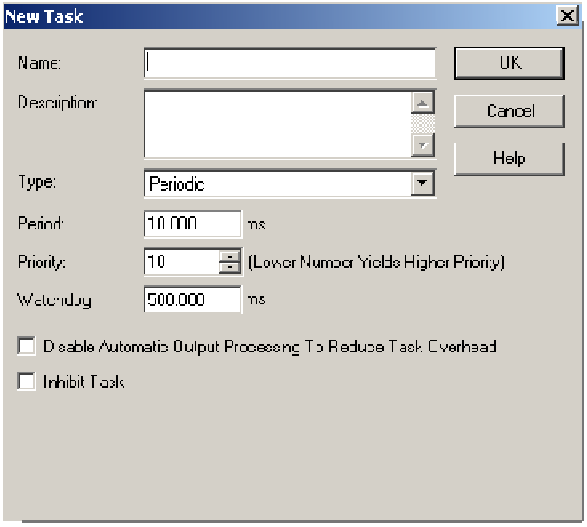
In this lab, we will learn to add and configure a periodic task and add a program and routine with some logic. This will also be done while online with the controller. It will be shown that multiple tasks are running.

Adding a Periodic Task

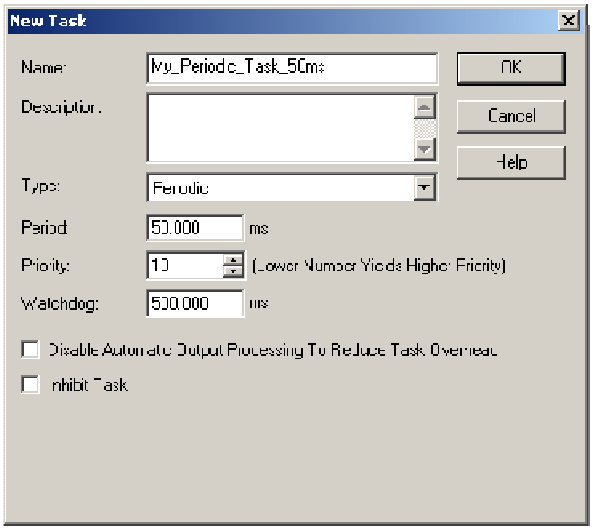
1. In the **Controller Organizer**, right click on ***Tasks*** and select ***New Task***.



The **New Task** window appears. In this window we can configure the properties of a task.

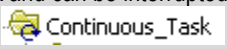


2. Fill in the Window as shown.
   Name – ***'My_Periodic_Task_50ms'***.
   Type - *Periodic.*
   Period – ***'50.000' ms***

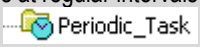


3. Click ***OK.***

Logix controllers have three task types: Continuous, Periodic, and Event.

Continuous – Runs at the lowest priority of any task and can be interrupted by other tasks. The continuous task is designated by a folder with a circular arrow. 

There can be a maximum of 1 continuous task.

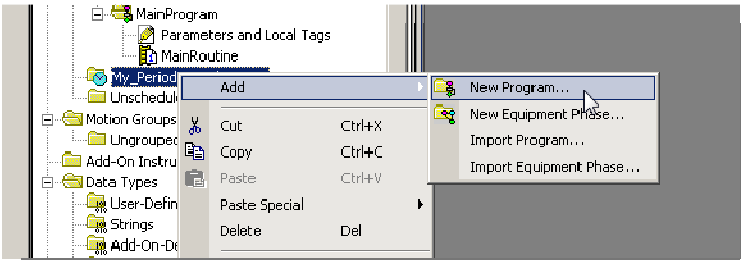
Periodic – Executes at regular intervals and can be assigned different priorities. It is designated by a circular blue clock symbol. 

There can be multiple periodic tasks.

Event – Triggers on specific events. 

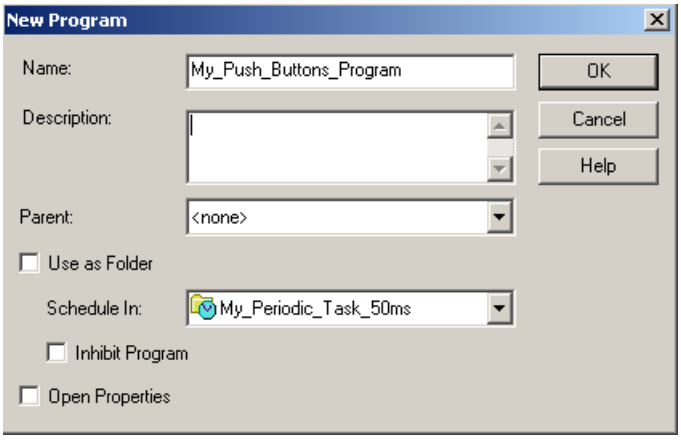
This allows code to execute as quickly as possible when some event happens. For example, a counting instruction can quickly be executed whenever a photeye turns on to get an accurate count of parts.

4. Right click on the ***My_Periodic_Task_50ms*** folder, select ***Add*** and then select ***New Program….***



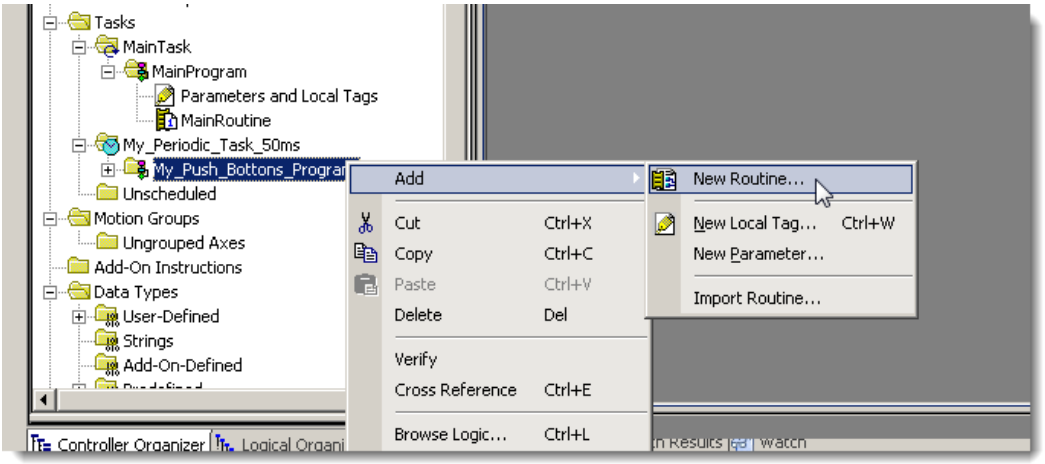
Tasks are divided into programs. Whenever a new task is added, a new program also needs to be created. Multiple programs are allowed in each task, and the programs execute in order one at a time whenever the task executes. Programs allow the code in a task to be visually organized in large applications.

5. Fill in the name '***My_Push_Buttons_Program'*** and click ***OK***.



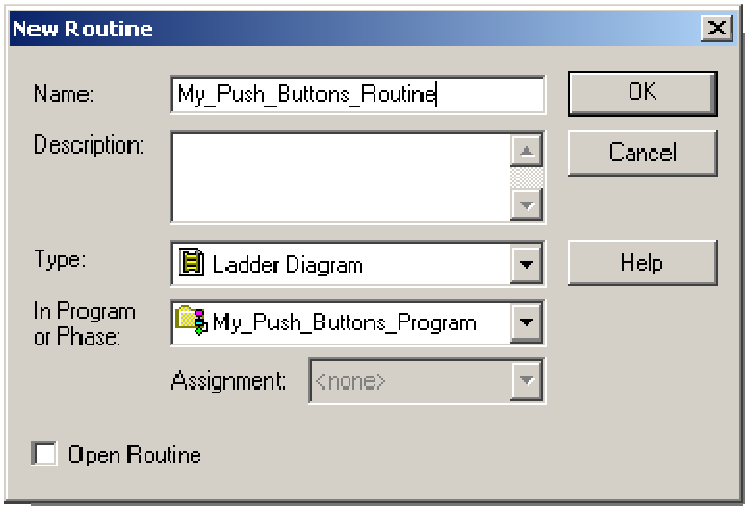
6. Expand the ***My_Periodic_Task_50ms*** folder.

7. Right click on the ***My_Push_Buttons_Program*** folder and select ***Add*** and ***New Routine.***



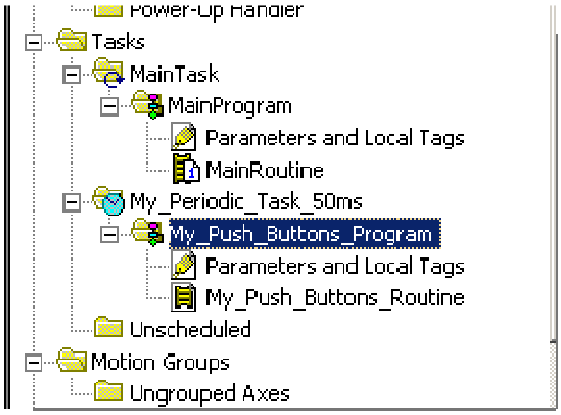
Routines are where the code resides. Different kinds of language routines can be created. The Logix platform supports Ladder, Function Block Diagram, Structured Text, and Sequential Function Chart as the types of routines that can be created.

8. Fill in the window as shown.
   Name – '***My_Push_Buttons_Routine'***
   Description – ***'<Any desired description.>'***
   Type – ***Ladder Diagram***
   In Program or Phase –***My_Push_Buttons_Program***.
   Assignment – ***None***.

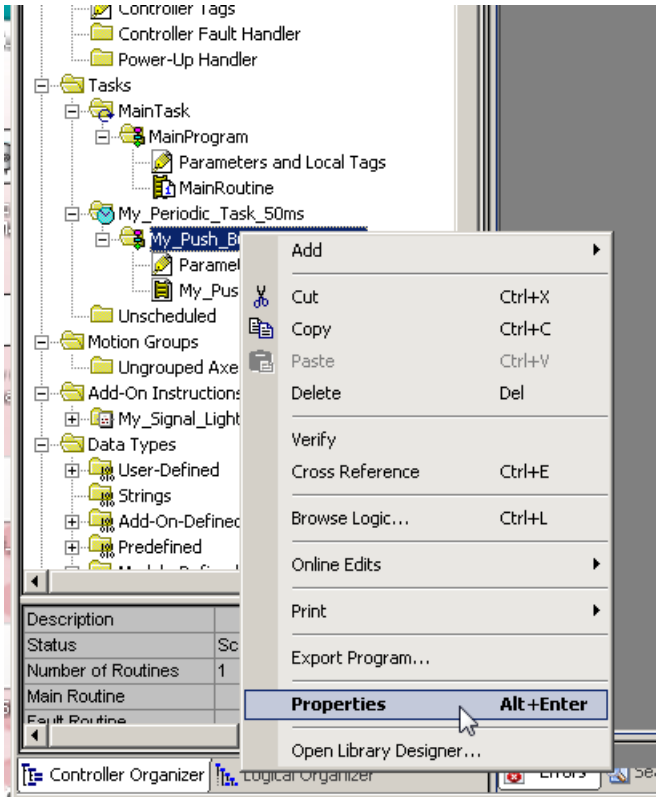


9. Click ***OK.***

The controller organizer should look like the following.



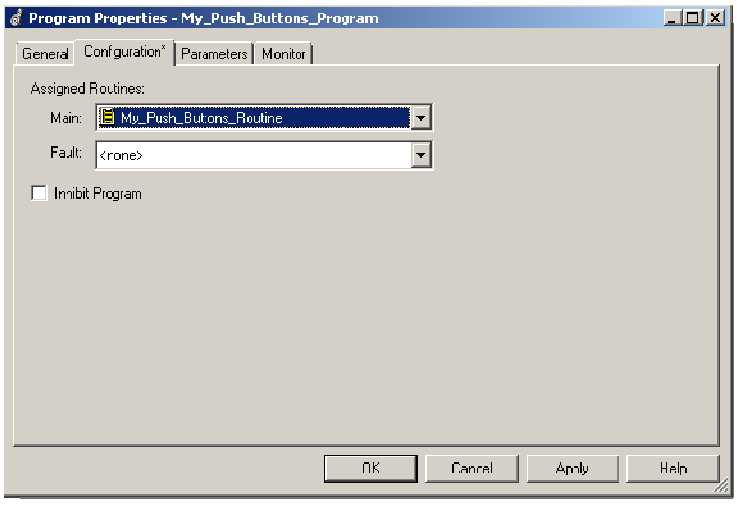
10. Right click on ***My_Push_Buttons_Program*** folder and select ***Properties***.



11. In the **Program Properties** window select the ***Configuration Tab.***

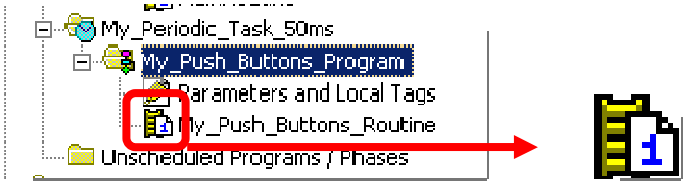
12. For **Main**, select ***My_Push_Buttons_Routine*** as shown.

We have just defined a main routine that will automatically be called.

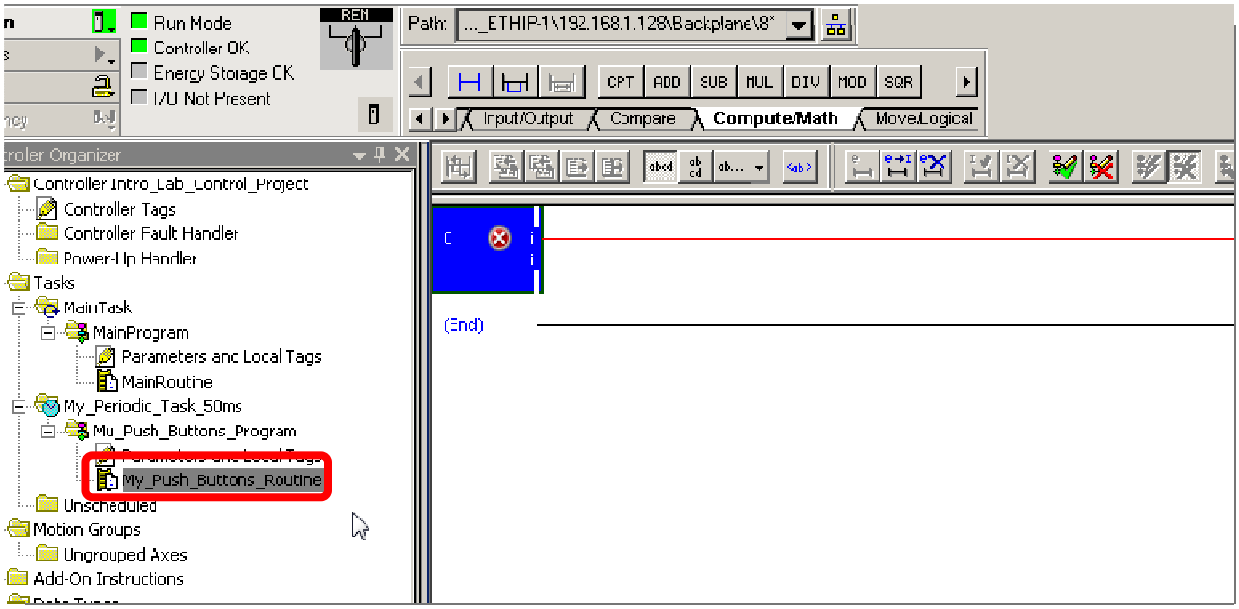


13. Click ***OK.***

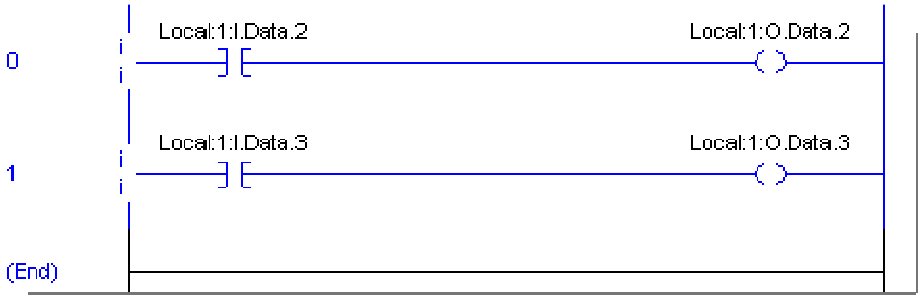
The controller organizer should look like the following with an "I" indicating the main routine. There is only one main routine for each program. Only the main routine runs by default. If no routine is select as the main routine, then no routines will execute. JSR (Jump To Subroutine) instructions are used to call other routines.



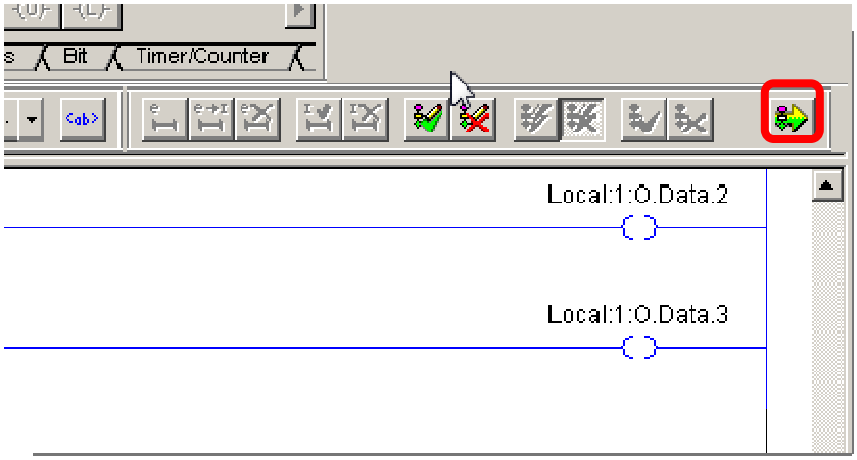
14. Double click on *the* ***My_Push_Buttons_Routine*** to open the routine. It should look like the following.



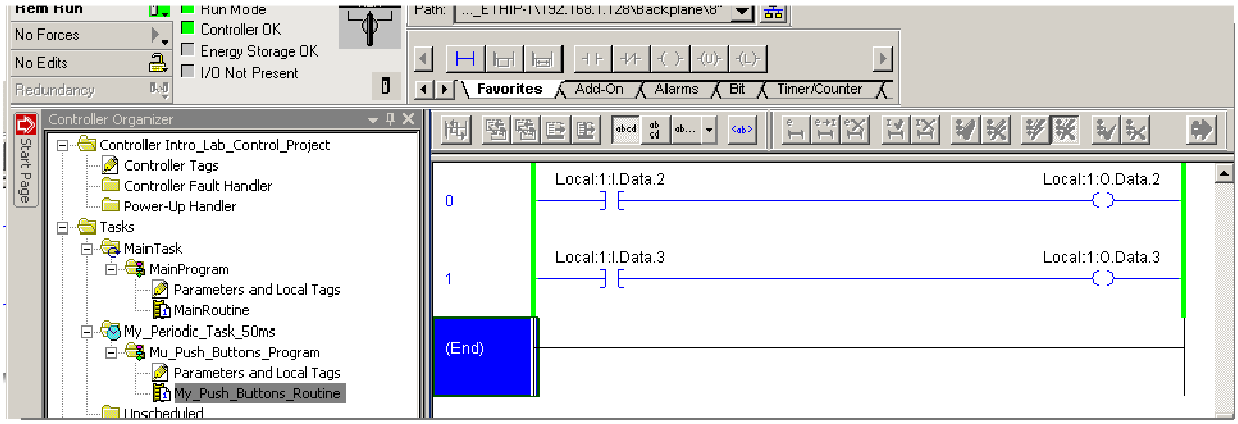
15. ***Create the following two rungs*** using the lessons previously learned. Use input point 2 to drive output 2 and input point 3 to drive output 3 as shown.



16. Click on the ***Finalize All Edits in Program button*** -> ***Yes*** to accept the changes.



The screen should now look like the following.



17. Try pushing the ***DI2*** and ***DI3*** buttons. They should light up while pressed.

Notice that the motor start and stop buttons still work as they did before. This demonstrates that both programs are running! The controller is running two tasks, one as “continuous”, and another as a 50ms periodic task.

Note: Notice the ability to make many kinds of changes while the controller is running and controlling.

We did these changes while online with a ‘live’ controller that was currently controlling our ‘machine’.

Congratulations! You have Completed Section 9. Please move on to Section 10.

Section 10: (Optional) Creating an AOI (Add On Instruction)

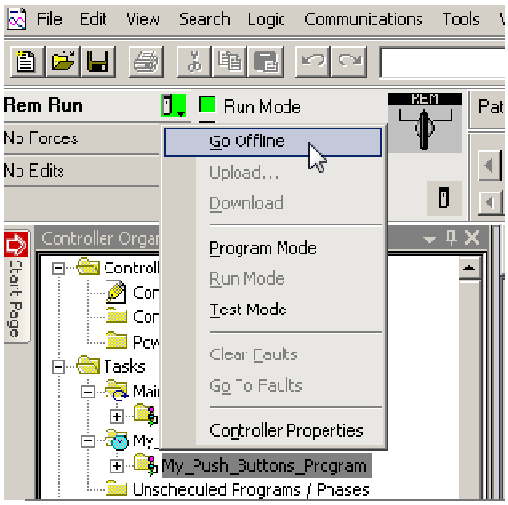
This lab section should take roughly 15 minutes to complete.

Objective:

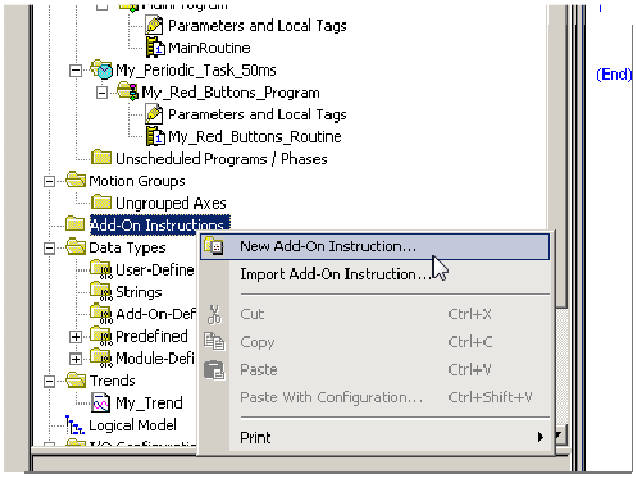
In this lab, an AOI instruction will be added to the project. AOI's cannot be created or modified online (although they can be imported online). The AOI will be created offline, code added, and downloaded to the controller to see it execute.

Adding an AOI instruction

1. In the **Logix Designer window**, click the ***save icon -> yes*** to save the project and tag values, then ***Go Offline***.

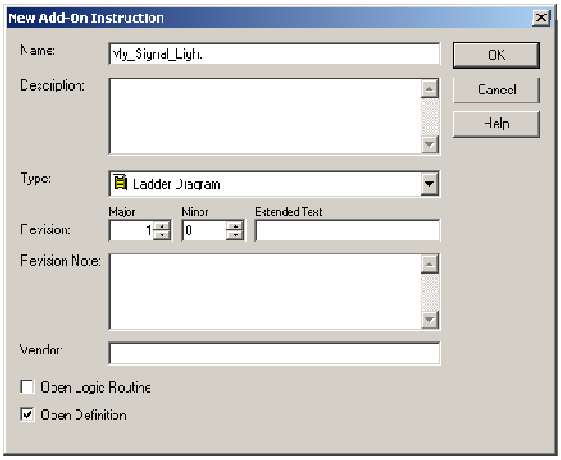


2. In the **Controller Organizer**, right click on ***Add-On Instructions*** folder and select ***New Add-On Instruction***.



The **New Add-On Instruction** window appears.

3. Enter the name '***My_Signal_Light'.***

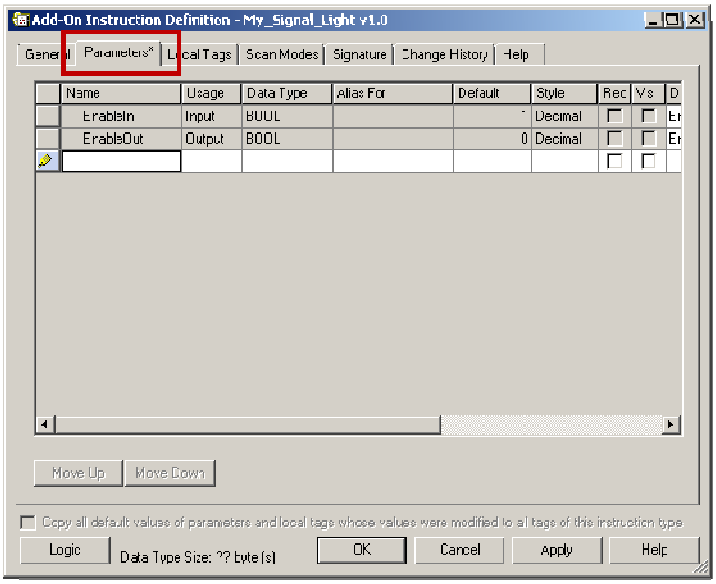


4. Click ***OK.***

The Add-On Instruction Definition window appears. In this window we can configure the properties of the AOI.

5. Click on the ***Parameters tab***.

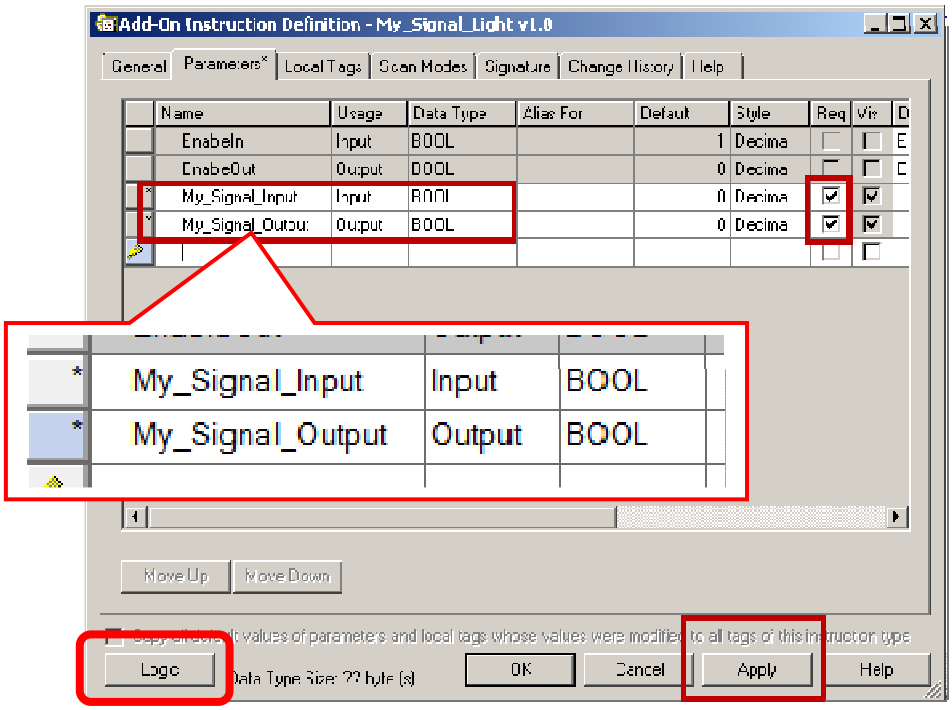
> The Parameters tab is where the inputs and outputs of the instruction are defined.
> The Local Tags tab contain tags that are only used by the AOI for internal storage.



6. Fill in the following two parameters as shown below.
   My_Signal_Input -- Input -- BOOL -- check Required (Req).
   My_Signal_Output -- Output -- BOOL -- check Required (Req).

> The required checkbox indicates a tag will need to be filled in on the instruction.
>
> The usage INPUT means the AOI will operate on a copy of the tags data. OUTPUT means the AOI will copy the result to the tag. There is also an INOUT usage, this passes tag by reference, and the tag is read from and written to directly while the AOI executes.

Verify the window is as follows

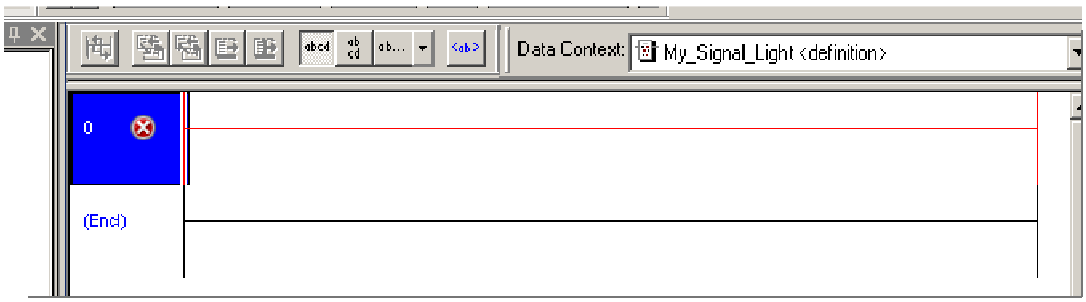


NOTE: Make sure to change the usage on **My_Signal_Output** to *"**Output**"*

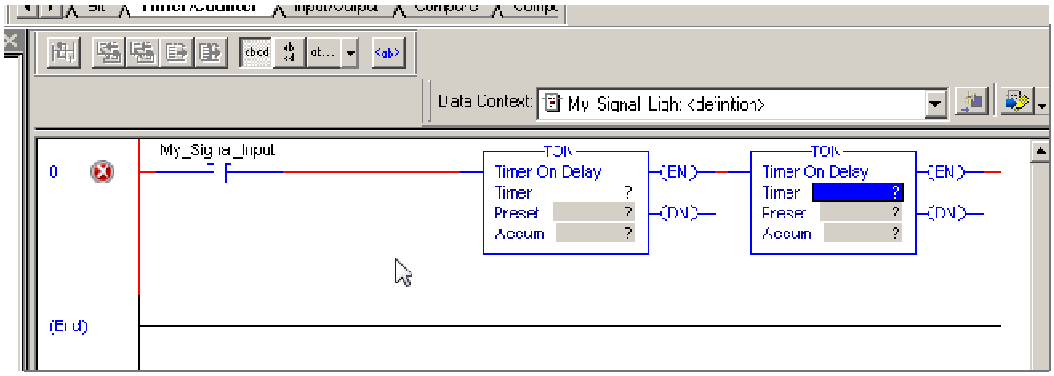
7. Click ***Apply.***

8. Click on the ***Logic Button.***

The following Logic window appears.

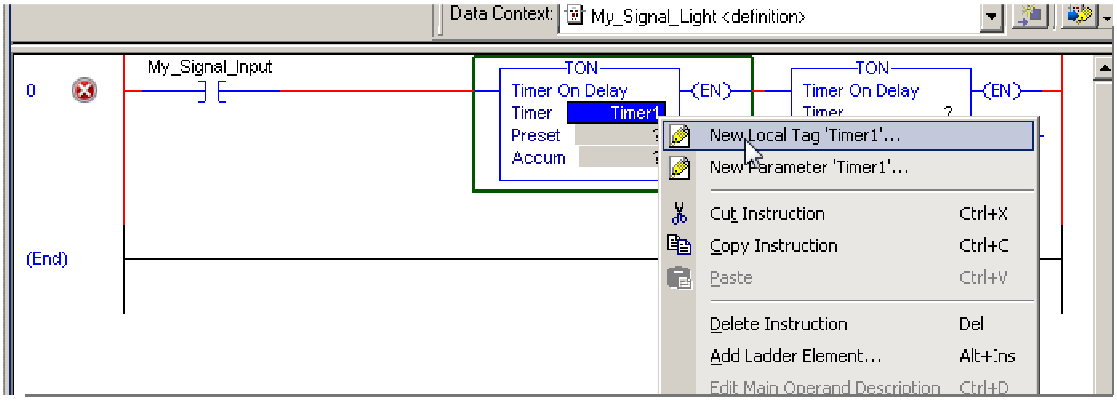


9. Add an ***XIC*** instruction with the tag ***My_Signal_Input*** and ***add two timers*** to a rung. The rung should appear as follows.

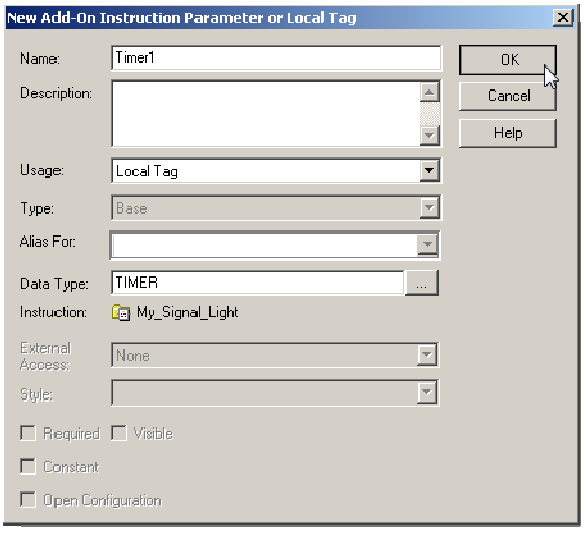


10. Enter the tag name '***Timer1'*** for the first timer.

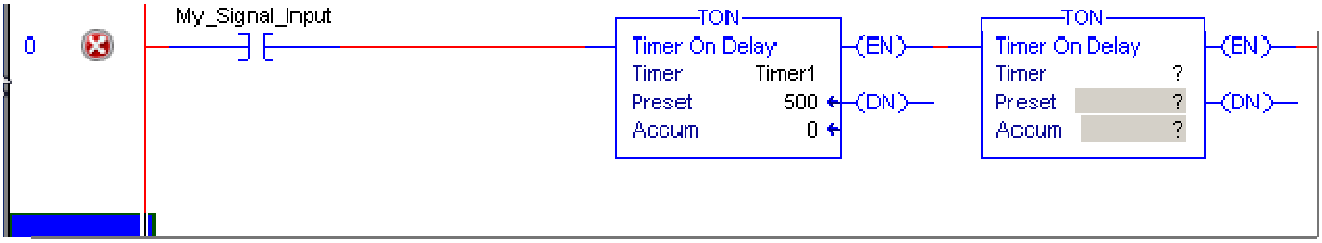
11. Right click on ***Timer1*** and select **New Local Tag '*Timer1'*** as shown



12. Click ***OK*** on the New Add-On Instruction Parameter or Local Tag window that appears.

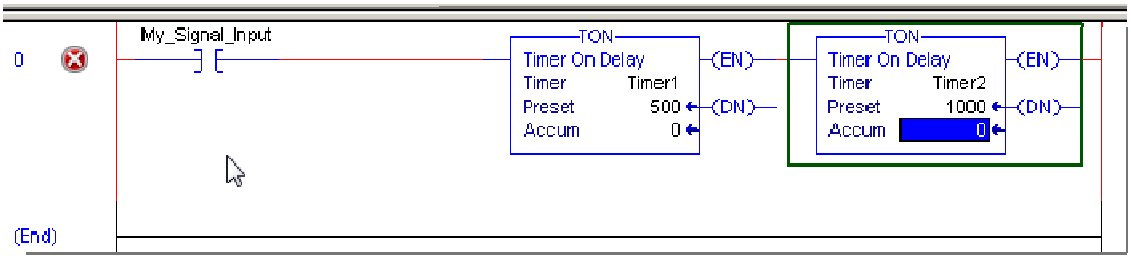


13. For **Timer1**, set the **Preset** to ***500*** and the **Accum** to ***0*** as shown

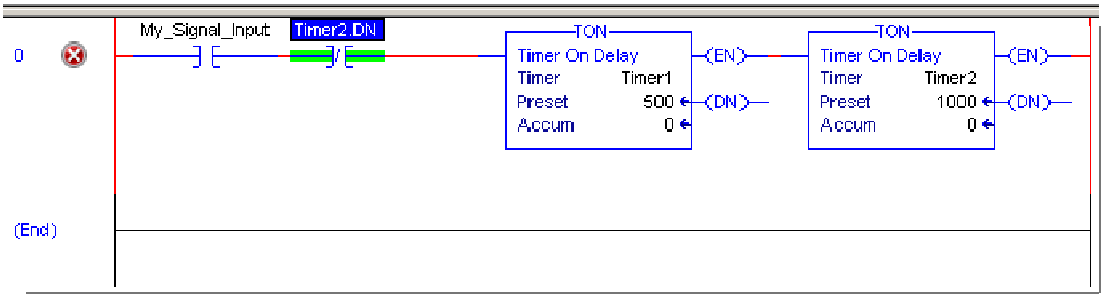


14. Now add ***Timer2*** to the remaining **TON** instruction and ***create it as a local tag*** as well.

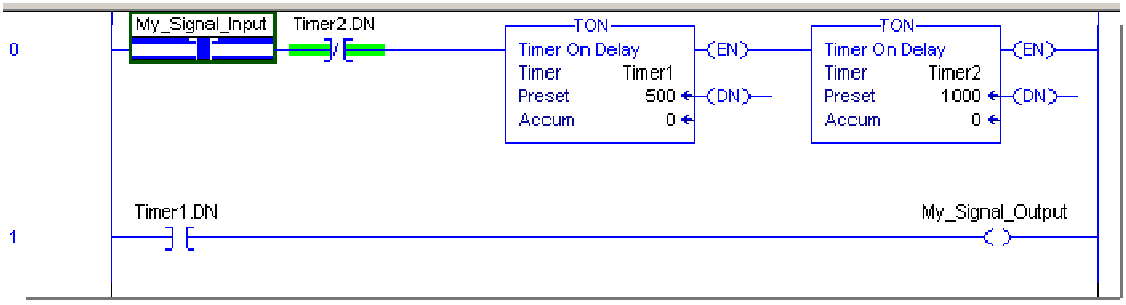
15. Set **Timer2 Preset** to ***1000*** and the **Accum** to ***0*** as shown.



16. Add an ***XIO*** instruction of ***Timer2.DN*** as shown after the **XIC My_Signal_Input** instruction.



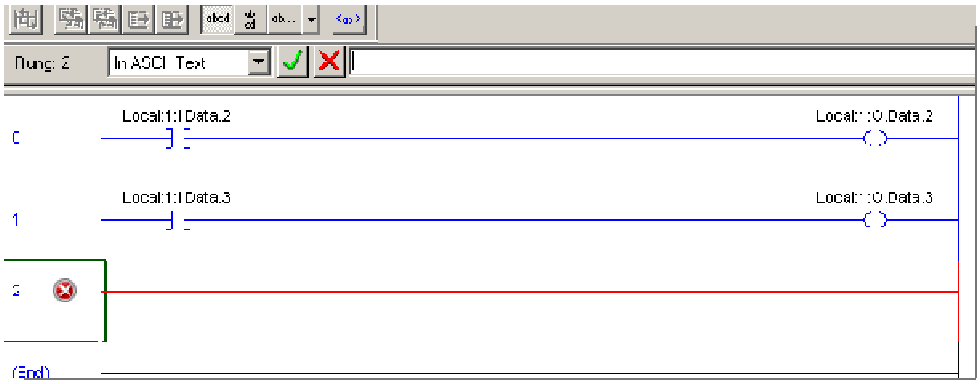
17. ***Add a second rung*** with an ***XIC*** of ***Timer1.DN*** and an ***OTE*** instruction of ***My_Signal_Output*** as shown.



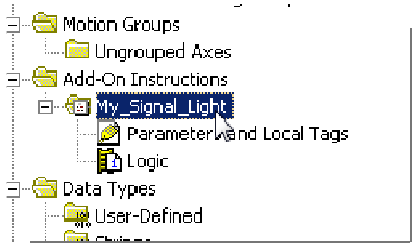
We have finished creating the AOI definition and logic! We will now add the AOI into the program to have it flash a light.

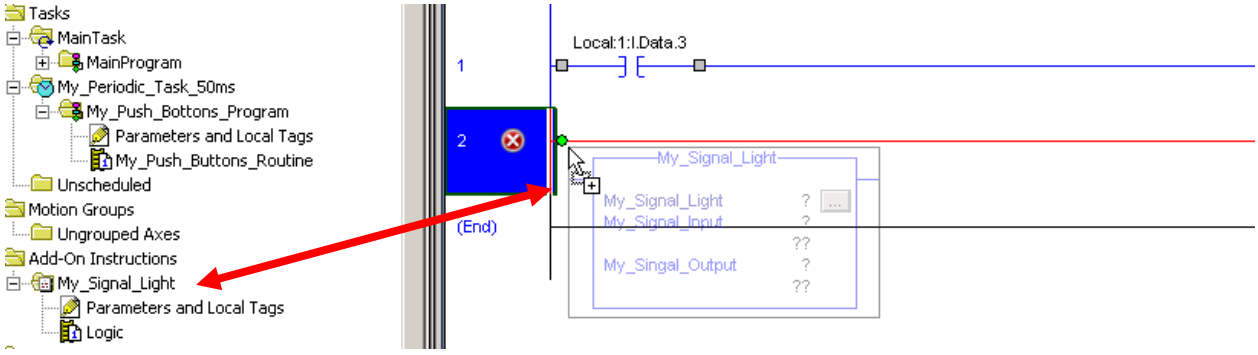
AOI's allow logic to be embedded into a single instruction. This allows new custom instructions to be tailored to the specific application. This also allows code to be easily reused between applications since AOI's can be imported and exported.

18. Open the ***My_Push_Buttons_Routine*** and ***add another rung*** as shown.



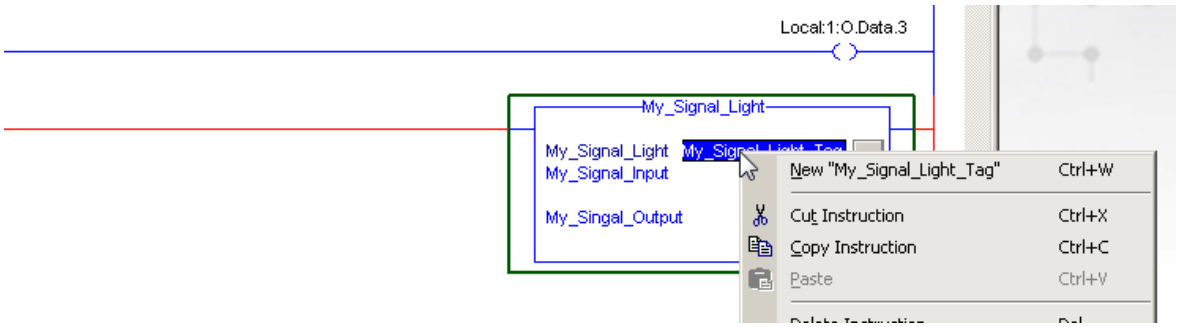
19. Click and drag the ***My_Signal_Light Instruction*** folder from the tree to the rung as indicated by the next two pictures.



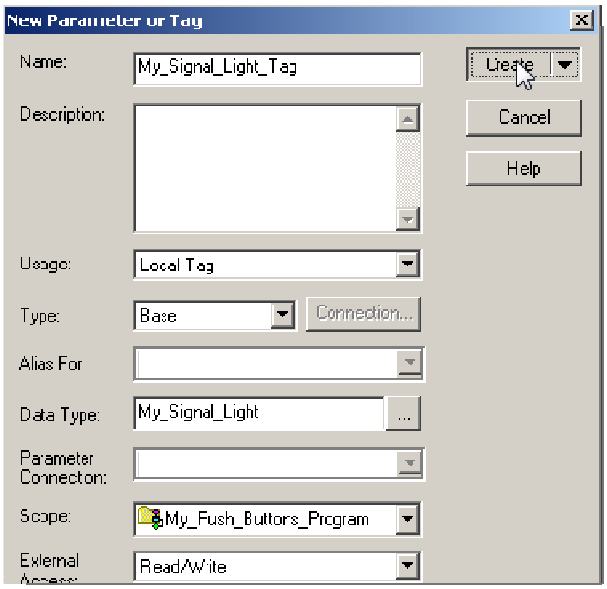


20. Enter '***My_Signal_Light_Tag'*** into the My_Signal_Light field.

21. Right click on the ***My_Signal_Light_Tag*** and select ***New "My_Signal_Light_Tag"***



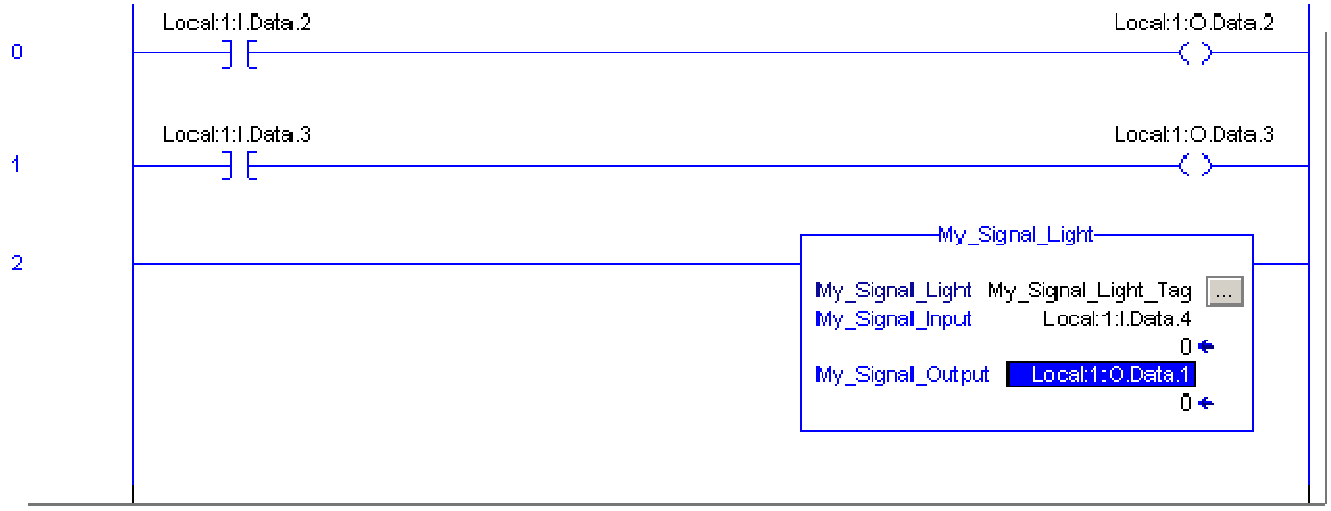
22. Click ***Create*** to create a new local tag.



The local tag being created is called the AOI's backing tag. This is the tag the AOI uses to store its status and local tag values.

23. Add ***Local:1:I.Data.4*** as the **input tag**, and ***Local:1:O.Data.1*** as the **output tag**

The ladder logic should now look as follows.



24. ***Save*** the program and ***download*** it to the controller.

25. Toggle the ***DI4*** input switch and watch what happens to the **DO1** light.

Every time the **DI4 input switch** is on (turned right), the **DO1 light** blinks at a 500 millisecond rate**.**

Notice the remaining buttons still work as they did before!

AOI's allow code to be encapsulated into a single instruction. This allows common code and functionality to be clearly defined and easily reused. The AOI can be reused as many times as desired. Each AOI should typically have a unique backing tag.

Congratulations! You have Completed Section 10. Please move on to Section 11.

Section 11: (Optional) Using Logical Organizer

This lab section should take roughly 10 minutes to complete.

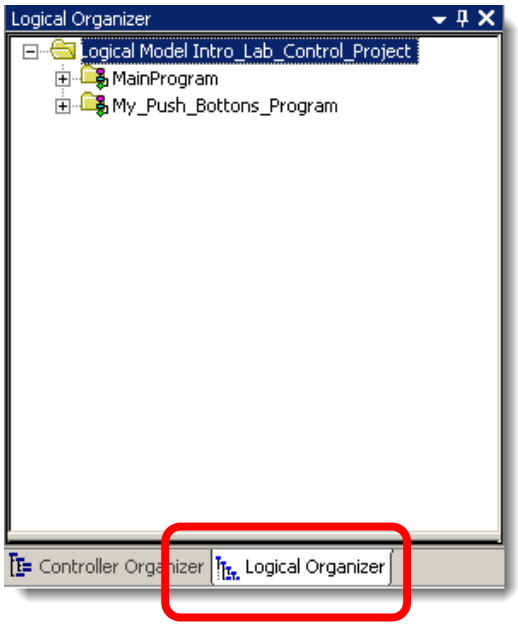
Objective:

The Logical Organizer will be used in this lab to group the code to model the demo box. This will demonstrate how the Logical Organizer can be used to group code regardless of the layout of the Controller Organizer. In this case, we will model it after our machine, the demo box. The Logical Organizer allows programs to be grouped in any manner desired, but grouping them by machine physical or logical function is typical.

Using the Logical Organizer

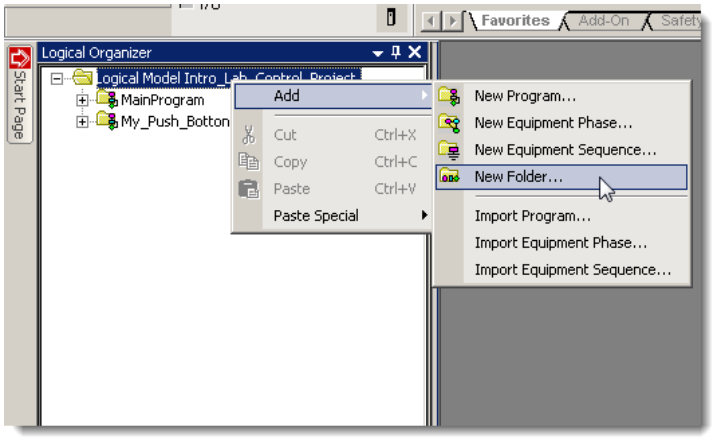
1. Click on the ***Logical Organizer tab***

   By default, all of the programs in the Controller Organizer are shown as an ungrouped list. Only the programs, and not the tasks, are shown. We can use the organizer to group the programs. We will typically use a "folder" object to group the programs.

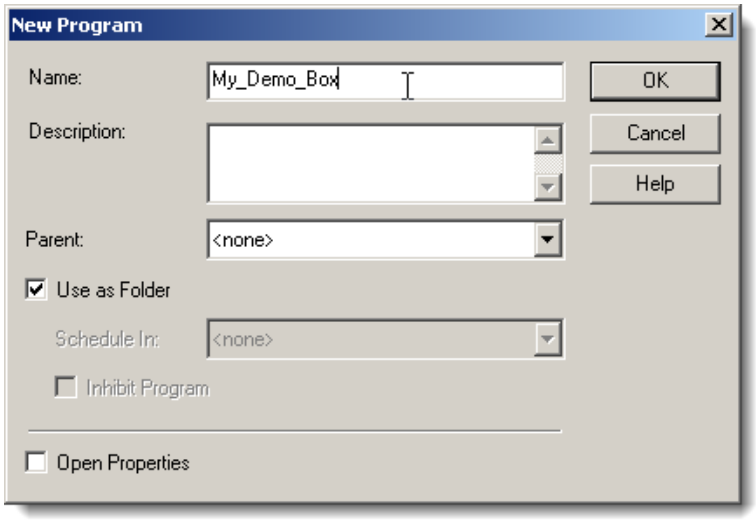


2. To add a new folder, right click on the ***Logical Model folder -> Add -> New Folder***

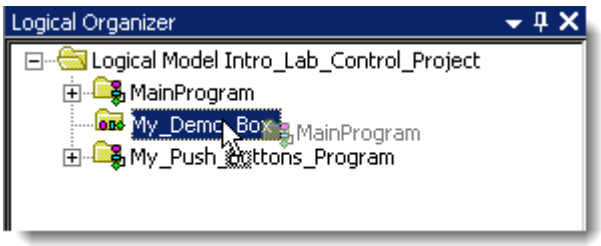
   Notice that we can also add programs here as well as in the Controller Organizer. If we add a program here, the configure window will allow us to pick which Controller organizer task the program is scheduled in.



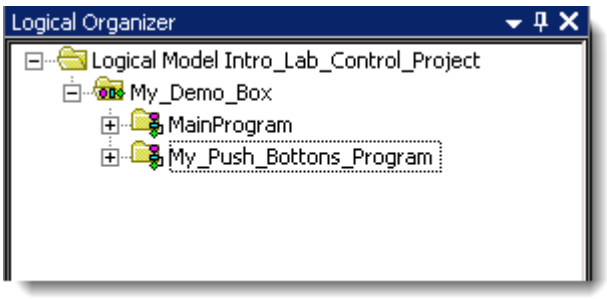
3. Enter the name ***'My_Demo_Box'*** and click ***OK.***



4. Click and drag the ***MainProgram*** onto ***My_Demo_Box*** so that the MainProgram is grouped under the My_Demo_Box folder.

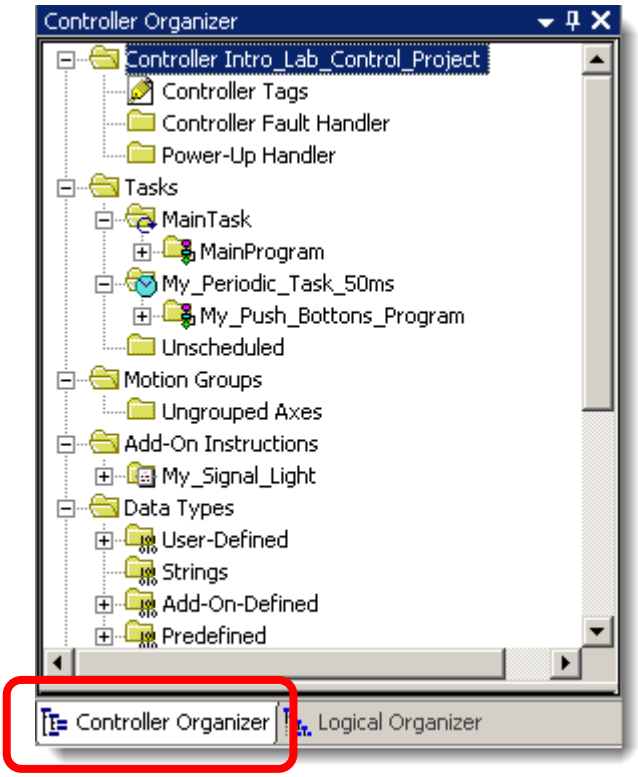


5. Do the same for ***My_Push_Buttons_Program*** to associate with ***My_Demo_Box***.



Notice that the two programs we are using to run the demo box are now grouped together. Anyone looking at the Logical Organizer will have a better idea that both programs are being used to run the demo box.

6. Click on the ***Controller Organizer*** and notice that the programs and tasks haven't changed.



Notice that the two programs we are using to run the demo box are now grouped together. Anyone looking at the Logical Organizer will have a better idea that both program are being used to run the demo box. In general, the organizer is used to group the program code to model the physical or logical application.

Congratulations! You have Completed Section 11. Please move on to Section 12.

Section 12: (Optional) Using Studio 5000 Help

This lab section should take roughly 10 minutes to complete.

Objective:

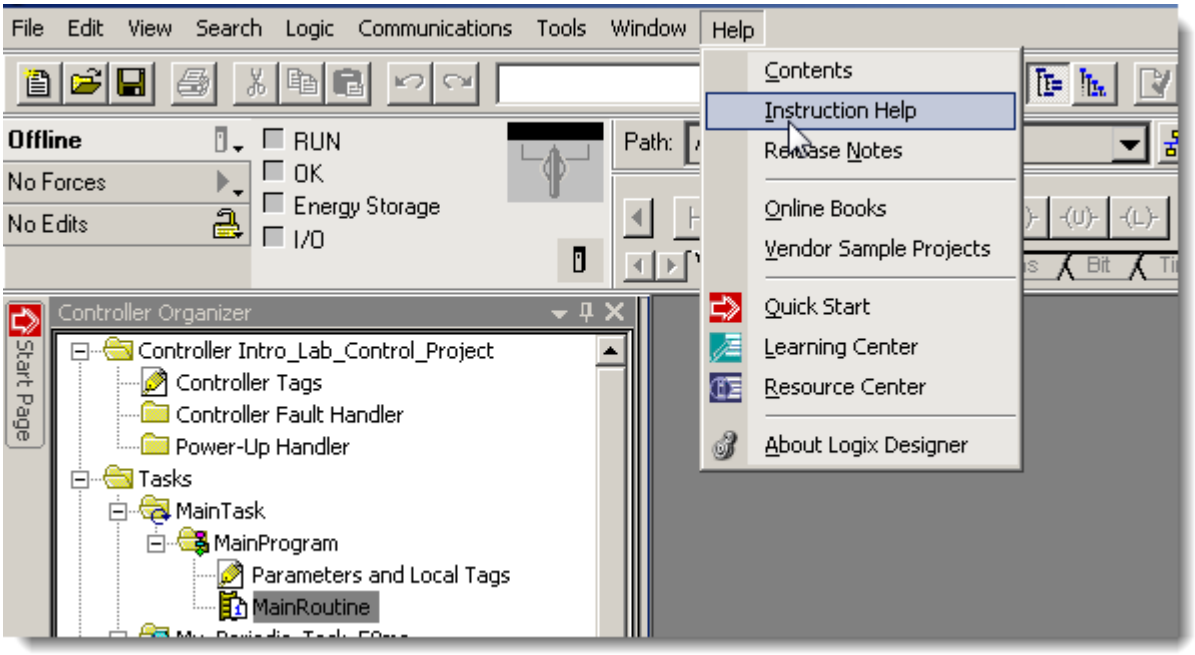
In this lab we will explore the extensive online Help system in Studio 5000. Feel free to look and poke around as desired.

In this lab you will be viewing:

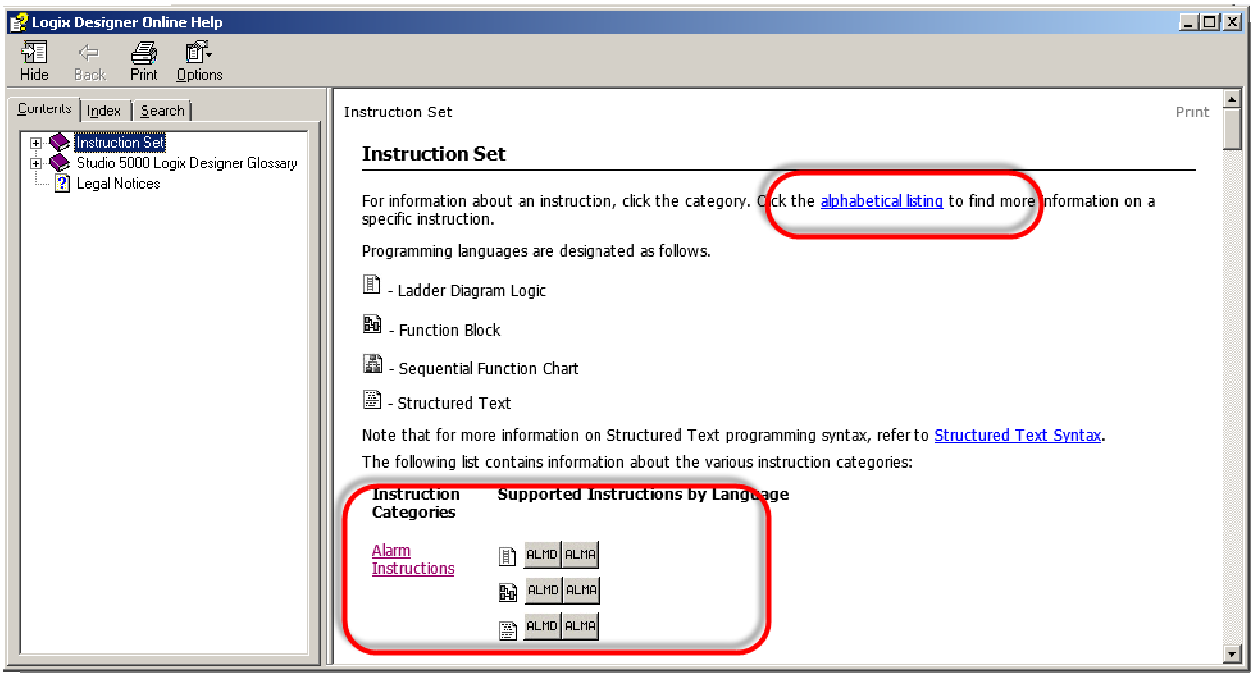
- Instruction help
- Module wiring diagrams
- On-line reference materials
- 3rd party vendor sample projects
- The Start Page – Quick Start

Instruction Help

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



The Help window will appear.



2. Click on the ***instruction links*** on the left (for example, Alarm Instructions), then click on an ***instruction icon*** to locate its description, details about its parameters, and related instructions along with examples on how to use the instruction.

   Alternately, pressing the ***F1 key*** while an actual instruction in ladder code is highlighted will also bring up the help for that instruction.

   The instruction help can also be accessed alphabetically as well.

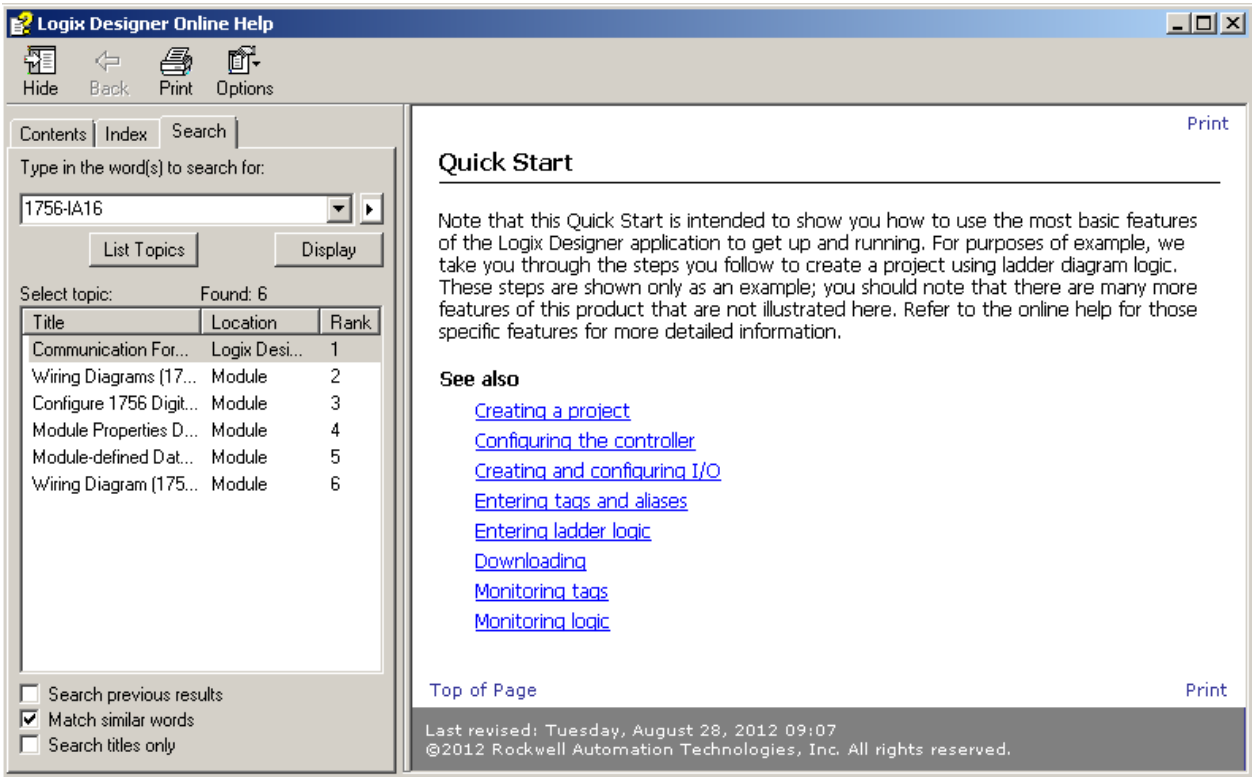
Viewing I/O Module Wiring Diagrams

3. From the ***Help*** pull down menu select ***Contents.***

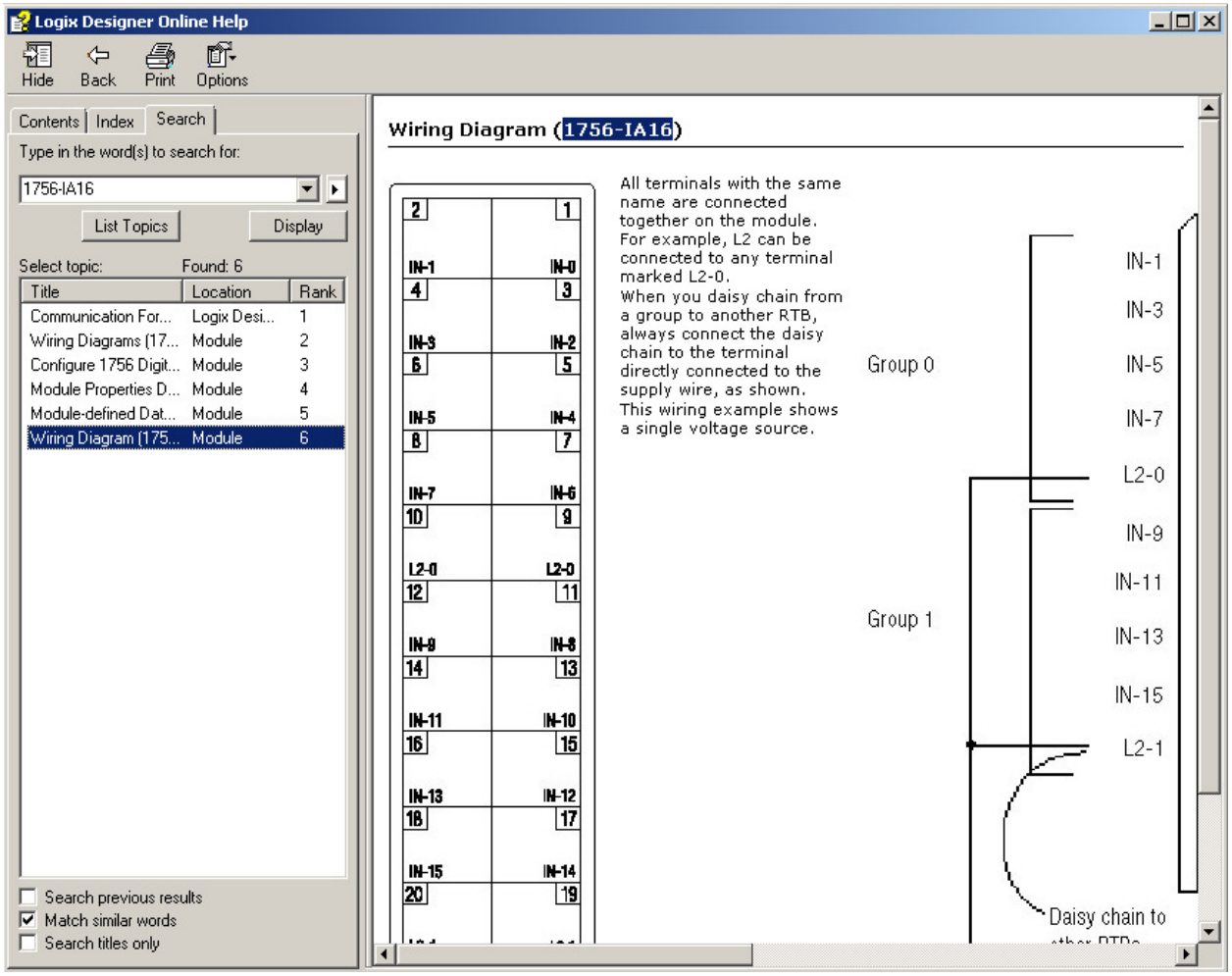
4. Select the ***Search*** tab if it is not already selected.

5. Type in ***1756-IA16*** as the **keyword** to find then click on **List Topics**.

6. Select a ***topic*** to display from the list such as, Wiring Diagram.



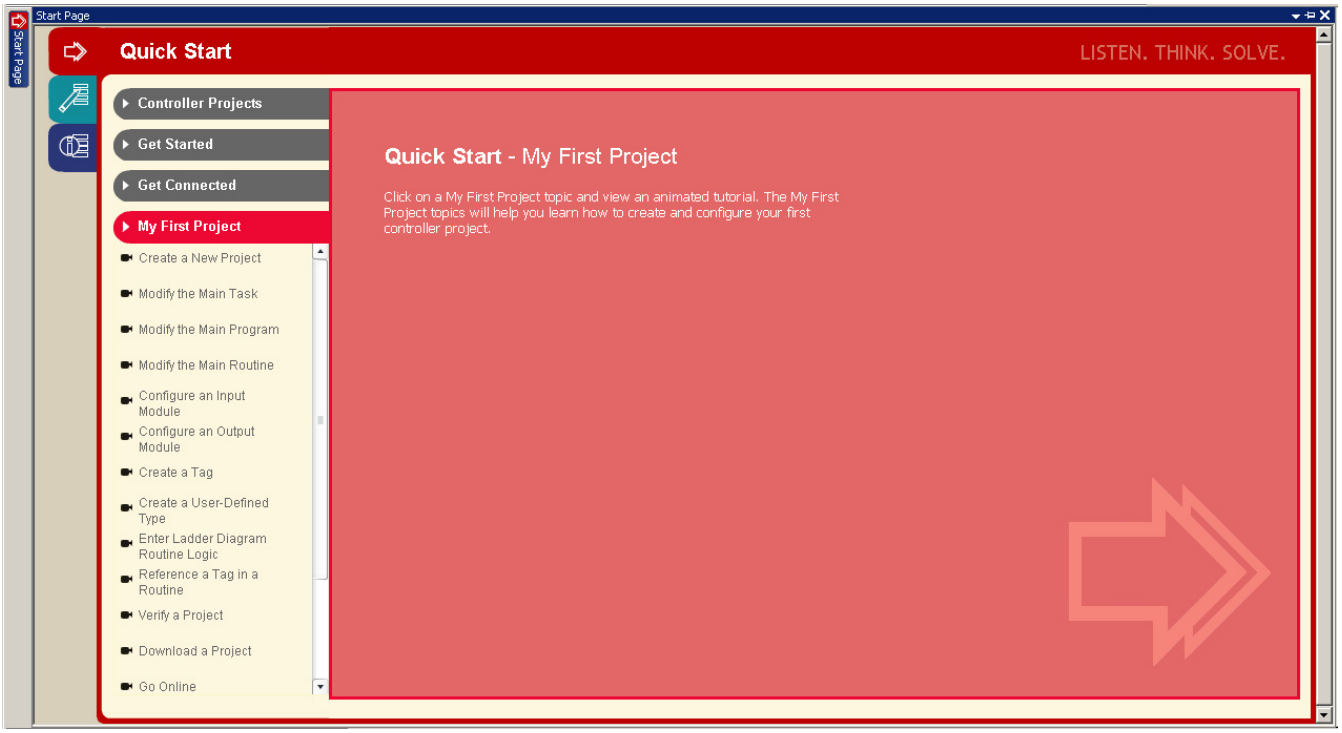
7. Click ***Display*** to view the wiring diagram for this module. Note you may need to maximize your screen.



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

Using Start Pages

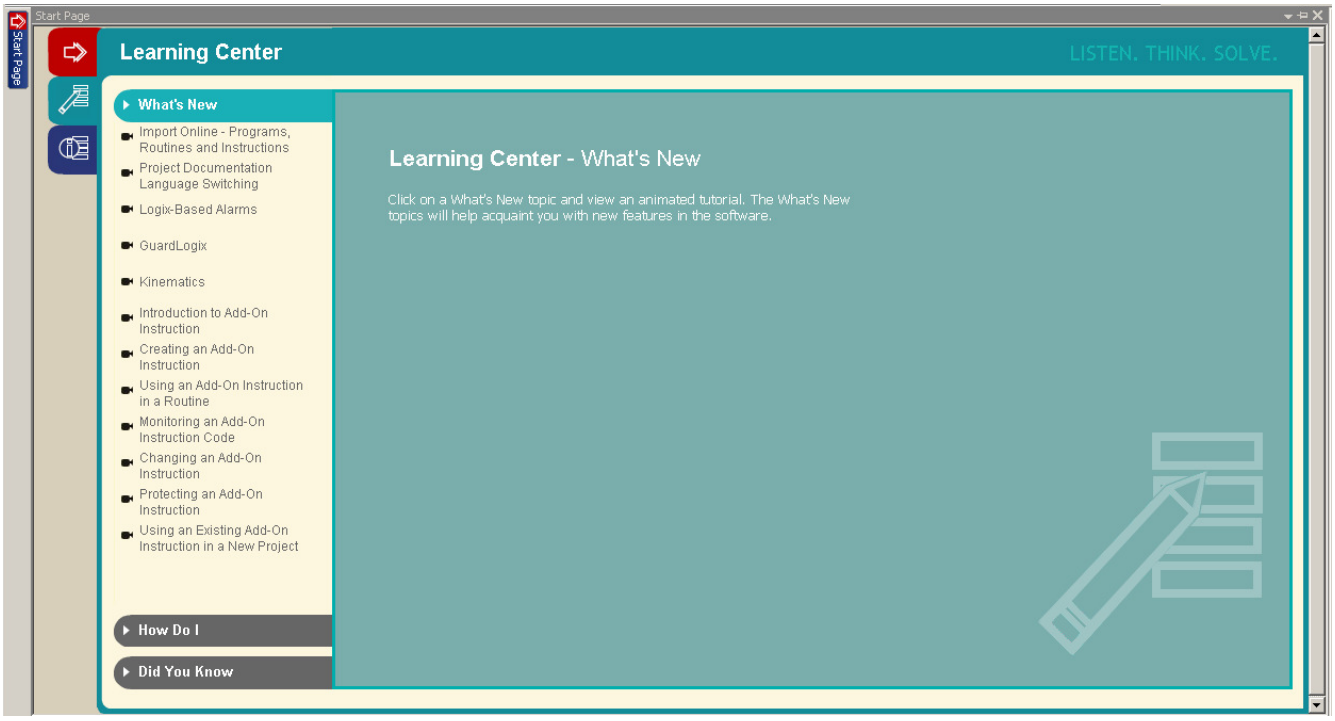
9. From the ***Help*** pull down menu select ***Quick Start***  which is one of the three tabs available from the **Start Page**.



- 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

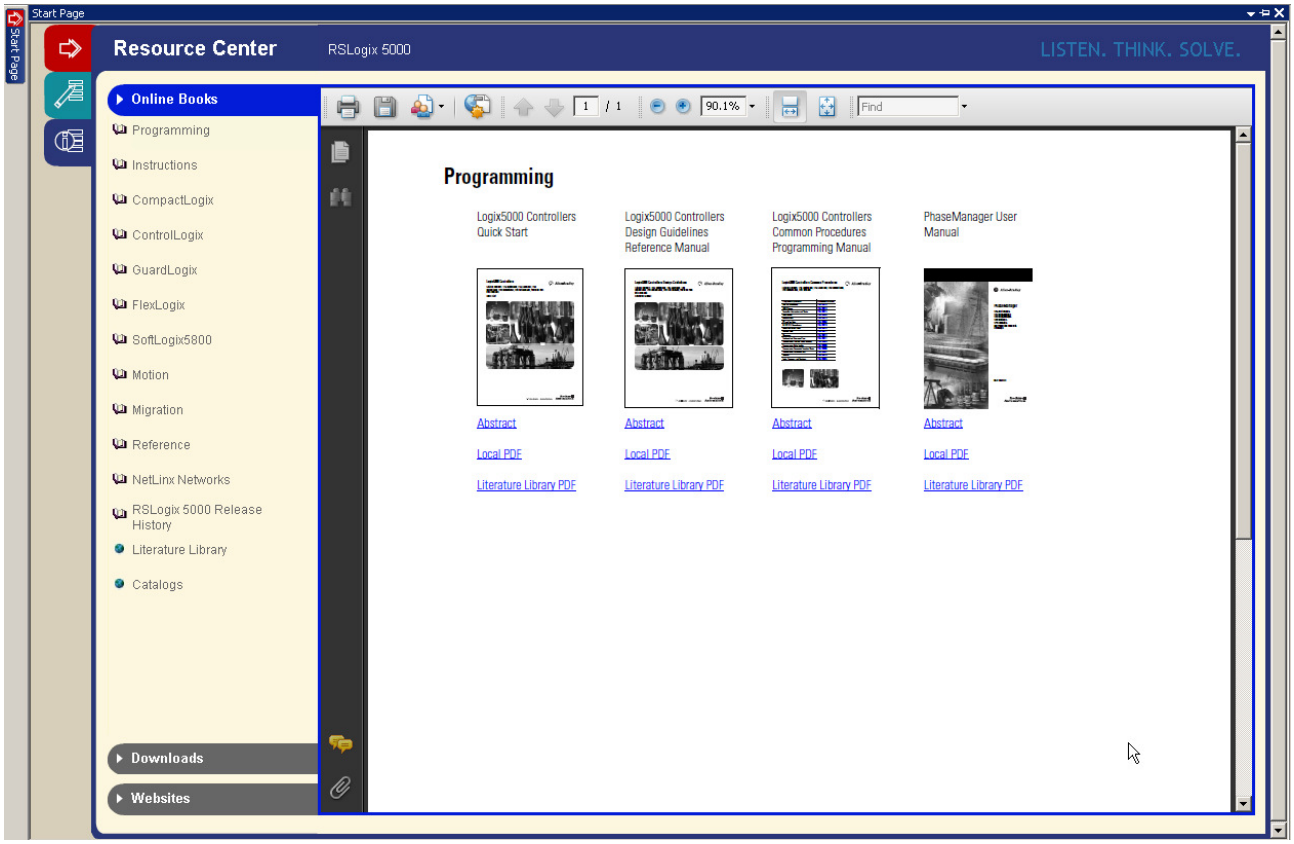
- Targets customers wanting to learn or explore how to use the software beyond just getting started reduces learning curve and helps increase productivity.



- ***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!

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication XXXX-XX###X-EN-P — Month Year
Supersedes Publication XXXX-XX###X-EN-P — Month Year