L25 - Studio 5000 Logix Designer®: Basics Lab



For Classroom Use Only!





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:



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.



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

ATTENTION

 \wedge

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.



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

Studio 5000 Logix Designer®: Basics Lab

Contents

Before you begin	5
About Studio 5000 Logix Designer	6
About CompactLogix Controllers	6
About Logix Controllers	7
Section 1: Creating a Project	8
Objective:	8
Launching Studio 5000 Configuration Software	8
Creating a New Controller Project	9
Adding Ladder Logic to the Main Routine	13
Creating Tags for the Ladder Code	
Monitoring/Editing Tags	
Section 2: Configuring I/O	
Objective:	
Adding CompactLogix I/O	
Viewing the CompactLogix I/O Tags	
Assigning Alias Tags	
Section 3: Connecting Your Computer to the Controller	50
Objective:	50
Launching RSLinx Software	50
Adding the AB_ETHIP (Ethernet/IP) Driver	50
Section 4: Downloading the Project from the Computer to the Controller	54
Objective:	
Downloading the Project to the Controller	55
Section 5: Testing Your Logic Program	58
Objective:	58
Switching the Controller into Run Mode and Testing the Program	

Section 6: Adding Logic and Tags Online	62
Objective:	62
Adding a MOV Instruction to the Logic	62
Adding the Timer to the Logic	65
Testing Your Logic	70
Section 7: Creating and Running a Trend	71
Objective:	71
Creating and Running a Trend	71
Section 8: (Optional) Creating and Using User Defined Types (UDT)	77
Objective:	
Creating User Defined Types	77
Add the UDT tag to an instruction	81
Monitoring UDT Tags	84
Section 9: (Optional) Using Periodic Tasks	
Objective:	85
Adding a Periodic Task	85
Section 10: (Optional) Creating an AOI (Add On Instruction)	92
Objective:	
Adding an AOI instruction	
Section 11: (Optional) Using Logical Organizer	100
Objective:	100
Using the Logical Organizer	100
Section 12: (Optional) Using Studio 5000 Help	103
Objective:	103
Instruction Help	103
Viewing I/O Module Wiring Diagrams	
Using Start Pages	
Learning Center Tab	
Resource Center Tab	109

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

Rockwell Software	lio 50	00	
	Create	Open	Explore
	New Project	Existing Project	Help
	From Import	Sample Project	About
Recent Projects	From Sample Project	From Upload	
Recent rojects			⊳

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.

Project Types	1769-136FRM
💕 Logix	1769-L36ERM CompactLogix™ 5370 Controller
🕥 View	1769-L36ERMS Compact GuardLogix® 5370 Safety Controller

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.

N <u>a</u> me:	Intro_Lab_Control_Pro	ject I		
Location:	: C:\Users\Labuser\Documents\Studio 5000\Projects Browse			
	Cancel	<u>B</u> ack	<u>N</u> ext	<u>F</u> inish

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:

🙆 New Project					<u>? ×</u>
1769-L36ERM (Intro_Lab_Control_	CompactLogix™ 5370 Control Project	ller			
Revision:	28 -				
Security <u>A</u> uthority:	No Protection	uthority f	or authentication	and authorization	1
Secure With:	© Logical Name <controller name<br="">© Permission Set</controller>			•	
Description:	Logix Basics Lab				
		S			
	Can	cel	<u>B</u> ack	Next	<u>F</u> inish

- 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.

<u>Controller Type:</u> 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.

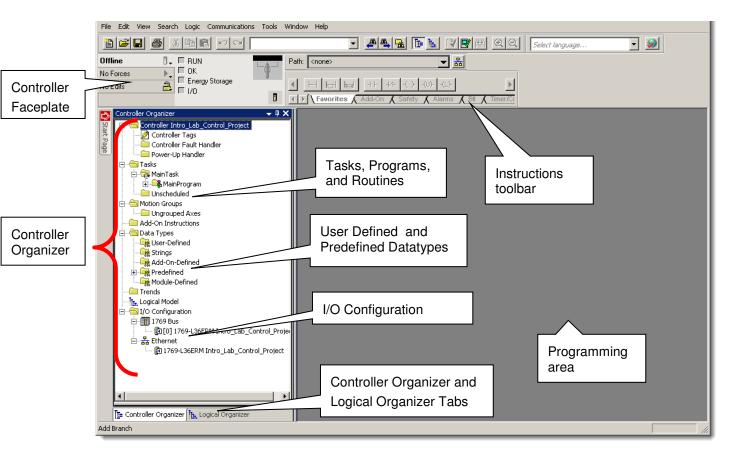
<u>Revision:</u> Here you are selecting the firmware revision of the project that will be created.

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

<u>Slot:</u> 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.

<u>Security Authority</u>: 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.

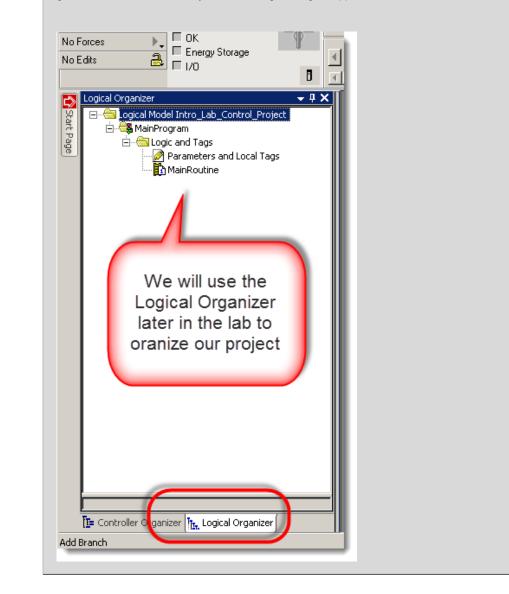


You have now created your first controller project!

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.

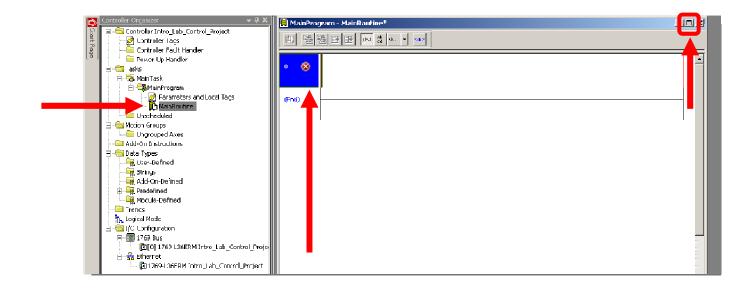
You will continue to use the project already open.

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

Controller Organizer 📃 🔫 🗸
Controller Intro_Lab_Control_Project M_ Controller Tags Controller Fault Handler Sever Up Handler
Tasks Tasks MainTask AninProgram Parameters and Local Tags MainRoutine Unscheduled
Motion Groups Motion Groups Motion Instructions Add-On Instructions Jata Types Motion Strings Model Model Trends Model
I/O Configuration - II/O Configuration - III 1769 Bus - III [0] 1769-L36ERM Intro_Lab_Control_Project - III 1769-L36ERM Intro_Lab_Control_Project

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.

H H				
电 醫院	E E abcd ab <abb< td=""></abb<>			
0 😣				
(End)				

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:

H H			
(End)			

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

Image: Text of the second s

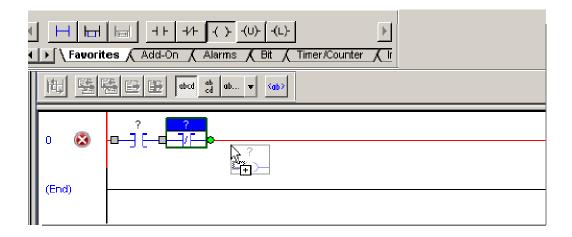
6. Drag the XIO onto rung 0 to the right of the XIC instruction as shown above. Again a green dot will appear to the right of the XIC instruction indicating where your new instruction will be inserted. Release the mouse button at the location you wish to place your instruction.

Verify your rung appears like the figure below:

_	H H						
	0 🐼						

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.



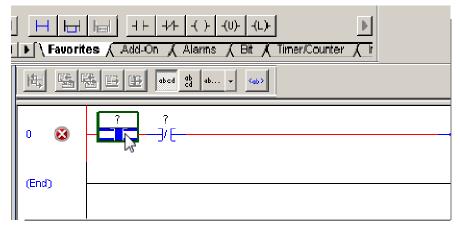
8. Drag the *OTE* onto **rung 0** to the right of the **XIO** instruction as shown above. Again a green dot will appear to the right of the XIO instruction indicating where the OTE instruction will be inserted. Release the mouse button at the location you wish to insert the instruction.

Verify the rung appears as follows:

H H					
陶醫	A E E abcd at ab (ab)				
0 🐼		?			
(End)					

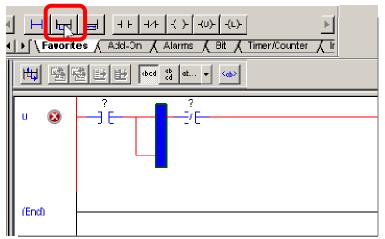
We will now add a branch around the XIC instruction.

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



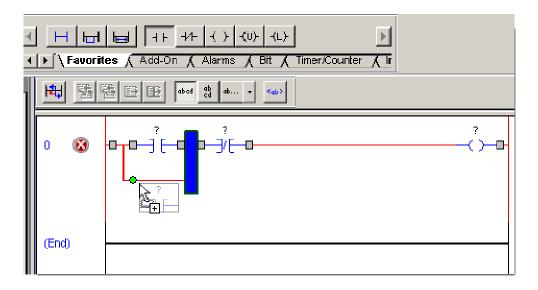
10. From the instruction toolbar click on the **Branch** instruction.

A branch will be inserted on the rung.

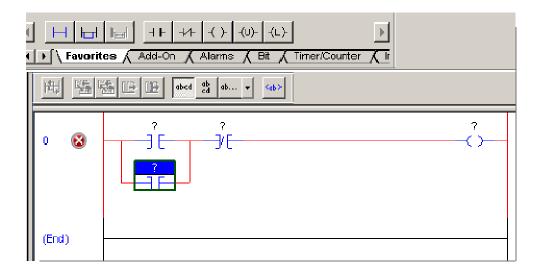


- 11. Left-click and hold on the *blue highlighted part of the branch* and drag your selected leg of the branch to the *left side of the XIC instruction*.
 - H H +/+ () (U) (L) ● ▶ **Favorites** Add-On Adarms A Bit A Timer/Counter χīr <u>ട</u> ടു 副 📑 abcd ab ab... 🔻 😘 H 8 JE-O 0 ヿ F λh. (End)
- 12. Place the branch over the green dot and release the mouse button.

- 13. From the instruction toolbar, left click and hold on the **XIC** $\stackrel{++}{\longrightarrow}$ 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** in 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**.

田雪	5 E E	abed ab v (ab)			
0 🐼	_	New Tag Cut Instruction Copy Instruction Paste Delete Instruction Add Ladder Element Edit Main Operand Description Save Instruction Defaults Clear Instruction Defaults Toggle Bit Eorce On	Ctrl+X Ctrl+C Ctrl+V Del Alt+Ins Ctrl+D	?	

A New Program Parameter or Tag window will appear.

New Paramel	ter or Tag	X
None:		Create 💌
Description:	<u>*</u>	Cancel
		Help
	T	
Usage:	I ocal I ag 💌	
-уре:	Base Connection	
Alias For:	<u> </u>	
Dia:a Туре:	DOOL .	
Palameter Connection:		
Scope:	🕞 Main ^o rugram 🔹	
External Access	Fead/Write	
Style:	Decimal 💌	
🔲 Constant		
🔲 Sequenci	ng	
🔲 Open Cor	figuration	
🔲 Open Par	ameter Connections	

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.

<u>Scope:</u> 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.

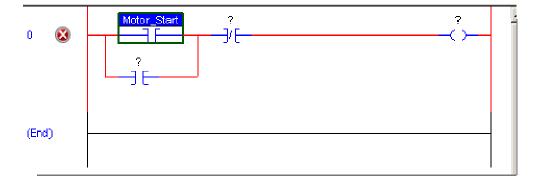
2. Enter the name "*Motor_Start*" and fill in the remaining fields as shown below.

Make sure the scope of the tag is MainProgram.

New Paramet	er or Tag	X
Name:	Molor_Start	Create 🖛
Description:		Cancel
		Нер
Lsaget	Lucal Tay	
Туре:	Base 💌 Connection	
Alia: For:	<u> </u>	
Data Type:	RUUI	
Farameter Connection	<u> </u>	
Scope	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	Decimal	
🗖 Donslart		
🗖 Bequencir	ng	
🔲 Open Con	liguration	
🔲 Dipen Para	ameter Connections	

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.

	- -(Bit 4	Timer/Counter / Ir		
0 🐼) 8 12	New Tag Cut Instruction Copy Instruction Paste Delete Instruction Add Ladder Element Edit Main Operand Description Save Instruction Defaults	Ctrl+X Ctrl+C Ctrl+V Del Alt+Ins Ctrl+D	

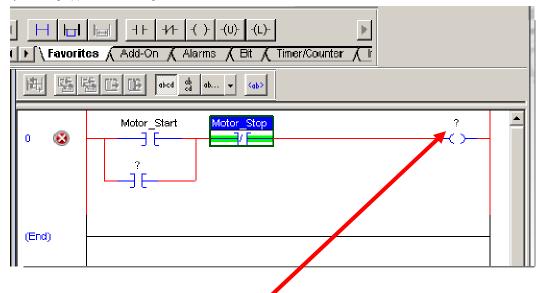
Again, the New Tag window will appear:

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

New Paramet	er or Tag	X
Name:	Motor_Stop	Crea:e 🖛
Description:	×	Cancel Help
Usage:	Local Tag	
Туре:	Base Connection	
Alas For:	T	
Data Type:		
Parameter Connection:	<u> </u>	
Scope:	🕞 MainProgram 📃	
External Access:	Read/Write	
Slyle:	Decimal	
🗖 Constant		
🔲 Sequencir	0	
🔲 Open Coni	figuration	
🔲 Open Para	meter Connections	

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.

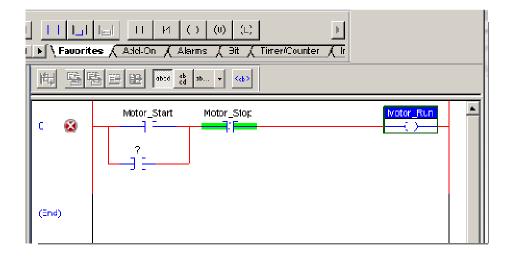
The New Tag window will appear.

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

New Paramet	er or Tag	×
Name:	Molor_Run	Create 💌
Description:	A V	Cancel Help
Usaget	Local Tag 💽	
Туре:	Base Connection	
Alias For:	<u> </u>	
Data Type:	BOOL	
Parameter Connection:		
Scope	🕞 Main Piogram 🔽	
External Access:	Read/Write	
Style:	Decimal	
🔲 Constant		
🔲 Sequence	ng	
🔲 Open Con	figuration	
🔲 Open Para	ameter Connections	

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

Your rung should now appear as shown below:

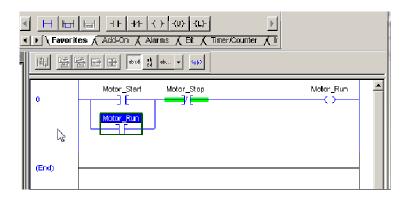


For the XIC instruction in the branch we do not have to create a tag. You will use the tag Motor_Run.

- 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.

() ► \ Favorit	H -/- (U) (L) > es Add On Alarms Bit Tmer/Counter I B B and C and and	
n 🐼	Motor_Start Motor_Stop	Mhtre_Pun
(End)		

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.

📕 File Edit	View	Search Logic	Communicati	ons Tools	Window H
2 2 2		<u>T</u> oolbars			
Offline	Ð	Zoom <u>I</u> n		P	ath: <none< td=""></none<>
No Forces	Q	Zoom <u>O</u> ut			
No Edits	E =	<u>C</u> ontroller Organi	zer A i t+0		
		Errors	Alt+1		(→
📷 Controller		<u>S</u> earch Results	^l ∕∂A l t+2	→ ₽ X	围垦
₩ <u>⊡~</u> @Co		<u>W</u> atch	Alt+3		
Start Page	Tr.	Logical Organizer	A i t+4		
		Start <u>Page</u>	Alt+9		0
📃 🚊 🔂 Ta	nsks	T			
	🖣 Mair	ıTask			
E	3 🚭	MainProgram			

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

Lugix Designer - Intro_Lab_Control_Project [1769-L File Edit View Search Login Communications Loois	
	🖃 🖉 🗮 🔚 💽 🔚 📃 🔠 🔍 🔍 🖉 Seel (angcaye 🖃 🌌
	Path: (none) 🗾 🛣
NoEdts	International Address & Add-On & Alarma & Bit & Timer/Counter & r
🖗 🕒 🔄 Controller Intro_tab_Control_Project	
Cuntroller Tags	Motor_Start Motor_Stop Mctor_Mun
Power-Up Handler	
📄 💼 🔄 Tasks 🔲 🔲 🔄 MainTask	Minior_R in
📋 🖼 MainProgram	
Parameters and Local Tags	
🛄 Urscheduled	(End)
- Motion Groups 	
- I Add-On Instructions 	
Interned	
	Image: State
È - 🤤 I)⊂ Cunfiguration ⊡ - 📶 1769 Bus	
🔚 📴 [0] 1769-LD6ERM Intro_ab_Control_Price	Froms Verifying Controller
문- 🚰 Ethernet 다. 👩 1769-106대M Intro_Leb_Control_Projec	Verifying routine 'MainRoutine' of program 'MainProgram'
	Veritying program connections Cumplete - Dierron(s), O warning(s)
Iyoe Ladder Diagram (Main) Description	
Program MainProgram	
📴 Conbroller Organizer 🗽 Logical Organizer	🙍 Errors 🔃 Search Results 🛃 Watch
Ready	SUBJET OF ANT

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"

💰 Logix Designer - Totro_Lab_Control_Project [1769-1361	
File Control View Search Logic Communications Tools V	Mindow Help
	🔽 🜉 🍇 🗽 🗽 🖉 🖉 🖳 🔍 🛛 Select language 💽 🎾
No Forces	
Sont oles Orcenteer • 9 × Controller Intro_Lat_Control Project • Controller Tags • Controller Fault Handler	
Toyler Un hander Tasks Tasks	0 Mitror Staft Motor Stan 0 Mitror Run (End) (En
Hind Predefined	Erors Verifying Controller Verifying routime 'MainPontime' of progrem 'MainProgrem' Verifying program connections Complete - 0 cluvi(s), 0 #alling(s)
E Controller Organizer	🧧 Errors 🗟 Search Results 😹 Watch
Greate Examine Off instruction	Rung (End) of 1 APP VER

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

The tag database of Logix versus a traditional PLC's fixed memory addresses help you create selfdocumenting 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.

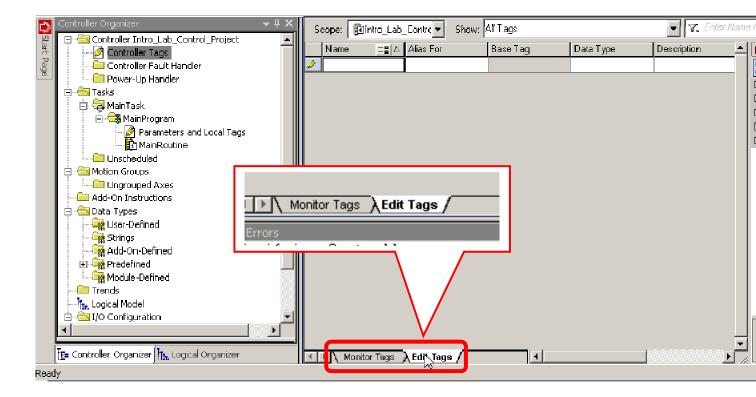
You will continue to use the project already open.

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

Offline Image: Constraint of the second se	Path: <none> Image: Horizontal and the state of the state o</none>
Controller Organizer	

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.

Pa	alh: <none></none>	
4	H H H -1+ -1/+ -(-)- (-)- Favorites & Safety & Alarms & Bit & Timer/Counter & Int	
	Scope: Bintro_Lab_Contre Show Al Tags	
	Name _=_∆ AliasEgr Base Tag Data Type Description External Access Constant Style ▲	Propert
		 Pro Con Par
	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.	

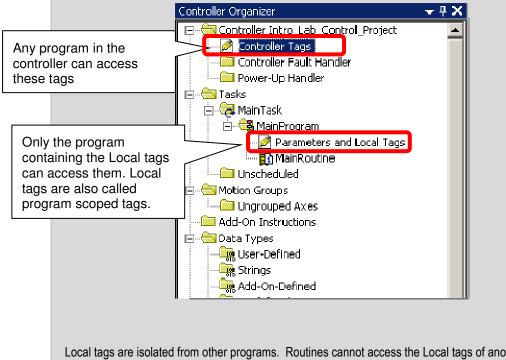
Data scope defines where you can access tags.

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

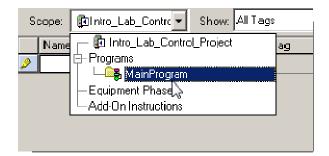
Data Scoping

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).



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** \rightarrow **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.

Win	dow Help										_l± 🗵
		- 4	88. 88. 58.	<u>e h 29</u>	QQ 5elec	t language	- 🥪				
'ath:	<none></none>										
đ	ны			+ -(U)(L)-	►						
्र 🗖	Favorite	c (Saf	iety 🔏 Alarm:	s 🖌 Bit 🔏 Timer/	Counter						
	1 avontes	N Our	<u> </u>								
	icope:	~		how: All Tags			• 7. <i>E</i>	nter Name Filter			•
		MainProgr			Base Tag	Data Type	Description	nter Name Filter External Access	Constant	Style	•
	icope: 🕞	MainProgr	ram 💌 S	how: All Tags	l	Data Type BOOL			Constant	Style Decimal	(m)
	icope: 🕞	MainProgr <u>=</u> 8 A Run	ram ▼ S Usage	how: All Tags	l			External Access			(m)
	icope: Sope: Sope: Name Motor_F	MainProgr === △ Run Start	ram 💌 S Usage Local	how: All Tags	l	BOOL		External Access Read/Write		Decimal	(m)
	icope: Name Motor_F Motor_S	MainProgr === △ Run Start	ram 🔽 S Usage Local Local	how: All Tags	l	BOOL BOOL		External Access Read/Write Read/Write		Decimal Decimal	Properties

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

Trends 	Moidel	ation		
Bus Size	i	New Module Discover Modules Paste	Ctrl+V	
<u>1945 9176</u>		Print	•	

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

atalog Modue Discov	very Favorites					
Ente: Search Texi h	or Medule Type		lear Filters]		Hide Filters 🗙
	Mucule Type Datego	ıy Filleis	I		Module Type Vendor	Filters
🗹 Analog				Allen-Bra	dley	
🛛 🗹 Communicatio	n			Acvance	d Micro Controls Inc. (AMI	J)
🗹 Digital				Hardy Pr	ocess Solutions	
🗹 Olher				Prosoit T	echnology	
🗹 Specialty				Spectrum	- Controls, Inc.	
J						
Catalog Number	Description	Vendor	Category			▲
-642	Dual Resolver In	Advanced Micro	Specialty			
769-ASCI	2 Channel RS 23	Aller-Bradley	Specialty			
769-Boolean	8 Point Inpul, 4	Aller-Bradley	Specialty			
1769-HSC	High Speed Cou	Aller-Bradley	Specialty			
1700-IA10	10 Point 120V A	Aller-Dradley	Digital			
769-1481	8 Point Isolated	Aller-Bradley	Digital			
769-IF16C	16 Channel Curi	Aller-Bradley	Analog			
769-IF167	16 Channel Volt	Aller-Bradley	Analog			•
57 of 57 Module Typ	e: Found					Add to Favorites

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

lect Module Type Catalog Module Discor	very Favorites					
IQ6		Clear Filters]		Hide Filters 🕱	
	Module Type Category Filters		Modu	le Type Vendor Fil	ters	_
🗹 Analog			Allen-Bradley	-		
🔽 Communicatio	n		Advanced Micro Co	ntrols Inc. (AMCI)		
🗹 Digital			Hardy Process Solu	tions		
🗹 Other			Prosoft Technology			
Specialty			Spectrum Controls,	Inc		
Catalog Number 1769406KDW4	Description Vendor	Category				
1 of 57 Module Type	s Found			Create	Add to Favorite	s Help

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.

Type: Vendor:	Allen-Eradley	16 Point 24V D.C. Sink /Sr		and the second second
Parent	Local			
Name:	Digita_Card_1		Slat	1 -
Description:		A V		
- Module Diefin				
Seres	В	Charge		
Revision	2 001			
Electronic Ke		ible Module		
Connector:	Eutou:			
Diata Formati	Integer			

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

Click on the *Change...* Button and change the <u>Series to A</u>. Notice the <u>revision changes to 1</u>. Click OK.
 Note: Some of the demos have series B modules. Compatible module keying will allow earlier profile version to work with newer modules.

Module Definition*	×
Series: Revision: Electronic Keying: Connection:	A 1 Compatible Module Dutput
Data Format:	
ОК	Cancel Help

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

RSLogix	5000	\times
<u> </u>	These changes will cause module data types and properties to change. Data will be set to default values unless it can be recovered from the existing module properties. Verify module properties before Applying changes.	
	Change module definition?	
	Ves No	

13. The properties window should now appear as follows.

🔜 New Module		X
General" Conn	ection"	_
Type: Vendor: Parent:	1769-IQ6XOW4 6 Point 24V DC Sink/Source Input, 4 Point AC/DC Relay Output Allen-Bradley Local	
Name: Description:	Digital_Card_1	
- Module Defin Series: Revision Electronic Ka Connection: Data Format:	A Change 1.001 ying: Compatible Module Output	
Stalus: Creating	DK. Cancel Help	

<u>Electronic Keying</u> – 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:

- <u>Exact Match</u> All of the parameters described above must match or the inserted module will reject the connection.
- <u>Compatible Module</u> 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.
- <u>Disable Keying</u> No keying used at all. This is not typically used.

<u>Connection</u> -- Input only modules use "Data". Modules that include outputs use "Output".

<u>Data Format</u> -- 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.

🗖 New Module 🛛 🕅
General Connection
Requested Packet Interval (RPI) 20.0 👘 ms (0.5 - 750.0)
Inhibit Module
💌 Najor Faut Dh Controller If Connection Fails While in Run Nado
Mudule Faut
Status Creating

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.

Select Module Type	
Catalog Module Discovery Favorites	
Cording Module Discovery Pavoines	
	ar Filters 🙊 Hide Fillers 🙊
Module Type Category Filters	Module Type Vendor Filters
Analog	Allen-Bradley
Communication	Advanced Micro Control: Inc. (AMCI)
🗹 🗹 Digital	Hardy Process Solutions
Other Other	Prosoft Technology
Specially	Spectrum Controls, Inc.
	1
	Category
	Analog
	Analog
	Analog
1769sc-IF4IH 4-Channel 4308 Spectrum Control / 4 Channel Input/2 Channel Out	put Low Resolution Analog
, 	
5 of 57 Module Types Found	Add to Favorites
Close on Create	Create Close Help

<u>Note:</u> 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"

Type: Vendor:	17691F4XOF2 4 Channel input/2 Channel Output Low Resolution Analog AllenBradley
	Local
Name:	Analog_Card_2 Slot 2
Description	× 7
_Module Definit	
Seres: Revision	A Change
Electronic Key	1.1
Connection:	Output
Data Formai:	Integer

- 19. Click on the *Input Configuration* tab.
- 20. Enable all 4 channels by clicking a *check* in each box.

Indule Properties: L	nnal:2 (1769-164)	(NF2 1.1)			
	nr al:2 (1769-1743)		guration		
 atus: Offine	[CK	Cancel	Арру	Help

- 21. Click on the *Output Configuration* tab.
- 22. Enable both channels by clicking a *check* in each box.

🔜 Module Properties: Local:2 (1769-IF4X	X0F2 1.1)	<u>_ ×</u>
General Connecton Irput Configuration	Output Configuration*	
Status: Offline	OK Cancel Appy He	lp

- 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.

s	cope: Dintro_Lab_Contr	: 🔻 Show	w: All Tags				Y. Enter Name Filter	r	
	Name	<u>-8</u>	Allas For	Base Tag	Data Type	Description	External Access	Constant	Style 🔺
					AB:1769_IQ6X0		Read/Write	Г	
					AB:1769_IQ6X0		Read/Write	Г	
	➡-Local:1:0				AB:1769_IQ6X0		Read/Write	Г	
	➡-Local:2:C				AB:1769_IF4X0F		Read/Write	Г	
	➡-Local:2:1				AB:1769_IF4×0F		Read/Write	Г	
	➡-Local:2:0				AB:1769_IF4×0F		Read/Write		
ø									

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".

Scope:	Dintro_Lab_Contrc	Sho	w: All Tags	_
			Alias For	Ba
<u>∓-L</u> α	peal:1:C			
_	peal:1:I			
	ocal:1:0			
H +-Lo	ocal:2:C			

	An I/O address follows this format:
	Location :Slot :Type .Member .Bit
Where:	= Optio
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
Туре	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.

Favorites	Ada	3-On 🔏 Al	<u>A ama</u>	Rit 🖌 Tiner/C	inunte	er <u>A</u> lf						
Scope: 🚺	ntro Lab	Contro 🔻	Show:	ଧା Lags					▼ Y.	Enter Name	Filier	•
Name	-= A	Value	÷	Force Mask	÷	Style	Data Type	-	Description	Constant		
+-Local:1:U			{}	{.)		AB:1769_1Q6X0			Γ		
			{}	{ -)		AD:1700_IQ0>(0					Properties
) <mark> </mark>		{}	{.)		AB:1769_1Q6×0			Г		l te
⊞-Local:2:0			{}	{.)		AB:1769 IF4×0	F				
			{}	{.)		AB:1769_IF4×0	F		Г		
🔄 🗌 Local:2:0	ן כ		{}	(.)		AB:1769_IF4%0	F		Г		
	•										-	
	r Tags	🖌 Edil Taga	1									2

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:

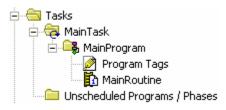
-Program logic in advance of wiring diagrams

-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*.

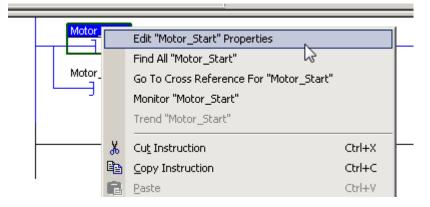


The ladder editor appears as shown below:

Image: Second and Second an									
0	Motor_Start Motor_Stop Motor_Run	-							
(End)									

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 change

💰 Tag Propert	ies - Motor_Start	<u> </u>
General*		
Name:	Motor_Start	
Description:	A	
		
Usage:	Local Tag	
Туре:	Alias Connection	
Alias For:	_	
Data Type:	BOOL	
Scope:	🕞 MainProgram	
External Access:	V	
Style:	Decimal	
Constant		
🗖 Open Para	ameter Connections	
	OK Cancel Apply	Help

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.

Alias For:		
Data Type:	Y. Enter Name Filter ▼ Show: All Tags	-
Scope:	Name <u>III</u> Data Type Usage De	scripti 🔺
	¶	
External	∎ Elecal:1:I AB:1769_IQ6 <controller></controller>	
Access:	∎ Eucal:1:0 AB:1769_IQ6 <controller></controller>	
Style:	∎ Local:2:C AB:1769_IF4 <controller></controller>	
	∎ H_Local:2:1 AB:1769_IF4 <controller></controller>	
🗖 Constant	jLocal:2:0 AB:1769_IF4 <controller></controller>	+
🔲 Open Param	Show controller tags	
	💌 Show program tags	
	Show parameters from other program:	
	<none></none>	

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

Usage:	Loc	al Tag			•				
Туре:	Alia	is 💌	Conne	ctior	ì				
Alias For:					•				
Data Type:		, Enter Name Filter.		•	Show: All	Tags			•
Scope:		Name		Data	Туре	Usage	De	scripti	▲
Scopo,	đ	, ⊥ -Local:0:C		AB:1	756_DO	<controller></controller>			
External	0	l,+-Local:0:I	1	AB:1	756_DO	<controller></controller>			
Access:	١ð	, Local:0:O	1	AB:1	756_DO:	<controller></controller>			
Style:	١ð	, argenting and the termination of termina	1	AB:1	756_DI	<controller></controller>			
Constant	Ē	, and the second secon	1	AB:1	756_DI	<controller></controller>			
	ð	<u>∓</u> -Local:7:C		AB:1	756_AO	<controller></controller>			-
🗖 Open Parar		Show controller tag	s						
	-12	Show program tags							
	Sho	ow parameters from (other pro	ogra	m:				
	<n< th=""><th>one></th><th></th><th></th><th>ľ</th><th>•</th><th></th><th></th><th></th></n<>	one>			ľ	•			

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.

Alia: For	Local1:LData	
Data Type:	Ŋ, Enfar Name Filler ▼ Show: All Tag: ▼	
Scope:	Name Image Descripti Image Descripti ▲ Image AB:1769_IQ3 <controler></controler>	
External Access:	AB:1769_Q5 <controler></controler>	
Style:		
🗖 Constant	0 1 2 3 4 5 6 7 0 1 2 3 4 5 7 0 1 2 4 5	
🔲 Open Param	Show controler tags	
	Show program tags	
	Show parameters from other program:	
	<none></none>	

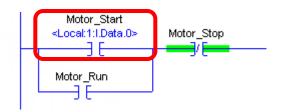
💰 Parameter/	Local Tag Properties - Motor_Start	
General*		
Name:	Motor_Stait	
Description:	A	
U∻a <u>c</u> e:	Local Tag 💌	
Турес	Alas Connection	
Alas Ful	Lccal 1:1.Data.0	
Data Type:	800	
Scope:	🕞 MainProgram	
Extema Access:	Read/Write	
Style	Decimal	
🔲 Constant		
🔲 Open Para	ameter Connections	
	OK Carcel Apply	Help

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:1:0.Data.0

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

Motor_Start <locat:1:i.data.0> E Motor_Run <locat:1:0.data.0></locat:1:0.data.0></locat:1:i.data.0>	Motor_Stop <local:1:i.data.1></local:1:i.data.1>	Motor_Run <local:1:0.data.0> ()</local:1:0.data.0>

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

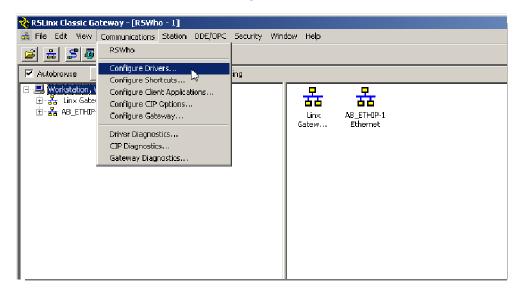
1. Double click on the **RSLinx icon** on the Desktop to launch RSLinx software to bring up the RSLinx Classic Gateway window.



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.

Help
Configure
Startup
Start
Stop
Delete
13

- 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

Configure Drivers		? ×
Available Driver Types: EtherNet/IP Driver	Add New	Close Help
Configured Drivers: Name and Description Add New RSLinx Classic Driver AB_ETHIP-1 A-B Eth Choose a name for the new driver. (15 characters maximum) AB_ETHIP-LAB	Cancel	Configure Startup Start Stop Delete

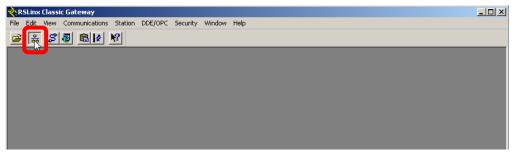
5. Choose "Browse Local Subnet" and the "Intel" network driver as shown. Click OK.

Configure driver: AB_ETHIP-1			? ×
EtherNet/IP Settings			
Browse Local Subnet	O Browse Remote Subnet		
Description		IP Address	
Windows Default			
Intel(R) PR0/1000 MT Network 0	Connection	192.168.1.1	
I			
	OK Cancel	Apply H	lelp

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.

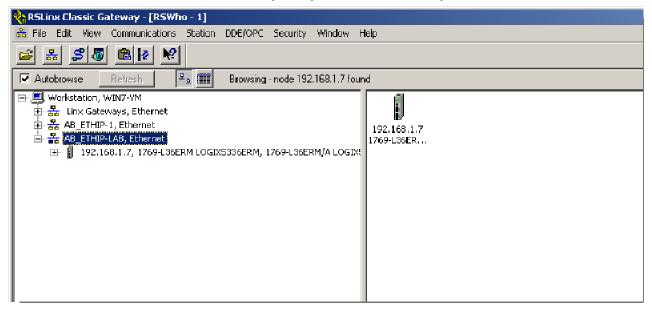
7. If no drivers are shown the software background is grey, click the **RSWho** icon in the toolbar. Otherwise move on to the next step.



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.

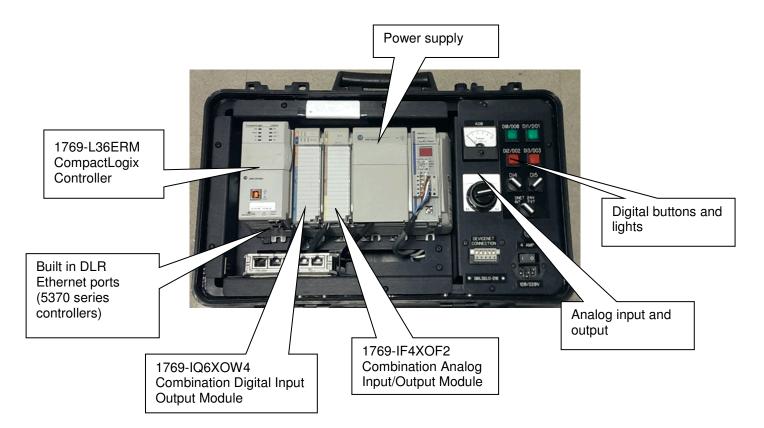
Objective:

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.

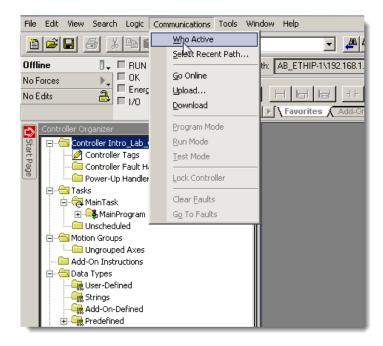


NOTE: Demo boxes may vary

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.

💰 Who Active	_ _ _ ×
Autobrowse Refresh	
िच. Workstation, WIN7-VM सफ्रि. Linx Gateways, Ethernet	Go Online
・ 器 AB_ETHIP-1, Ethernet	Upload
Ê - 윪 AB_ETHIP-LAB, Ethernet	Download
	Update Firmware
	Close
	Help
Path: <none></none>	Set Project Path
Path in Project: <none></none>	Clear Project Path

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

💰 Who Active	
Autobrowse Refresh	
E텔 Workstation, WIN7-VM 판금 Linx Gateways, Ethernet 판 몲 AB_ETHIP-1, Ethernet 판 몲 AB_ETHIP-1 AB, Ethernet	Go Online Upload
	Download
	Updale Firmware Close
	Help
Path: AB ETHIP-LAB\192.168.1.7	Set Project Path
Path in Project: <none></none>	Clear Project Path

. (

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

The project will then begin to download to your controller.

Download		×
<u> </u>	Download offline project 'Intro_Lab_Control_Project' to the controller.	
	Connected Controller:	
	Name: Intro_Lab_Control_Project	
	Type: 1769-L36ERM/A CompactLogix™ 5370 Controller	
	Path: AB_ETHIP-1\192.168.1.7	
	Serial Number: 603BC36C	
	Security: No Protection	
	The controller is in Remote Run mode. The mode will be changed to Remote Program prior to download.	
	DANGER: This controller is the system time master. Servo axes in synchronized controllers, in this chassis or other chassis, may be turned off.	
	DANGER: The controller image stored in nonvolatile memory might be out of date following the download. Failure to update the contents of nonvolatile memory could result in running old logic following a power up or corrupt memory condition.	
	1 DANGER: Unexpected hazardous motion of machinery may occur.	
	Some devices maintain independent configuration settings that are not loaded to the device during the download of the controller.	
	Verify these devices (drives, network devices, 3rd party products) have been properly loaded before placing the controller into run mode.	
	Failure to load proper configuration could result in misaligned data and unexpected equipment operation.	
	Download Cancel Help	

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.

Rem Run	1.	🗖 Run Mode	
No Forces	▶_	Controller OK Energy Storage OK	P i
No Edits	2	Energy Storage UK	

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.

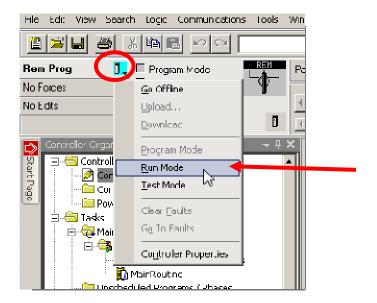
Objective:

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.

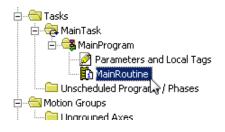
Rem Run	٥.	🗖 Run Mode	
No Forces	⊳⊧_	Controller OK Energy Storage OK	P
No Edits	2	Energy Storage UK	
Redundancy	₽¢Ţ	- 10 01	

Notice that the faceplate shows four controller status LEDs.

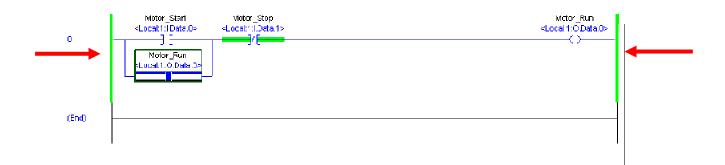
2. From the Controller Organizer expand the *MainProgram* by clicking on the "+".



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.

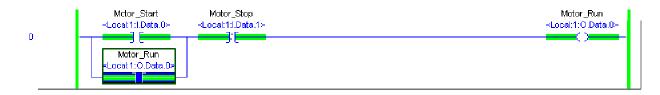


This correlates to the XIO instruction for Motor_Stop. Notice the instruction is no longer green because the instruction is no longer true.

Motor_Stop
≺Local:1:I.Data.1≻

5. Press button DIO (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.

U	Motor_Start	Motor_Stop	Motor_Run
	<locat:1: .data.0=""></locat:1:>	<locat:1:i.data.1></locat:1:i.data.1>	<local:1:o data.0=""></local:1:o>
	Motor_Run <local:1:0.data.0></local:1:0.data.0>		

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

	Motor_Stert Motor_Stop	Mctor_Run
	<local:1:i.data.0> <local:1:i.data.1></local:1:i.data.1></local:1:i.data.0>	<local1:0.data.0≻< th=""></local1:0.data.0≻<>
0		O
	Motor Run	
	<local:1:o.data.0></local:1:o.data.0>	

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* H on the toolbar.

No Forces ▶, ■ Controller OK ♥ = No Edits	Path:ETHIP-1\192.168.1.128\Backplane\8" 🗾 📇
Controller Organizer	Image: Second state Motor_State Motor_Sta

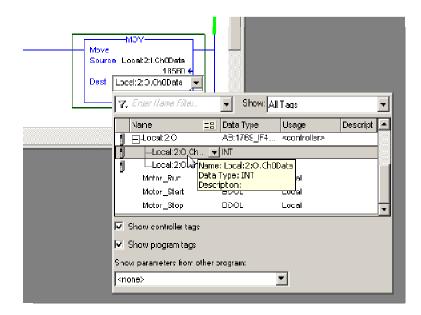
3. Use the *scroll buttons* if necessary to scroll to the *Move/Logical* instruction group tab in the instruction toolbar. Under the **Move/Logical** category tab, click and drag a *MOV* instruction to the new rung.

em Run III Parkan Run Mode	AltrETHIP-1\192.168.1.128\Backplane\8" _ 器 H h_ h_ h_ h_ hov num and a xoa not swee () (Compute/Math Move/Logical File/Misc. (File/Shift (
Controller Organizer	Motor_Start Motor_Stop

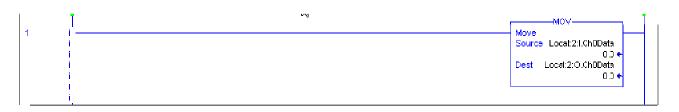
4. Double click the '?' by the source in the MOV instruction and select *Local:2:I.Ch0Data* by double-clicking the tag. You might have to *scroll down* to find the Channel data tags.

Move Source	MOV-			
Dest L		Show: Z	Usaye	▼ Descripti →
	l −Local:2:1.ChC.	VINT Data NT me: Local:2:I ChIDs ta Type: INT scrption: tRa NT		
	Show controller tags Show program tags Show parameters from o (none)			

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.

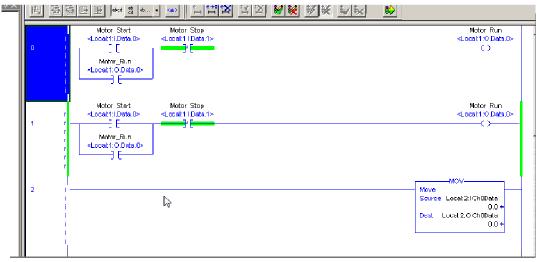


Adding a Timer to the Logic

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

0	Motor_Start ~Local:1:I.Data.0:][] Mutor_Run	Motor_ KLocal:11. 	Motor_Run ⊲Local:1:0Data.0 €Э
j Be	ji <u>C</u> opy Rung	Ctrl–X Ctrl–C Ctrl–V	
1	Delste Rung Add Rung Edit Rung Edit Rung Comment Import Rungs Export Rungs	Del Ctrl–R Enter Ctrl–D	Move Source Locat21Ch0Dsta 0.0 6 Cest Locat20.Ch0Data 0.0 6
	Start Pending Rung Edits Accept Pending Rung Edits Cancel Perding Rung Edits	Ctrl–Shift+S	

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

		RTO CTU	CTD RES	
Favorites	Add-On 🔏	Alarms 🔏 E)it λ Timer/Counter /	Input/Output

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

0 😵	Motor_Start <loca:1:i.data.0> </loca:1:i.data.0>	Motor_Stop <local:1:idata.1> //</local:1:idata.1>	Motor_Run ≺Local:1:0.Data.0⊧ < >	Timer Cn Delay -(EN) Timer ? Preset ? -(DN)- Accum ?
1 r 1 r r r r	Molor_Start <loca:1:i:data.0> </loca:1:i:data.0>	Motor_Stop ≺Local:11:Data.1>]/ [Motor_Run ≺Local:1:0.Data.0> < >

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*.

× •			
Motor_Run <local:1:0 data.0=""></local:1:0>	TON Timer On Delay Timer Preset Accum	New Tag.	Ctrl+X Ctrl+C Ctrl+V
		<u>D</u> elete Instruction	Del

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.

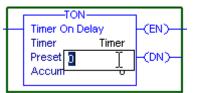
11. In the Name field enter '*Timer*' then click *Create*

New Program	Parameter or Tag	×
Name:	Timer	Create
Description:	A	Cancel
		Help
		· · /
Usage:	Local Tag	
Parameter Connection:		
Туре:	Base Connection	
Alias For:	<u> </u>	
Data Type:	TIMER	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	_	
Constant		
🔲 Sequencin	g	
🔲 Open Coni	figuration	
🔲 Open Para	meter Connections	

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

	Motor_Start <local:1:i.data.0> C Motor_Run <locat:1:o.data.0></locat:1:o.data.0></local:1:i.data.0>	Motor_Stop <local:1:i.data.1> 3/[]</local:1:i.data.1>	Motor_Run <local:1:o.data.0> ()</local:1:o.data.0>	TON- Timer On Delay -(EN)- Timer Timer Preset 0 Accum 0
1 r - r r	Motor_Start <locat:1:1.data.0> Motor_Run <locat:1:0.data.0></locat:1:0.data.0></locat:1:1.data.0>	Motor_Stop ≺Local:1:I.Data.1>]/[Motor_Run <locat:1:0.data.0></locat:1:0.data.0>

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.

Timer On		
Timer Preset Accum	Timer <u>32767</u> 0	-(DN)
Accum	U	

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

		醫臣臣 [bed 양 ob	· who is the state of the state	K 🗙 🕺 👯 🗎 🕅		
0	i i i i i	Mttor_Start <local:11.data.0> Moter_Run <local:1 o.data.0=""></local:1></local:11.data.0>	Motor_Stop <loca:1:idata:1>]/ []</loca:1:idata:1>		Mnhr⊾Run ≺Locatit:OData.0≻ ()	Timer On Delay Imer I mer Preset 02707 Acroum C
1	r r r r r r	Motor Start <local:11.data.0> U Motor_Run <local:10.data.0></local:10.data.0></local:11.data.0>	Motor Stop ⊲Loca:1:IData.1≻ /			Motor Run ≺Lccal/1:0.Data.0> (`)
2	i i i i					MOV Source Local.21.Ch0Data 18580 ← Cest Local.2:0.Ch0Data 18550 ←
(End)	I					

.

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

RSLogix 5000	×
 Finalize all edits in program 'MainProgram'. Finalize all 'i', 'i', 'd', 'I' and 'D' rung edit zones in all ladder routines in this program. Finalize all edits in pending and test edits views of all other routines in this program. 	
The following routines contain edits:	
A MainRoutine	
The Finalize All Edits in Program operation will leave the following outputs in their last state:	
 Dutputs in 'r', 'd', 'R' and 'D' rung edit zones. Butputs in the Original View. Outputs in the Original View. Outputs in the Original View. 	
Indicates Sequential Function Chart routines that will be reset to their initial steps along with their stored actions being reset.	
This operation cannot be undone.	
Finalize all edits in program?	
Yes No Help	

The ladder editor will now appear as follows:

o	Motor Starl Mctor Stop <local:1:date:0= <local:1:deta:1=""> Motor_Fun <local:1:d.data.j>]</local:1:d.data.j></local:1:date:0=>	Motor Run <_ocel.1:O.Data.C> Timer Ou Delay (CEN) Umer Umer Preset 32767 (ON) Accum 0 (CEN)
1		MOV Source Locat2:LCP0Data 16560 € Des: Locat2:O.CP0Data 18560 €
(End)		

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 *Al0* potentiometer to 5.
- 23. Verify that the *AO0* meter reads 5 Volts.
- 24. Turn the *Al0* 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*.

- L	<u> </u>			
		New Trend		
_		Import Trend	М	em
		Open Trend Log.		
				ocalENB
	¥	Cut	Ctrl+X	
	Þ	Сору	Ctrl+C	
	ß	Paste	Ctrl+V	Discrete_IO nalog IO

w Trend - Genera	ı		
Name:			
Description:		<u>^</u>	
		T	
Sample Period: 1	0 📑 M	lillisecond(s) 💌	

The New Trend window appears.

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

Name:	My_Trend		
Description:		* *	
Sample Period:	10	Millisecond(s) •	

3. Click Next.

The New Trend Add/Configure Tags window appears.

New Trend - Add/Configure Tags	X
Intro_Lab_Control_Project	
AvailableTags:	
Name	- I
Local:1:C	
E Local:1:1	
] ⊞Locat 2:C] ⊞Locat 2:I	
D ⊞-Locat 2:1	-
Add	
Tags To Trend:	
Remove	
Cancel (Back Mext) Finish	Holp

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

This will allow the trend to monitor the input from AI0

v Trend - Add/Configure Tags Scope:		X			
Intro_Lab_Control_Project					
AvailableTags:					
Name					
T-Lucal.2.C		-			
🖞 🖃-Local: 2:1					
Local:2:1.Fault					
Local:2:1.Ch1 Data					
Ac	id 🔪				
Tags To Trend:	1,3				
Local:2:1.Ch0Data					
I					
Rem	cve				
Cancel < Back Ni	ext> Finish Help				
Cancer C Back IN	Exc.y I must Help				

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:

New Trend - Add/Contigure Tags
🕼 Intro_Lab_Control_Projec: 🖉 🚽
👘 Iniro Lab Control Project
🚔 Main ^o rogram
l] ⊞Local:1:C
land a statistic and a statis

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

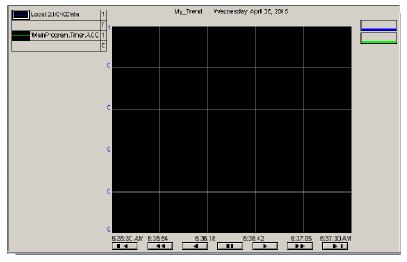
New Trend – Add/Configure Tags	×
Scope:	
🕞 MainProgram	
AvailableT ags:	
Name 📰]
Motor_Run	
Motor_Start	
Motor_Stop	
⊞-Timer	
	1
Add	
Tags To Trend:	

- 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.

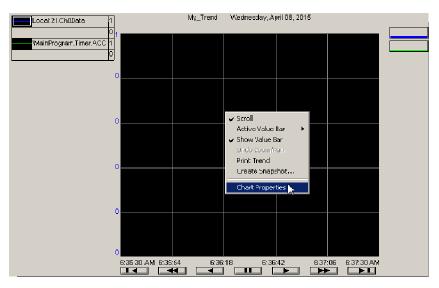
w Trend - Add/Configure Tags	×
Scope:	
🕞 Main Program 💌	
AvailableTags:	
Name	
Timer.PRE	
-Timer.ACC	<u> </u>
Timer.EN	
Timer.TT	+
Add	
Tags To Trend:	
Local:2:1.Ch0Data VMainProgram.Timer.ACC	_
Remove	
Cancel < Back Next > Finish	Help

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.

RSTrendX Properties	×
Name General Display Pels X-Axis Y-xis Template Sampling Start Trigger Stop Trigger	
Chart time range Start date 9/25/2013 Start Date and Start Time are not available when scrolling is allowed. To clear Allow Scrolling, use the Display tab Time span 2 Minute(s) Display optic Hour(s) Display optic Hour(s) Display grid lines 4 Minor grid lines 0 Minor grid lines 0 Minor grid lines 0 Minor grid lines	
OK Cancel Apply Help	

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.

et 1 Stop Errors Log 🗸 Logging Periodic 10 ms Capture: 1:05:20 PM Ny_Trend Nonday, April 06, 2015 local: 2:1.Ch0Data 30,208 18176 ainProgram.Limer.Ai 32,76 32767 24,166 18,124 12,083 6,041 1:05:32 1:05:08 1:04:20 PM 1:04:44 1:06:20 PM 1:05:53 Г _ Г

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

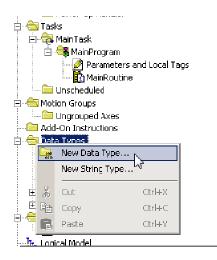
In this section of the lab you will create a custom User Defined Type (UDT).

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.

🚟 Data Type:	New UDT23*				<u>_ ×</u>
Name:	Gallons_to_Liters		Data Type Size: ??	Properties	→ ┦
			bata rypo bizor ···	Extended Propertie	s 🔻
Description:		Holds gallons and the equivalent in $-$		🖂 General	
				Data Type Size	?7
				Description	Holds gallons an …
Members:				Name	Gallons_to_Liters
Name	Data Type d Member	Description			
* 40					
	1				
			*		
		OK Cancel App	ly Help		

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

Memb	ers:					
	Name		Data Type	Description		L
	Gallons	Т			<u>^</u>	E
						L
						L
						L

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

Me	mbe	rs:							External A
	4	Name	Data Ty	oe 🕇	Description				Name
>	*	Gallons	REAL	Ι			-	Г	
	Г	💥 Add Member						L	
								L	

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

Me	embe	ers:				E:
	4	Name	Data Type	Description		N
3	*	Gallons	DEAL	×	L	SI
3	*	Liters	REAL		Г	
		💥 Add Member		2	L	
					L	

7. Click Apply.

The window should appear as shown.

🛗 Data Type: Gallons_to_Liters											
Name:	lame: Gallons_to_Liters		Data Type Size: 8 bytes			Properties					
								E×	tended Propertie:	5	•
Descrip	otion:			Holds g	gallons and				General		
					quivalent Liters				Data Type	REAL	
	1								Description		
Membe	ers:								External Access	Read/Write	•
- 4	Name		Data Type	Description					Name	Liters	
	Gallon	s	REAL				<u></u>		Style	Float	•
	Liters		REAL					Г			
	<mark>∦</mark> Aa	dd Member									
								F			
							v				
				OK	Cancel	Apply	Help				

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.

Controller Organizer	S	cope: 🕞 MainProg	am 💌 Si	how: All Tags		•	Y. Enter Name Fi
Controller Tags		Name 🔤 🛆	Usage	Value 🔸	Force Mask 💦 🔦	Style	Data Type
الله المعالم ال المعالم المعالم		Motor_Run	Local	0		Decimal	BOOL
Power-up Handler		Motor_Start	Local	0		Decimal	BOOL
🗇 🖶 Tasks		Motor_Stop	Local	0		Decimal	BOOL
ian - Carl Main Task ian Program		+ -Timer	Local	{}	{}		TIMER
Add-On Instructions							

- 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.

Scope: 🕞 MainProgr	am 💌 S	how: All Tags			▼ 7.	Enter Name Filter			
Name == 🛆	Usage	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style	
Motor_Run	Local	Local:0:0.Data.0(C)	Local:0:0.Data.0(C)	BOOL		Read/Write		Decimal	
Motor_Start	Local	Local:2:1.Data.0(C)	Local:2:1.Data.0(C)	BOOL		Read/Write		Decimal	
Motor_Stop	Local	Local:2:1.Data.1(C)	Local:2:1.Data.1(C)	BOOL		Read/Write		Decimal	
P Time	Local			TIMER		Read/Write	Г		
Gallons_to_Liters	Logal			Gallons_to_Liters 💀		Read/Write		Decimal	
			Select Data Type	e		×			
			Data Types:		*0				
			Gallons_to_Liters	s		ок			
				24.00					
			FILTER_HIGH_F		-	Cancel			
			FILTER NOTCH			Help			
			FIVE_POS_MOD	E_SELECTOR					
			FLIP_FLOP_D						
			FLIP_FLOP_JK						
		(Gallons_to_Liters	FRATUR					
									
			- Array Dimension	าร					
			Dim 2	Dim 1	Dim 0				
			0 🗧	0 🚊	0 🗦				
									_
			_ 🗌 Show Data T	ypes by Groups					
▲ ► \ Monitor Tags	∖Edit Tags /								

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

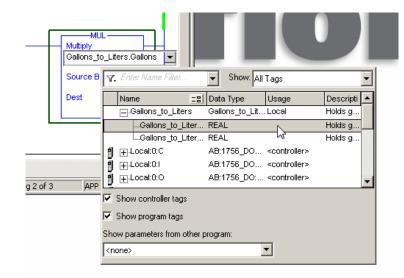
14. Double click on the *MainRoutine*.

Rem Run Run Mode No Forces Controller OK No Edits Energy Storage OK	Path: AB_ETHIP-1\192.1681.7* S H </th <th></th>	
Controller Organizer 4 X	Image: Second State Motor_State Motor_State	TON- Timer On Delay Timer Timer Preset 32767 (DN)- Accum 0 (DN)- Move Source Locat21.Ch0Data 256 (Dest Locat2.0.Ch0Data 256 (Dest Locat2.0.Ch0Data 256 (Dest Locat2.0.Ch0Data)

- 15. Make sure the **End rung** is highlighted. Click on the **H** 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**.

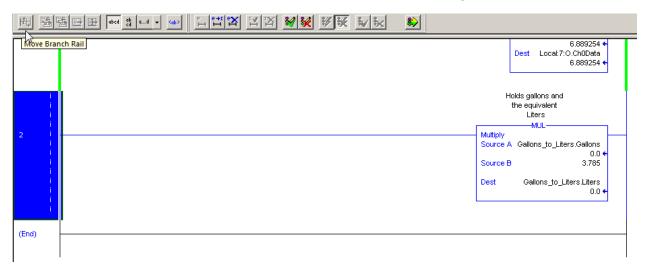
n 1	Motor_Start <locat111data.1> Motor_Run <locat110data.2> Motor_Run <locat110data.2></locat110data.2></locat110data.2></locat111data.1>	TON Timer On Delay Timer Timer Timer Preset 0 ← (DN) Accum 0 ← MOV Move Source Local:2:1.Ch0Data 0.0 ← Dest Lccal:2:0.Ch0Data 0.0 ←
2 <table-cell></table-cell>		 MUL Source A 7 Source B 7 Dest 7 7?

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



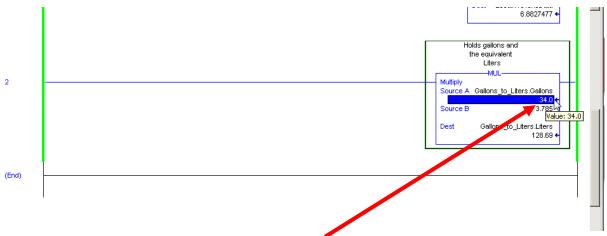
Note: the Gallons_to_Liters tag will need to be expanded to select the Gallons member.

- 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.

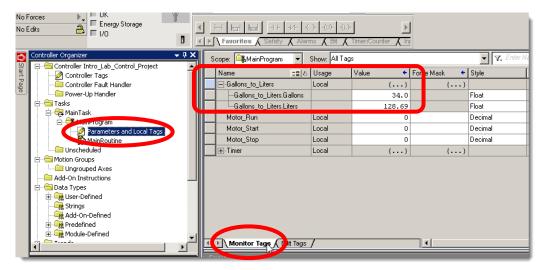
Notice that the values of the tags are shown on the instruction. The multiply instruction converts the number in gallons to liters.



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

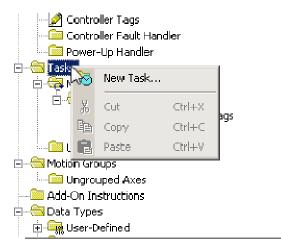
This lab section should take roughly 15 minutes to complete.

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.

New Task			X
Name:			UK.
Description		<u>_</u>	Cancel
		<u></u>	Həlp
Туре:	Periodic	•	
Period	10.000	ms	
Priority:	10 🗄	Cower Number Yields Highe	r Priority)
Watendog	500.000	ns	
🔲 Disable Auton	natio Oulput Pr	ocessing To Reduce Task Cy	/erhead
🔲 Inhibit Fask			

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

New Task	2
Name:	My_Periodic_Task_5Cm; IK
Description.	Cancel
Type:	Ferudic
Period	53.000 ms
Priority:	10 📑 (Lower Number Yields Higher Frierity)
Walchdog:	500.000 III×
🔲 Disable Aut	on ato Dutput Processing To Reduce Task Overhead
🔲 nhibit Task	

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. Continuous_Task 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. Periodic_Task There can be multiple periodic tasks.

Event – Triggers on specific events. Event_Task 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....

	- 🔂 MainR	eters and Local Ta	gs			
		Add		Þ	<u> </u>	New Program
E	🗉 😑 Motion Groups	Cut	⊂trl+X		.	New Equipment Philise
	🔄 🛄 Ungrouped 📑		Ctrl+C			Import Program
F			Ctrl+V			Import Equipment Phase
	User-Defin	Paste Special		•		
		Delete	Del			

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*.

New Program		X
Name:	My_Push_Buttons_Program	ОК
Description:	A	Cancel
	•	Help
Parent:	<none></none>	
🔲 Use as Folder		
Schedule In:	My_Periodic_Task_50ms	
🔲 Inhibit Progra	m	
C Open Properties		

- 6. Expand the *My_Periodic_Task_50ms* folder.
- 7. Right click on the *My_Push_Buttons_Program* folder and select *Add* and *New Routine*.

Tasks	-				
⊕		Add	۱.		New Routine
🚍 🔄 Motion Groups	ж	Cut	Ctrl+X		New Local Tag Ctrl+W
Ungrouped Axes		Сору	Ctrl+C		New <u>P</u> arameter
	ß	Paste	Ctrl+V		Import Routine
🕀 🚂 User-Defined		Delete	Del	_	
		Verify			
		Cross Reference	Ctrl+E		
TT= Controller Organizer TT- Logical Organi		Browse Logic	Ctrl+L	лк	esuits #" watch

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.

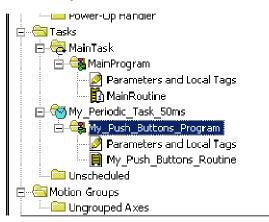
8. Fill in the window as shown.

Name –	'My_Push_Buttons_Routine'
Description –	<pre>'<any description.="" desired="">'</any></pre>
Туре –	Ladder Diagram
In Program or Phase	-My_Push_Buttons_Program.
Assignment –	None.

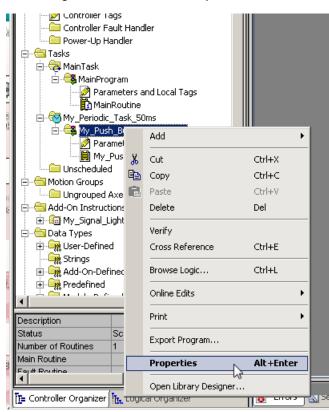
New Routine			×
Name:	My_Push_Buttons_Routine		OK
Description:		<u>▲</u>	Cancel
Туре:	🗎 Ladder Diagram	-	Help
In Program or Phase:	➡ My_Push_Buttons_Program	-	
	Assignment: <a>knone>	-	
🗖 Open Rou	uline		

9. Click OK.

The controller organizer should look like the following.



10. Right click on My_Push_Buttons_Program folder and select Properties.



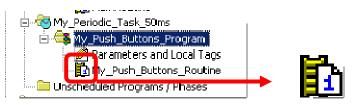
- 11. In the Program Properties window select the Configuration Tab.
- 12. For Main, select My_Push_Buttons_Routine as shown.

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

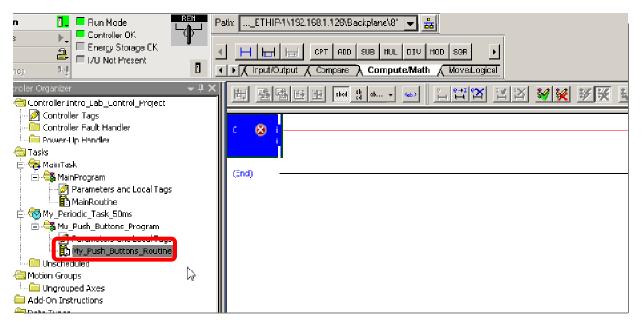
💰 Program Properties - My_Push_Buttons_J	Program			
General Confguration* Parameters Monitor				
Assigned Routines:				
Main: 📕 My_Push_Buttons_Routine	-]		
Fault: krone>	-	I		
🔲 Innibit Program				
	ПК	Cancel	Apply	Help

13. Click OK.

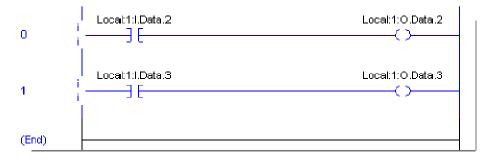
The controller organizer should look like the following with an "I" indicating the main routine. There is only one main routine for each program. Only the main routine runs by default. If no routine is select as the main routine, then no routines will execute. JSR (Jump To Subroutine) instructions are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



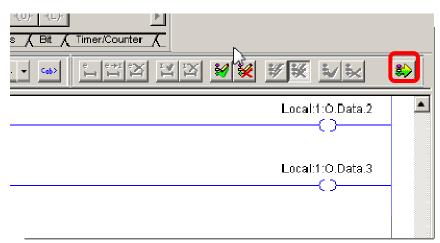
14. Double click on the My_Push_Buttons_Routine to open the routine. It should look like the following.



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.

No Forces	Image: Second	_
Controller Organizer		NA NK NK N
Controller Intro_Lab_Control_Project	0 Local:1:I.Data.2 [Local:1:0.Data:2
ia - 🤕 MainTask ia:- 🚭 MainProgram - Ø Parameters and Local Tags - ∰ MainRoutine	1 [] []	Local:1:0.Data.3
My_Periodic_Task_50ms My_Push_Buttons_Program Parameters and Local Tags My_Push_Buttons_Routine My_Push_Buttons_Routine	(End)	

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)

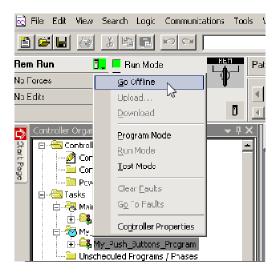
This lab section should take roughly 15 minutes to complete.

Objective:

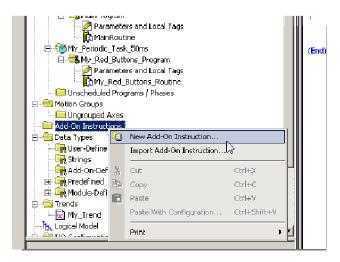
In this lab, an AOI instruction will be added to the project. AOI's cannot be created or modified online (although they can be imported online). The AOI will be created offline, code added, and downloaded to the controller to see it execute.

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'.*

Name:	My_Signal_Ligh.		ОК
Cescription:			laneel Help
Туро:	📕 Ladder Jiaqram	_	
Fevision:	Major Minor Extended Text		
Fevision No	6	×	
Vendor			

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.

Add- ener	On Instruction Do			i ght v1. 0 Signature Dhange	History 🗍 Help			
	Name	Usage	Data Type	Alia: For	Default	Style	Rec V	_
	Erablein ErableOut	Input Output	BOOL		0	Decimal Decimal		Er Er
Ø]						
al								
•				1				Þ
■ Mi	oye Up 🔰 Move G	own		1				<u> </u>
			re and local ter	s whose values ware	modified to all	ture of this	instruction	. F

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

6	a Add-On Instruction Definition - My_Signal_Light v1.0						
	General Parameters* Local Tags Sican Modes Signature Change History Help						
	Name U	lsage Da	ata Type 🛛 Alia	as For	Default	Style I	Reg Vir D
	Enabeln In	nput BC	DOL		1	Decima	
	EnabeOut 0	u:put DC	DOL		0	Decima	
	My_Signal_Input In	nput Rf	וחו		N		<u> </u>
		u:put BC	DOL		0	Decima	<u> </u>
						-	
_							
*	My_Signal_Inpu	ıt	Input	BOO	L		
*	My_Signal_Out	put	Output	BOO	L		
<u></u>							
	4						F
	Move Up Move Down						
	Europy of doits it values of param	neters and lo	ocal tags whose \	values were m	nodifier <mark>to al</mark> I	ags of this in	tructor type
	Logo Data Type Sizer ?	22 hyte (s)	ОК		ncel	Apply	Help

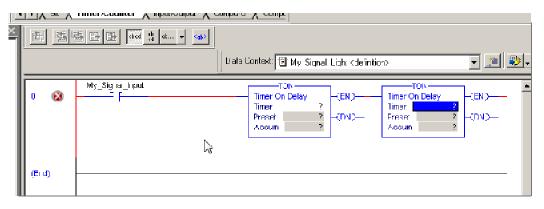
NOTE: Make sure to change the usage on My_Signal_Output to "Output"

- 7. Click Apply.
- 8. Click on the *Logic Button*.

The following Logic window appears.

ф X	問題	🖀 📴 🏥 🐽 😴 🖦 🚽 🖾 Data Context: 🖼 My_Signal_Light (definition)	•
	0 😣		-
	(End)		-

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

Data Context: 🖼 My_Signal_Light <definition></definition>				
0 🐼	My_Signal_Input	Accum	CEN Timer On Delay Timer Image: New Local Tag 'Timer' New Parameter 'Timer' New Parameter 'Timer' Cut Instruction Cut Instruction Copy Instruction	Ctrl+X Ctrl+C
(End)		1	Baste Delete Instruction	Ctrl+V Del
			<u>A</u> dd Ladder Element Edit Main Operand Descriptio	Alt+Ins

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

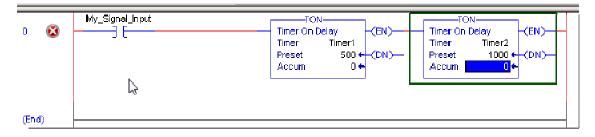
New Add-On I	nstruction Parameter or Local Tag	×
Name:	Timer1	ОК
Description:		Cancel
		Help
Usage:	Local Tag	
Туре:	Base	
Alias For:	<u> </u>	
Data Type:	TIMER	
Instruction:	🕞 My_Signal_Light	
External Access:	None	
Style:	<u> </u>	
E Required	🔲 Visible	
🔲 Constant		
🗌 Open Con	figuration	

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

0	8	My_Signal_Input	Timer On	DN Delav	-(EN)	Timer Or	on Delav	
			Timer Preset	Timer1 500 ←	-(DN)	Timer Preset	?	
			Accum	• 0		Accum	?	J

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.

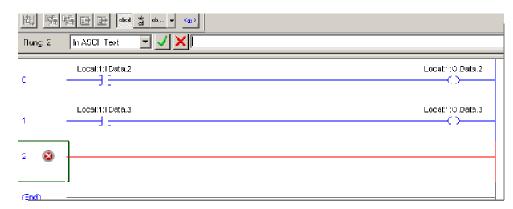
0 😵	My_Signal_hput Timer2.DN	TON Timer On Delay Timer Timer1 Preset 500 € Accum 0 €	-TON- Timer On Delay Timer Timer2 Preset 1000 Accum 0	←(DN)—
(End)				

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

0	My_Signal_Input Timer2.DN	-ToN Timer On Delay Timer Timer1 Preset 500 ← Accum 0 ←	-(EN)		2 (EN)- 0 (CDN)- 0
1	Timer1.DN			My_Si	gnal_Output -< >

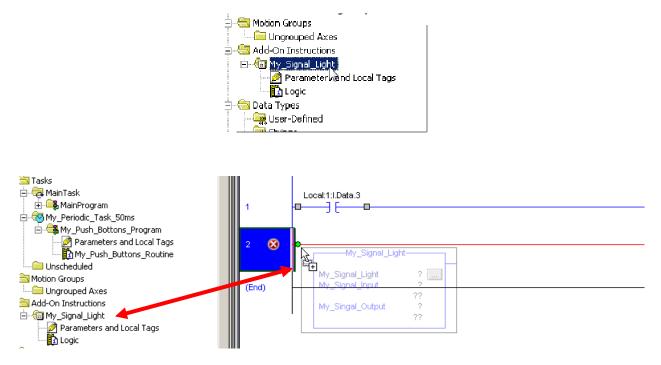
We have finished creating the AOI definition and logic! We will now add the AOI into the program to have it flash a light.

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"*

		Local:1:O.Data.3	
My_Signal_ My_Signal_Light My_Sig My_Signal_Input		New "My_Signal_Light_Tag"	Ctrl+W
My_Singal_Output	ж Ва	Cu <u>t</u> Instruction <u>C</u> opy Instruction <u>P</u> aste	Ctrl+X Ctrl+C Ctrl+V
		Delete Testuretien	D-1

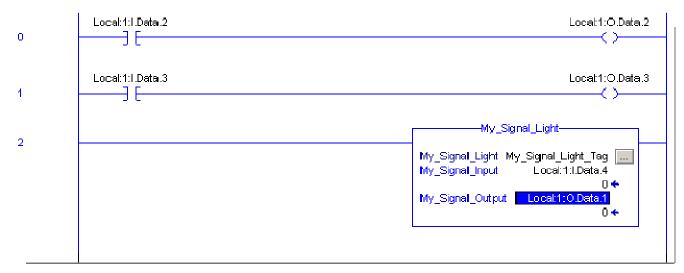
22. Click Create to create a new local tag.

New Parame	ler or Tay	×
Name:	My_Signal_Light_Tag	Lieate ▼
Description:	×	Cancel Help
Usage:	Local Tag	
Туре:	Base Connection	
Alias For	T	
Data Type:	My_Signal_Light	
Parameter Connecton:		
Scope:	🕞 My_Fush_Buttors_Program	
External	Read/Wite	

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.

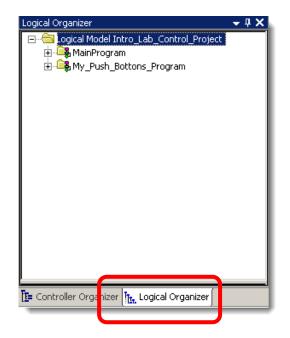
Objective:

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

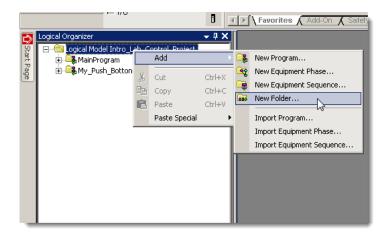
1. Click on the *Logical Organizer tab*

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

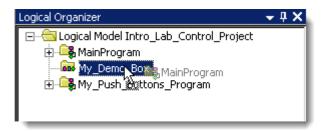
Notice that we can also add programs here as well as in the Controller Organizer. If we add a program here, the configure window will allow us to pick which Controller organizer task the program is scheduled in.



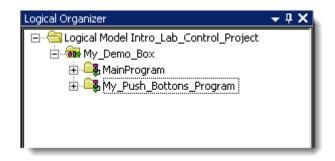
3. Enter the name '*My_Demo_Box*' and click *OK*.

New Program		×			
Name:	My_Demo_Box	ОК			
Description:		Cancel			
		Help			
Parent:	<none></none>	•			
🔽 Use as Folder					
Schedule In:	<none></none>	v			
🔲 Inhibit Progr	am				
Open Properties					

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

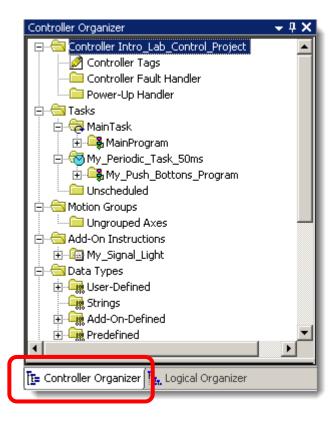


5. Do the same for *My_Push_Buttons_Program* to associate with *My_Demo_Box*.



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:

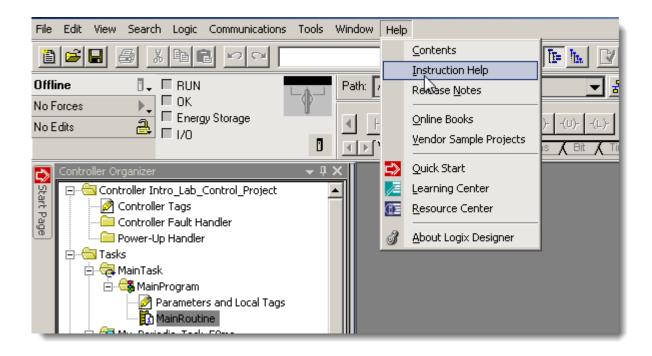
In this lab we will explore the extensive online Help system in Studio 5000. Feel free to look and poke around as desired.

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.

🐕 Logix Designer Online Help		
Hide Back Print <u>O</u> ptions		
	Instruction Set For information about an instruction, click the category. Ock the <u>alphabetical listing</u> to find more information on a specific instruction. Programming languages are designated as follows. Ladder Diagram Logic - Function Block Sequential Function Chart Structured Text Note that for more information on Structured Text programming syntax, refer to <u>Structured Text Syntax</u> . The following list contains information about the various instruction categories: Instruction Supported Instructions by Language	Print
		T

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.

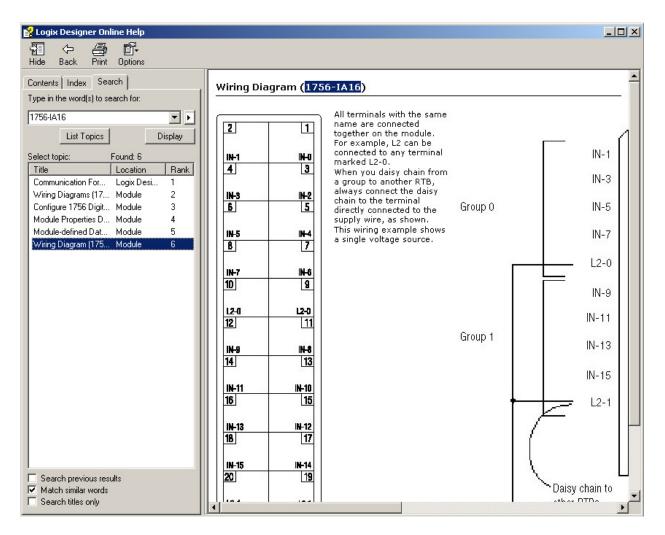
The instruction help can also be accessed alphabetically as well.

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.

💕 Logix Designer Online Help	
To the stack Print Options	
Contents Index Search Type in the word(s) to search for: 1756-IA16 List Topics Display Select topic: Found: 6 Title Location Rank Communication For Logix Desi Wiring Diagrams (17 Module A Module Properties D Module Module -defined Dat Module Wiring Diagram (175 Module Module Hord Properties Wiring Diagram (175 Module Comparison (17 Module M	Print Quick Start Note that this Quick Start is intended to show you how to use the most basic features of the Logix Designer application to get up and running. For purposes of example, we take you through the steps you follow to create a project using ladder diagram logic. These steps are shown only as an example; you should note that there are many more features of this product that are not illustrated here. Refer to the online help for those specific features for more detailed information. See also Creating a project Creating and configuring I/O Entering tags and aliases Entering ladder logic Downloading Monitoring tags Monitoring logic
 Search previous results Match similar words Search titles only 	Top of Page Print Last revised: Tuesday, August 28, 2012 09:07 @2012 Rockwell Automation Technologies, Inc. All rights reserved.

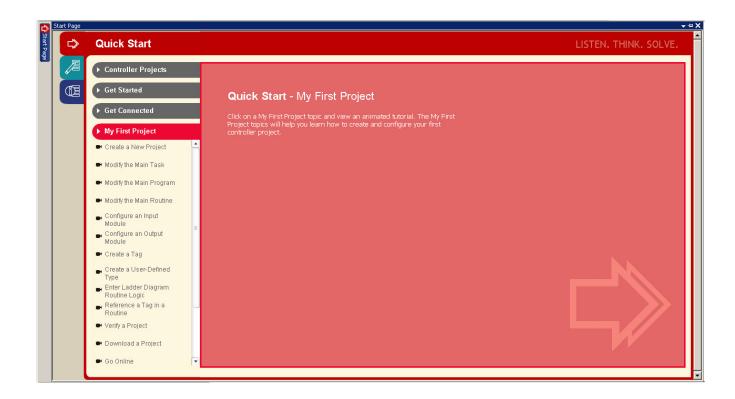
7. Click *Display* to view the wiring diagram for this module. Note you may need to maximize your screen.



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

Using Start Pages

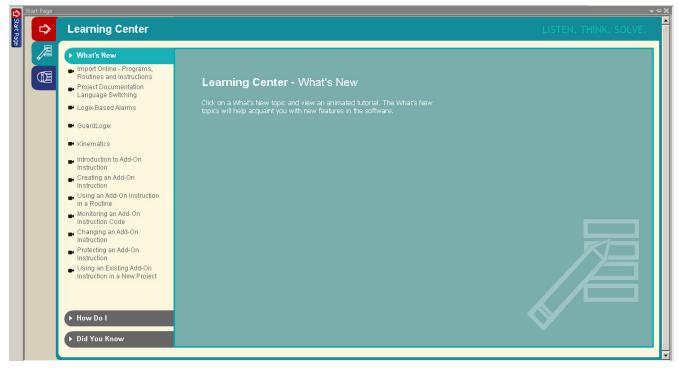
9. From the *Help* pull down menu select *Quick Start* which is one of the three tabs available from the **Start Page**.



- 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

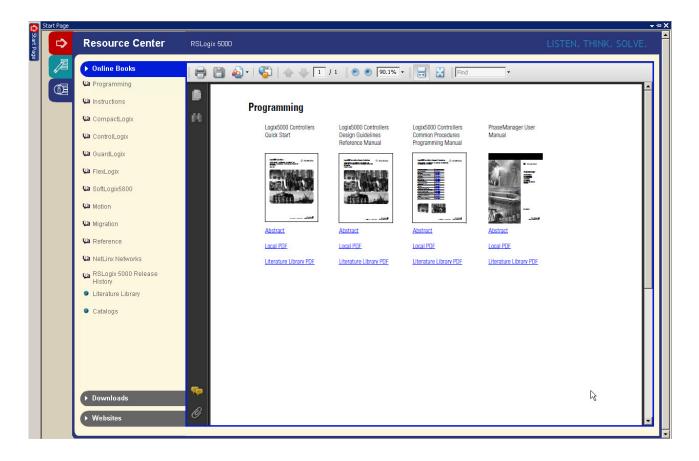
 Targets customers wanting to learn or explore how to use the software beyond just getting started reduces learning curve and helps increase productivity.



- 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