Introduction to Studio 5000 Logix Designer™



For Classroom Use Only!



Allen-Bradley • Rockwell Software



Publication Number -- Date

Copyright © 2013 Rockwell Automation, Inc.

Important User Information

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

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

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

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

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

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

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

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



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

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



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

BURN HAZARD



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

Introduction to Studio 5000 Logix Designer™

Contents

Before you begin	.5
What You Will Accomplish In This Lab	. 5
Who Should Complete This Lab	. 5
Lab Materials Hardware	.6 6
Software	. 6
Files	. 6
Document Conventions	.7
About Logix Controllers	.8 8
CompactLogix: Perfect for smaller, machine-level control applications	. 8
About Studio 5000 What's new?	. 8 9
Logix Designer	. 9
View Designer	. 9
Section 1: Creating a Project	10 10
Launching Studio 5000 Configuration Software	10
Creating a New Controller Project	11
Adding Ladder Logic to the Main Routine	15
Creating Tags for the Ladder Code	22
Monitoring/Editing Tags	30
Section 2: Connecting Your Computer to the Controller	35 35
Launching RSLinx Software	35
Adding the AB_ETHIP-1 (Ethernet/IP) Driver	35
Section 3: Downloading the Project from the Computer to the Controller	39 39
Downloading the Project to the Controller	40

Section 4: Configuring I/O	44 ۸۸
Adding Controll ogix I/O Manually	
Adding ControlLogix I/O Using "Module Discovery"	
Viewing the ControlLogix I/O Tags	
Assigning Alias Tags	
Section 5: Testing Your Logic Program	
Switching the Controller into Run Mode and Testing the Program	63
Section 6: Adding Logic and Tags Online	67 67
Adding a MOV Instruction to the Logic	67
Adding the Timer to the Logic	70
Testing Your Logic	75
Section 7: Creating and Running a Trend Objective:	
Creating and Running a Trend	
Section 8: (Optional) Creating and Using User Defined Types (UDT)	
Creating User Defined Types	
Add the UDT tag to an instruction	
Monitoring UDT Tags	89
Section 9: (Optional) Using Studio 5000 Help	
Objective:	
Viewing I/O Module Wiring Diagrams	
Using Start Pages	
Learning Center Tab	
Resource Center Tab	

Before you begin

About This Lab

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

What You Will Accomplish In This Lab

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

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

Who Should Complete This Lab

This hands-on lab is intended for:

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

Lab Materials

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

Hardware

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



2. CompactLogix demobox (either L43 or L35E)



Software

This hands-on lab uses the following software:

- Studio 5000 programming software
- RSLinx Classic software

Files

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

6 of 98

Document Conventions

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

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

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

About Logix Controllers

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

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

ControlLogix offers the following benefits:

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

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

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

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

CompactLogix: Perfect for smaller, machine-level control applications

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

CompactLogix offers the following benefits:

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

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

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

About Studio 5000

8 of 98

What's new?

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

Logix Designer

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

View Designer

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

Section 1: Creating a Project

This lab section should take roughly 20 minutes to complete.

Objective:

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

Launching Studio 5000 Configuration Software

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

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



The Studio 5000 Splash Screen appears.



FYI

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

Creating a New Controller Project

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

1. Select New Project under the Create section.



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

📀 New Proje	ct					<u>?</u> ×
Logix		1756-L75				×
		1756-L75 Contro	olLogix® 5570 Controlle	r		
Name:						
Location:	C:\User	s\Labuser\Docume	nts		•	Browse
			Cancel	Back	Next	Finish

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

3. Type 'Controller1' into the name field.

Name:	Controller1				
Location:	C:\Users\Labuser\Documents			•	Browse
		Cancel	Back	Next	Finish

4. Press the Next button.

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

New Project						? X
Project Configuration Controller1 (1756-L75 ControlLogix® 5570 Controller)						
Revision:	22 💌					
Chassis:	1756-A10	10-Slot Co	ntrolLogix Chassis			•
Slot:	1 🔻					_
Security Authority:	No Protection	e selected S	Security Authority	for authenticatior	and authorization	▼ n
Description:	Introduction	to Logix Cor	ntrollers			
🗌 Enable redundar	ncy					
			Cancel	Back	Next	Finish

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

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

FYI

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

<u>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.

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

Name: The name of the controller and project.

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

<u>Stot</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.

💰 Logix Designer - Controller1 [1756-L75 22.1] Logic Communications Tools Window Help Search - **44 6 6 6 8 9** - 🥪 Select language Offline . ■ RUN Path: <none> - * □ ок No Forces 1 Energy Storage 4 HHH Þ No Edits Controller ٥ TFAvorites K E - 4 × Faceplate oller Controller1 Controller Tags Controller Fault Handler Power-Up Handle Instructions E G Tasks toolbar AlinTask Motion Groups Add-On Instructions Data Types E Grings + Predefined 🤹 Module-Defined Controller Data Logs I/O Organizer Trends I/O Configuration Configuration - 1 [1] 1756-L75 Controller1 Programming window Controller Organizer and 1756-L75 ControlLogix® 5570 Control Logical Organizer Description Introduction to Logix controllers 1 T= Controller Organizer

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

You have now created your first controller project!

13 of 98

FYI

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

-Controller File Name

-Tasks

-Motion Groups

-Add-On Instructions

-Data Types

-Trends

-I/O Configuration*

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

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

Adding Ladder Logic to the Main Routine

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

You will continue to use the project already opened.

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



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

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



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



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

5. Verify your rung appears like the figure below:



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



- Drag the XIO onto rung O to the right of the XIC instruction as shown above. Again a green dot will appear to the right of the XIC instruction indicating where your new instruction will be inserted. Release the mouse button at the location you wish to place your instruction.
- 8. Verify your rung appears like the figure below:



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.

- + +Favorites Add-On Alarms Bit Timer/ Ē ab... 👻 abcd ab e 7 e 0 ЭF e e (End)
- 9. From the instruction toolbar, left click and hold on the **Output Energize (OTE)** () instruction.

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



FYI

We will now add a branch around the XIC instruction.

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



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



- 14. Left-click and hold on the **blue highlighted part of the branch** and drag your selected leg of the branch to the left side of the XIC instruction.
- 15. Place the branch over the green dot and release the mouse button.



- 16. From the instruction toolbar, left click and hold on the **XIC** + instruction.
- Drag the XIC onto your newly created branch until the green dot appears. The rung should now appear as shown below.



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

		equencer <u>(</u> E
間層		
0		?
(End)		

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

FYI	
ΓYI	

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

Creating Tags for the Ladder Code

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

FYI

What is a tag and why are they better?

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

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

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

	bcd ab ab ▼ <ab></ab>	
	2 NewnTag	
	Cut Instruction	Ctrl+X Ctrl+C Ctrl+V
(End)	Delete Instruction Add Ladder Element Edit Main Operand Description	Del Alt+Ins Ctrl+D
	Save Instruction Defaults Clear Instruction Defaults	
	Toggle Bit	Ctrl+T
	Eorce On	
	Force Off	
	<u>G</u> o To	Ctrl+G

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

New Program	n Parameter or Tag	×
Name:	2	Create 🛛 💌
Description:	<u> </u>	Cancel
		Help
Usage:	Local Tag	
Parameter Connection:		
Type:	Base Connection	
Alias For:	<u> </u>	
Data Type:	BOOL	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	Decimal	
Constant		
🗖 Sequencin	ig	
C Open Con	figuration	
C Open Para	ameter Connections	

FYI

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

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

Type: Defines how the tag operates within the project

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

Alias: A tag that represents another tag

Produced: Send data to another controller

Consumed: Receive data from another controller

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

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

<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.

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

Read Only: External application can only read the tag.

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

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

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

23 of 98

2. Enter the tag fields as shown below.

Make sure the scope of the tag is MainProgram.

New Program	Parameter or Tag	×
Name:	Motor_Start	Create 💌
Description:		Cancel
		Help
Usage:	Local Tag	
Parameter Connection:	<u> </u>	
Type:	Base Connection	
Alias For:		
Data Type:	BOOL	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	Decimal	
Constant		
E Sequencin	g	
C Open Coni	figuration	
🔲 Open Para	meter Connections	

 Click Create to accept and create the tag. The rung will now look like the figure below.



Again, the New Tag window will appear:

5. Enter the fields as shown below:

New Program	n Parameter or Tag	×
Name:	Motor_Stop	Create 🔻
Description:	A	Cancel
		Help
Usage:	Local Tag	
Parameter Connection:		
Type:	Base Connection	
Alias For:	<u> </u>	
Data Type:	BOOL	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	Decimal	
Constant		
E Sequencin	Ig	
C Open Con	figuration	
Den Para	ameter Connections	

- 6. Click *Create* to accept and create the tag.
- 7. Verify the rung appears like the figure below:

	Here Add-On Alarma (Bt (TimerCounter (InnutDutrut (Compare (ComputeMeth (Moved opical (ElleMisc (ElleShift (Sequencer (E
田盛	
e 0 e e e e e (End)	Motor_Start Motor_Stop
	You will now create the tag Motor Run
8.	Right click on the ? of the OTE instruction and select New Tag.

The New Tag window will appear.

9. Enter the fields as shown below:

New Program	Parameter or Tag	×
Name:	Motor_Run	Create 🛛 🔻
Description:		Cancel
		Help
	T	
Usage:		
Parameter Connection:		
Type:	Base Connection	
Alias For:	T	
Data Type:	BOOL	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	Decimal	
Constant		
E Sequencin	g	
C Open Con	figuration	
🔲 Open Para	ameter Connections	

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

•	⊢ ▶ \Fau	Image:	
F			
0	e e e e e	Motor_Start Motor_Stop Motor_Run	<u> </u>
(E	ind)	k₂	

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



- 11. Left click and hold the mouse button over the tag *Motor_Run* on the *OTE* instruction.
- 12. Drag the tag *Motor_Run* over to the *XIC* instruction until a green dot appears next to the **?** then release the mouse button.

(H H						
間醫							
0 e e e e e	Motor_Start Motor_Stop Motor_Run	•					
(End)	\						

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



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

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

💰 Logix Desig	jner	- Controller1 [1756	-L75 22.	1]* - [Ma	ainProgram -
📕 File Edit	Viev	, Search Logic Co	mmunicati	ons Too	ls Window
1		<u>T</u> oolbars			
Rem Run	€	Zoom <u>I</u> n		REM	Path: 🗸
No Forces	Q	Zoom <u>O</u> ut			
No Edits	Ē.	<u>C</u> ontroller Organizer	Alt+0		
Redundancy		<u>E</u> rrors	Alt+1		I → \ Fav
Controller		<u>S</u> earch Results ^し	Alt+2	· 中 🗙	
। <mark>२</mark> ि⊖्य त		<u>W</u> atch	Alt+3		
P P	h.	Logical Organizer	Alt+4		
		Start <u>P</u> age	Alt+9		
	isks	- I			Ů

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

ø Logix Designer - Controller1 [1756-L75 22.1]* - [N ■ File Edit View Search Logic Communications To	fainProgram - MainRoutine]	
	- <u>*</u> ** * * * * *	Select language
Offline RUN No Forces Charge Controller Organizer Charge RUN Reduits Renergy Storage RUN Reduits RUN Reduits RUN Reduits RUN Reduits RUN Reduits RUN Reduits RUN RUN RUN RUN RUN RUN RUN RUN	Patr:	
Controller Tags Controller Fault Handler Power-Up Handler Tasks Tasks Tasks MainProgram Ma	0 Motor_Start Motor_Stop (End)	Motor_Run
Trends	MainRoutine	×.
🗄 🚍 1756 Backplane, 1756-A10 └──10 [1] 1756-L75 Controller1	Errors Verifying routine 'MainRoutine' of program 'MainProgram' Verifying program connections Verifying Overall Data Log Storage	→ 1 ×
Type Ladder Diagram (Main)	Verifying Overall Data Log Data Processing Bandwidth Complete - 0 error(s), 0 warning(s)	×
The Controller Organizer The Logical Organizer Verify complete with no errors or warnings.	🐱 Errors ∫ 🗟 Search Results 🔛 Watch	Rung 0 of 1 APP VER

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



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

New starting with Version 21 - Extended Tag Properties

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

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

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

Monitoring/Editing Tags

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

You will continue to use the project already opened.

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



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



FYI

Monitor/Edit Tags Tabs

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

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

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

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

Path: <none></none>							
						▶	
Favorites Add-On Alarms Bit Time	r/Counter 👗 Input/Output 🗼 Company	e 🗼 Compute/Math 🗼 M	love/Logical	🕻 File/Misc. 🔏 File/S	Shift 🖌 Sequencer	ΤĒ	
Scope: 🚺 Controller1 💌 Show: All Tags		▼ 7.	. Enter Name F.	iller			-
Name II Alias For Base Tag	Data Type Description	External Access	Constant	Style			
							^o roperties
Notice a field in the upper left corner of the about Controller and Program scoped tag tags.	ne Tag Editor window labe gs. Currently the selection	eled Scope Earl n is Controller1,	lier in the which are	lab we talked e controller sc	briefly oped		

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).



FYI

- 2. Click on the down arrow for the Scope selection box.
- 3. Select *Programs* → *MainProgram*



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

S	Scope: 🖓 MainProgram 🔻 Show: All Tags 🖉 🏹 Enter Name Filter										
lη	Name 📰 🛆	Usage	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style	<u> </u>	1
۲	Motor_Run	Local	Local:0:0.Data.0(C)	Local:0:0.Data.0(C)	BOOL		Read/Write	Г	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	Г	Decimal		
٨											
											ľ

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

💰 Logix Designer - Controller1 [1756-L75 22.1]* - [Pro	ogram '	Tags - MainPro	gram]								بلواجع	хI
Pile Edit View Search Logic Communications Tool	ls Wind	dow Help									그리:	×
		•	🚜 🖧 🔓		<u>*</u> QQ			Select lang	uage	- 🥪	HH	F
Offline 🛛 🗸 🗖 RUN	Pa	ath: AB_ETHIP-1	1\192.168.1.1	1\Backplane\1 👻	윪							
No Forces					•							
No Edits		H H H	$\neg \vdash \neg \neg \vdash$	-(_)(U)(L)-		▶						
Redundancy D	$\overrightarrow{} \models$	Favorites	Add-On 🔏 /	Alarms 🔏 Data Log	K Bit K Timer/Cou	inter 🕻 Input/Output						
Controller Organizer	Scope	e: 🅞 MainProgra	am 🔻 S	how: All Tags			• Y.	Enter Name Filter			•	Ī
Controller Tags	Na	ame <u>=8</u> A	Usage	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style		7
🖉 Controller Fault Handler		Motor_Run	Local	Local:0:0.Data.0(C)	Local:0:0.Data.0(C)	BOOL		Read/Write	Г	Decimal		P
Power-Up Handler		Motor_Start	Local	Local:2:1.Data.0(C)	Local:2:1.Data.0(C)	BOOL		Read/Write	Г	Decimal	- ope	nope
		Motor_Stop	Local	Local:2:I.Data.1(C)	Local:2:1.Data.1(C)	BOOL		Read/Write		Decimal		rtie
A Main Program	2											"

- 5. **Save** the program by clicking on the Save icon **I** on the toolbar.
- 6. Minimize the Studio 5000 software.

Section 2: Connecting Your Computer to the Controller

This lab section should take roughly 5 minutes to complete.

Objective:

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

- Launch RSLinx Classic communications software
- Configure your communications driver

Launching RSLinx Software

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

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



Adding the AB_ETHIP-1 (Ethernet/IP) Driver

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

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



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

Configure Drivers		<u>? ×</u>
Available Driver Types:	Add New	Close Help
Name and Description	Status	
AB_ETHIP-1 A-B Ethernet RUNNING	Running	Configure
		Startup
		Start
		Stop
		Delete
		45
1		

- From the Available Driver Types pull-down menu, choose EtherNet/IP Driver then click on the Add New button.
- 3. Change the name of the driver from AB_ETHIP-2 to AB_ETHIP-LAB as shown and click OK

Configure Drivers		Add New 1	Close
Configured Drivers:	Add New RSLinx Classic Driver		Help
AB_ETHIP-T A-B Eth	Choose a name for the new driver. (15 characters maximum) AB_ETHIP-LAB	Cancel	Startup
			Stop Delete
4. Choose "Browse Local Subnet" and then the "Intel" network driver as shown below and click OK

Configure driver: AB_ETHIP-1			<u>?</u> ×
EtherNet/IP Settings			
Browse Local Subnet	C Browse Remote Subnet		
Description		IP Address	
Windows Default		10010011	
Intel(R) PRU/1000 MT Network	Connection	192.168.1.1	_

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

FYI

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

6. Click the **RSWho**icon

in the toolbar.





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

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

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

🗞 RSLinx Classic Gateway - [RSWho - 1]			
🚓 File Edit View Communications Station DDE/OPC Security Window Hel)		_ B ×
≝ # \$® ® № <u>₽</u>			
Autobrowse Refresh 🖭 🕅 Browsing - node 192.168.1.11 found	l		
E 🗐 Workstation, WIN7-VM	Address Device Type	Online Name	Status
🕀 💑 Linx Gateways, Ethernet	🖞 19 1756-EN2TR	1756-EN2TR/A	ок
田一番 AB_ETHIP-1, Ethernet			
E 192 168 1 11 1756-EN2TR 1756-EN2TR			
45			
			F
For Help, press F1		NUM 09/25/1	3 05:34 PM //

FYI

RSWho

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

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

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

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

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

This lab section should take roughly 10 minutes to complete.

Objective:

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

You will:

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

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



Downloading the Project to the Controller

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

- 1. Maximize Studio 5000 and your Controller1.ACD project.
- 2. From the Communications menu, choose Who Active.



The Who Active Screen appears.

💰 Who Active	
Autobrowse Refresh	Go Online Upload Download Update Firmware Close Help
Path: <none> Path in Project: <none></none></none>	Set Project Path

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

💰 Who Active	
Autobrowse Refresh	Go Online Upload Download Update Firmware Close Help
Path: AB_ETHIP-LAB\192.168.1.11\Backplane\1 Path in Project: <none></none>	Set Project Path Clear Project Path

FYI

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

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

Download	ł			X
A	Do	wnload offline pro	ject 'Controller1' to the controller.	
		🔽 Download Pi	roject Documentation and Extended Properties	
	Co	nnected Controlle	r:	
		Name:	Controller1	
		Type:	DB_1756-L75/A ControlLogix® 5570 Controller	
		Path:	AB_ETHIP-LAB\192.168.1.11\Backplane\1	
		Serial Number:	006C614E	
		Security:	No Protection	
	⚠	The controller is i Remote Program	n Remote Run mode. The mode will be changed to prior to download.	
	⚠	DANGER: The con out of date follow of nonvolatile me power up or corre	ntroller image stored in nonvolatile memory might be ving the download. Failure to update the contents mory could result in running old logic following a upt memory condition.	1
	≜	DANGER: Unexpe	ected hazardous motion of machinery may occur.	
		Some devices ma not loaded to the	intain independent configuration settings that are edevice during the download of the controller.	
		Verify these devi have been prope mode.	ces (drives, network devices, 3rd party products) rly loaded before placing the controller into run	
		Failure to load pr and unexpected	oper configuration could result in misaligned data equipment operation.	
		Download	Cancel Help	

FYI

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

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



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

 Run Mode Controller OK Energy Storage OK 	
I/O Not Present	٥



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

Section 4: Configuring I/O

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

This lab section should take roughly 20 minutes to complete.

Objective:

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

You will continue to use the project already opened.

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

- <u>1756-IB16D/A</u> Isolated DC Input Module
- <u>1756-OB16D/A</u> Isolated DC Output Module
- <u>1756-IF6I/A</u> Isolated Analog Input Module
- <u>1756-OF6VI/A</u> Isolated Analog Voltage Output Module

Adding ControlLogix I/O Manually

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



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

5elect	Module Type						
Cab	alon I Module Discov	veru Esvorites I					
							1
	E (C) 7 (/	44 4 4 7			-		
	Enter Search Text I	or module Type		Clear Filter:	s		Hide Filters 🛠
		Module Type Catego	ru Filters			Module Type Vendor I	Filters
	Analog	incluie Type edicge			Allen-Bra	dlev	
	Communicatio	n			Hardy Ins	struments, Inc.	
	Controller				Molex Inc	corporated	
	🗹 Digital				Phoenix [Digital Corporation	
				والعري	Dranafi T	ooboologu	
	Catalog Number	Description	Vendor	Categor	rv		
	1756-CFM	Configurable Flo	Allen-Bradley	Special	ty		
	1756-CN2	1756 ControlNet	Allen-Bradley	Commu	nication		
	1756-CN2R	1756 ControlNet	Allen-Bradley	Commu	nication		
	1756-CNB	1756 ControlNet	Allen-Bradley	Commu	nication		
	1756-CNBR	1756 ControlNet	Allen-Bradley	Commu	nication		
	1756-DHRIO	1756 DH+ Bridg	Allen-Bradley	Commu	nication		
	1756-DMA30	1756 SA3000 Dr	Allen-Bradley	Drive			
	1756-DMA31	1756 SA3100 Dr	Allen-Bradley	Drive			•
	127 of 127 Module T	ypes Found					Add to Favorites
ſ	Close on Create					Create	Close Help

FYI

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

3. Locate the 1756-IB16D.

4. Select the 1756-IB16D module and click Create

ID			Clear Filters		Hide Filters 🕱
	Module Type Catego	ry Filters	• V	Module Type Vendor Fil	ters 🔺
☑ Analog			Aller	n-Bradley	
Communicat	ion		🚽 🗹 Adv	anced Micro Controls Inc. (AMCI)	
Controller			Har	dy Instruments, Inc.	
Digital			🗹 Mole	ex Incorporated	
				no Douoloomont Inc (Automotion)	(alua)
Catalog Number	Description	Vendor	Category		
1756-EN2F	1756 10/100 M	Allen-Bradley	Communication	<u>.</u> ו	
1756-IB16	16 Point 10V-31	Allen-Bradley	Digital		
1756-IB16D	16 Point 10V-30	Allen-Bradley	Digital		
1756-IB16I	16 Point 10V-30	Allen-Bradley	Digital		
1756-IB16IF	16 Point 24V Hi	Allen-Bradley	Digital		
1756-IB16ISO	E 16 Channel Isola	Allen-Bradley	Digital		
1756-IB32	32 Point 10V-31	Allen-Bradley	Digital		
	0.0 1 01	Allers Desallers	Consider		

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

Select Major Revis	ion		×
Select major revis being created.	ion for new 175	6-IB16D module	
Major Revision:	2	•	
ОК	Cancel	Help	

FYI

Module Configuration Wizard

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

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

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

New Module	
Type: Vendor: Parent: Name:	1756-IB16D 16 Point 10V-30V DC Diagnostic Input Allen-Bradley Local Demo Slot: 2
Description:	
Comm Format:	Full Diagnostics - Input Data
Revision:	2 Electronic Keying: Compatible Keying
🔽 Open Module	e Properties OK Cancel Help

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

FYI

Comm Format

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

Electronic Keying

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

The following data is read and compared:

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

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

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

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

Disable Keying - No keying used at all.

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



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

🔜 Module Properties Report: Local:2 (1756-IB16D 2.7)	×
General Connection Module Info Configuration Diagnostics Backplane	
Requested Packet Interval (RPI): 20.0 🚔 ms (0.2 - 750.0 ms)	
🗖 Inhibit Module	
Major Fault On Controller If Connection Fails While in Run Mode	
Module Fault	
Status: Running OK Cancel Apply Help	

FYI

Requested Packet Interval (RPI)

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

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

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

In the Controller Organizer, the *I/O Configuration* folder will show the digital input module in Slot 2. It is possible that you may see a yellow triangle over the I/O module (¹). In this instance, it indicates a firmware mismatch. Since we are deleting this module in the next step, do not worry about the I/O Fault.



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

Adding ControlLogix I/O Using "Module Discovery"

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



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

 1756 Backplane, 1756-A10 [] (00) 1756-0B16D/A 			
🖞 [00] 1756-0B16D/A			
	2.3		Create
🖤 🖞 [02] 1756-IB16D/A	2.5		Create
🖞 [04] 1756-0B16D/A	2.3		Create
🗂 🗍 [05] 1756-M08SE	19.6	Note: This module cannot be created or	nline
🖞 [06] 1756-EN2TR	5.5		Create
🗂 🗂 [07] 1756-OF6VI/A	1.9		Create
🛄 [08] 1756-IF6I/A	1.12		Create

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

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

	1				1.1.1
_		Modules	Revision	Additional Information	Action
	al	1756 Backplane, 1756-ATU			Crasta
	U al		2.3		Create
	0	[U2] 1756-IB16D7A	2.5		Create
	0	[U4] 1756-UB16D7A	2.3	• • • • • • • • • • • •	Lreate
	U el	[U5] 1756-MU85E	19.6	I his module cannot be created online	Create
	0	[U6] 1756-EN2TR	5.5		Create
	U al	[U/] 1756-UF6VI/A	1.9		Create
	J	[U8] 1756-IF6I/A	1.12		Lieale

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

New Module		X
Type: Vendor: Parent:	1756-0B16D 16 Point 19.2V-30V DC Diagnostic Output Allen-Bradley Local	
Name:	DigitaLOUT Slot: 0 +	
Description:		
Comm Format:	Full Diagnostics - Output Data	
Revision:	2 3 🚊 Electronic Keying: Compatible Keying	
🔽 Open Modul	e Properties OK Cancel Help	

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



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

Module Properties Report: Local:0 (1756-0B16D 2.3)						
General Connection Module Info Configuration Diagnostics Pulse Test Backplane						
Requested Packet Interval (RPI): 20.0 🚎 ms (0.2 - 750.0 ms)						
🗖 Inhibit Module						
Major Fault On Controller If Connection Fails While in Run Mode						
r Module Fault						
Status: Running OK Cancel Apply Help						

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



- 7. Using the *Module Discovery* feature, add the 1756-IB16D/A module in Slot 2 to the I/O Configuration. Name the module *Digital_IN*.
- 8. Using the *Module Discovery* feature, add the 1756-OF6VI/A module in Slot 7 to the I/O Configuration. Name the module *Analog_OUT*.
- Using the *Module Discovery* feature, add the 1756-IF6I/A module in Slot 8 to the I/O Configuration. Name the module *Analog_IN*.
- 10. When you are finished, your I/O Configuration should look like the following:



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



- 12. Select Go Offline.
- 13. Save the program by clicking on the Save icon in the toolbar.

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

You will continue to use the project already opened.

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

The tag editor window will appear.	If necessary, drag to the right to increase the size of the Tag Name field. This will allow you to view the entire Tag Name.
Controller Tags - Controller1(controller)	

S	cope: 🛐 Controlle	erl 🔽 Chow:	All Tags		▼ 7.	Enter Name Filter				•
	Name 💷	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style		
	+-Local:0:C			AB:1756_D0_DC		Read/Write	Г			P
	+-Local:0:1			AB:1756_D0_DC		Read/Write	Γ			rope
	+-Local:0:0			AB:1756_D0:0:0		Read/Write	Г			ortie
	⊞-Local:2:C			AB:1756_DI_DC		Read/Write	Γ			0
	±-Local:2:1			AB:1756_DI_DC		Read/Write	Г			
	⊞-Local:7:C			AB:1756_A06_Fl		Read/Write	Γ			
	±-Local:7:1			AB:1756_A06_Fl		Read/Write	Г			
	±-Local:7:0			AB:1756_A06_Fl		Read/Write	Γ			
	+-Local:8:C			AB:1756_AI6_Flo		Read/Write	Г			
	±-Local:8:1			AB:1756_AI6_Flo		Read/Write	Γ			
¢										
									◄	
•	Monitor Tags	λEdit Tags /		•				•		

FYI

	An I/O address follows this format:
	Location :Slot :Type .Member .B:
Wheney	= Up
wnere:	IS:
Localion	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
	0 = 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 modu

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

	Controller Tags - Controller1(controller)							
	Scope: 🚺 Controller1		ller1	Show: A		All Tags		
		Name ==	$ \Delta $	Alias For		Base Tag		
Ľ		⊞-Local:0:C						
		⊞-Local:0:I						
		+-Local:0:0						
16								

Scope: Controller1 Show: All Tags V Enter Mane Filter Name Image: A Value Force Mask Style Data Type Description Constant Image: A Value Force Mask Style Data Type Description Constant Image: A Value Force Mask Style Data Type Description Constant Image: A Value Force Mask Style Data Type Description Constant Image: A Value Force Mask Style A8:1756_D0_DC Image: Arrow Image: Arrow Image: A Value Force Mask Style A8:1756_D0_DC Image: Arrow Image: Arrow Image: A Value Force Mask Force Mask Force Mask Style Data Type Description Image: A Value Force Mask Force Mask Force Mask A8:1756_D0_DC Image: Arrow Image: Arrow Image: A Value Force Mask Force Mask Force Mask Force Mask Image: Arrow Image: Arrow Image: Arrow Image: Arrow	Controller Tags - Controller1(controller)						Ø	
Name Tele Force Mask Style Data Type Description Constant ⊞-Locat0.C () () AB:1756_D0_DC □ ⊞-Locat0.1 () () AB:1756_D0_DC □ ⊞-Locat0.0 () () AB:1756_D0_DC □ ⊞-Locat0.0 () () AB:1756_D0_DC □ ⊞-Locat2.C () () AB:1756_D0_DC □ ⊞-Locat2.1 () () AB:1756_D0_DC □ ⊞-Locat7.C () () AB:1756_A06_FL □ ⊞-Locat7.1 () () AB:1756_A06_FL □ ⊞-Locat7.0 () () AB:1756_A06_FL □ ⊞-Locat8.C () () AB:1756_A06_FL □ ⊞-Locat8.1 () AB:1756_A16_Flo □	-	🔽 🛛 🔽 Enter Name Filter			All Tags	1 💌 Show:	cope: 🛐 Controller	s
H-LocatOC () () AB:1756_D0_DC Image: Constraint of the state	▲ _{[∞}	escription Constant	Data Type	Style	🖌 Force Mask 🛛 🗲	Value 🗧	Name 📰 🛆	
⊞Hocat01 () () A8:1756_D0_DC □ ⊞Hocat00 () () A8:1756_D0:00 □ ⊞Hocat2C () () A8:1756_D1_DC □ ⊞Hocat21 () () A8:1756_D1_DC □ ⊞Hocat21 () () A8:1756_D1_DC □ ⊞Hocat71 () () A8:1756_A06_FL □ ⊞Hocat7:0 () () A8:1756_A06_FL □ ⊞Hocat8:C () () A8:1756_A06_FL □ ⊞Hocat8:C () () A8:1756_A06_FL □			AB:1756_D0_DC		} {}	{}	±-Local:0:C	
⊞Hocat0:0 () () A8:1756_D0:0:0 □ ⊞Hocat2:C () () A8:1756_D1_DC □ ⊞Hocat2:I () () A8:1756_D1_DC □ ⊞Hocat2:I () () A8:1756_A06_FL □ ⊞Hocat7:0 () () A8:1756_A06_FL □ ⊞Hocat8:C () () A8:1756_A06_FL □ ⊞Hocat8:C () () A8:1756_A06_FL □	rop	Г	AB:1756_D0_DC		} {}	{}	+-Local:0:I	
⊞Hocat2C () () A8:1756_DL_DC ⊞Hocat21 () () A8:1756_DL_DC ⊞Hocat7C () () A8:1756_A06_FL. ⊞Hocat70 () () A8:1756_A06_FL. ⊞Hocat8C () () A8:1756_A06_FL. ⊞Hocat8C () () A8:1756_A06_FL. ⊞Hocat8C () () A8:1756_A06_FL. ⊞Hocat8C () () A8:1756_A06_FL.	ortie		AB:1756_D0:0:0		} {}	{}	⊞-Local:0:0	
⊞-Local.2.1 {} {} AB:1756_DI_DC □ ⊞-Local.7.C {} {} AB:1756_A06_FL. □ ⊞-Local.7.1 {} {} AB:1756_A06_FL. □ ⊞-Local.7.0 {} {} AB:1756_A06_FL. □ ⊞-Local.8.C {} {} AB:1756_A06_FL. □ ⊞-Local.8.1 {} {} AB:1756_A16_Flo □			AB:1756_DI_DC		} {}	{}	⊞-Local:2:C	
⊞-Local7.C {} {} AB:1756_A06_FL □ ⊞-Local7.1 {} {} AB:1756_A06_FL □ ⊞-Local7.0 {} {} AB:1756_A06_FL □ ⊞-Local8.C {} {} AB:1756_A16_Flo □ ⊞-Local8.1 {} AB:1756_A16_Flo □			AB:1756_DI_DC		} {}	{}	+-Local:2:I	
⊞-Local 7:1 () () AB:1756_A06_FL □ ⊞-Local 7:0 () () AB:1756_A06_FL □ ⊞-Local 8:0 () () AB:1756_A16_Flo □ ⊞-Local 8:1 () () AB:1756_A16_Flo □		Г	AB:1756_A06_Fl		} {}	{}	⊞-Local:7:C	
⊞-Local 7.0 () () AB:1756_A06_FL □ ⊞-Local 8.C () () AB:1756_A16_Flo □ ⊞-Local 8.1 ()) AB:1756_A16_Flo □		Г	AB:1756_A06_Fl		} {}	{}	⊞-Local:7:I	
⊞-Local.8.C {} AB:1756_AI6_Flo □ ⊞-Local.8.1 {} AB:1756_AI6_Flo □			AB:1756_A06_Fl		} {}	{}	⊞-Local:7:0	
⊞-Local:8:1 {} AB:1756_Al6_Flo			AB:1756_AI6_Flo		} {}	{}	⊞-Local:8:C	
			AB:1756_AI6_Flo		}	{}	+-Local:8:I	
		k}						

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

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

Tag Properties Pane:

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



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

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

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

Assigning Alias Tags

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

FYI

Aliasing

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

Both tags share the same value

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

Use Aliases in the following situations:

-Program logic in advance of wiring diagrams

-Assign a descriptive name to an I/O device

-Provide a simpler name for a complex tag

-Use a descriptive name for an element of an array

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



The ladder editor appears as shown below:

Favo	コートコート・キャー・オート・オート・オート・コート・コート・コート・コート・コート・コート・コート・コート・コート・コ	equencer X E
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.

- Motor_Start will be Aliased to input point zero of the 1756-IB16D in slot two.
- Motor_Stop will be Aliased to input point one of the 1756-IB16D in slot two.
- Motor_Run will be Aliased to output point zero of the 1756-OB16D in slot zero.
 - 2. Right click on the tag Motor_Start and select Edit 'Motor_Start' Properties.



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

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

💰 Tag Propert	ies - Motor_Start	
General*		
Name:	Motor_Start	
Description:		
Unana		
Usage:		
Туре:	Alias Connection	
Alias For:		
Data Type:	BOOL	
Scope:	🕞 MainProgram	
External		
Style:	Decimal	
Constant		
🗖 Open Par	ameter Connections	
	OK Cancel Apply	Help

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

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

Usage:	Local Tag	•	L
Туре:	Alias Conn	rection	
Alias For:		•	
Data Type:	Y. Enter Name Filter	Show: All Tags	•
Scoper	Name _=	Data Type Usage	Descripti 🔺
Deope.	¶ ⊞-Local:0:C	AB:1756_DO <controller></controller>	·
External	∎ ⊕Local:0:I	AB:1756_DO <controller></controller>	
Access:	Ē ⊞-Local:0:O	AB:1756_DO: <controller></controller>	
Style:	∎ ELocal:2:C	AB:1756_DI <controller></controller>	
	∎ -Local:2:I	AB:1756_DI <controller></controller>	
Composite	∎ ⊕Local:7:C	AB:1756_AO <controller></controller>	-
🔲 Open Parar	Show controller tags		
	🔽 Show program tags		
	Show parameters from other p	program:	
	<none></none>	•	
	,		

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

Usage:	Local Tag	–	
Type:	Alias Conn	rection	
Alias For:		•	
Data Type:	Y. Enter Name Filter	Show: All Tags	-
Scope:	Name 18	Data Type Usage	Descripti 🔺
	∎ ⊕Local:0:C	AB:1756_DO <controller></controller>	
External	l 🖣 ⊞-Local:0:I	AB:1756_DO <controller></controller>	
Access:	∎ 🗍 🕀 Local:0:0	AB:1756_DO: <controller></controller>	
Style:	∎ ⊞-Local:2:C	AB:1756_DI <controller></controller>	
Constant	∎ ⊕-Local:2:I	AB:1756_DI <controller></controller>	
Constant	∎ ⊥ocal:7:C	AB:1756_AO <controller></controller>	-
🗖 Open Paran	Show controller tags		
	Show program tags		
	Show parameters from other j	program:	
	<none></none>	•	

- 6. Expand Local:2:1 by clicking on the + sign and select Local:2:1.Data.
- Click the **down arrow** for *Local:2:I.Data* as shown below. This will open the table of data points for the 1756-IB16D module.

	Alias For:	Local:2:I.Data														
	Data Type:	7,	E	nter.	Nan	ne Fi	lter		•	·	Sho	w: All	Tags			•
	Scope: I		N	ame	- ek 0			==	: D	ata 1 D-4 2	Гуре		Usage		Descripti	
	External Access: Style:	0	+ +	j-Loc	cal:2	::C			A	B:17	56_ '56_	DO DI	<controller></controller>			Γ
		1		Loc L	cal:2 Loca	21 al: 2:1	.Fau	łt	A Di	B:17 INT	'56_	DI	<controller></controller>			
	Constant	ð			Loca	al:2:1	.Data	a		INT		•				
	C Open Paran	1		0	1 9	2	3	4	5	6 14	7 15					•
			Sł	16	17	18	19	20	21	22	23					
		L. Cha	SH	24	25	26	27	28	29	30	31					
		Shc Kni	one	para »>	mete	ers fr	om o	other	proj	gram	1:		-			
		<u></u>														

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

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

ổ Tag Propert	ies - Motor_Start	<u> </u>
General*		
<u>N</u> ame:	Motor_Start	
Description:	A	
<u>U</u> sage:	Local Tag 💌	
Typ <u>e</u> :	Alias Connection	
Alias <u>F</u> or:	Local:2:I.Data.0	
Data <u>T</u> ype:	BOOL	
Scope:	िड्ड MainProgram	
E <u>x</u> ternal	Read/Write	
Style:	Decimal	
□		
C Open Para	ameter Connections	
	OK Cancel <u>A</u> pply	Help

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

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

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



- 10. Using the previous steps, alias the remaining two tags.
 - Motor_Stop = Local:2:I.Data.1
 - Motor_Run = Local:0:O.Data.0
- 11. When you are finished the ladder code should appear as follows:

Motor_Start	Motor_Stop	Motor_Run
<local:2:i.data.0></local:2:i.data.0>	<local:2i.data.1></local:2i.data.1>	<local:0:o.data.0></local:0:o.data.0>
∫ C Motor_Run <locat:0:o.data.0></locat:0:o.data.0>	, <u>_</u>	

- 12. Save the program by clicking on the Save icon in the toolbar.
- 13. Click Download on the Controller Faceplate.



- 14. When prompted to confirm the download, press Download.
- 15. Click **Yes** when prompted to change the controller mode to Remote Run.

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

Section 5: Testing Your Logic Program

This lab section should take roughly 5 minutes to complete.

Objective:

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

FYI

I/O Mapping

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

Motor_Start = DI0 Motor_Stop = DI1

Motor_Run = DO0

Switching the Controller into Run Mode and Testing the Program

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



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

Rem Run	1.	🗖 Run Mode	
No Forces	⊳⊾		_₽_
No Edits	2	Energy Storage UK	
Redundancy	Ŀ¢Ų	- 10 01	

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

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



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



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



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



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



5.

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



6. Press button 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.



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

Motor_Start	Motor_Stop	Motor_Run
<local:2:i.data.0></local:2:i.data.0>	≺Local:2:I.Data.1>	<locat:0:o.data.0></locat:0:o.data.0>
Motor_Run <local:0:0.data.0></local:0:0.data.0>	, <u></u>	

8. Press pushbutton *D*1 (Motor_Stop) and verify that output *DOO* (Motor_Run) turns off.

Motor_Start	Motor_Stop	Mot	or_Run
<local:2:i.data.0></local:2:i.data.0>	<local:2:i.data.1></local:2:i.data.1>	<local:0< td=""><td>):O.Data.0></td></local:0<>):O.Data.0>
Motor_Run <local:0:0.data.0></local:0:0.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.

💰 Logix Designer - Controller1 [1756-L75 22.1]* - [M	ainProgram - MainRoutine*]								
File Edit View Search Logic Communications Tools Window Help									
	- <u>#49</u> [= <u>1</u>] <u>9</u>	Select language							
Rein Run Image: Controller OK Controller OK No Edite Image: Controller OK Energy Storage OK Redundancy Mage: Controller OK Image: Controller OK	Park AB_ETHIP111921681.11193cstplane1 2 25								
Controller Organisar Controller Tags Controller Tags Controller Tags Controller Ass Controller Ass	Image: State of the state o	Meter Run <locatil (b)<br="" 0="" beta="" o=""></locatil>							

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.

💰 Logix Designer - Controller1 [1756-L75 22.1]* - [N	lainProgram - MainRoutine*]	_D×
File Edit View Search Logic Communications To	ols Window Help	_ 8 ×
	- <u>#44 12 12 22</u> 22	Select language H H H H H H H H H H H
Rem Bun	Path: AB_ETHIP-1\192.168.1.11\Backplane\1 ▼ 者	
No Forces P. Energy Storage DK. Redundancy Storage T/D DK	H Heri He Mou HAM RAO OR XOR SUPE HOT CLR BTO K	
Controller Organizer - 7 X	国 瞬間用 ● 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Controller Tags Controller Tags Power-Up Hander Task Manifrogram Perameters and Local Tags Perameters and Local Tags Matino Groups Matino Groups Ma	0 1 C (End) (End) Motor_Start Moto	Motor Run <local ()="" dels="" o=""></local>

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

7.	Enter Name Filter	Mo	MO ve urce Loc Show: All	V	
	Name	 ⊒≘ Data	, Туре	Usage	Descripti
1	Local:8:1.Ch5Ll	L BOOL	-		
Ĩ.	Local:8:1.Ch5Hl	H BOOL	-		
1	_Local:8:1.Ch0D;	ata REAL			
Ĩ	Local:8:I.Ch1D	ata RE			
ā.	Local:8:1.Ch2D	ata REAL			
ă.	Local:8:1.Ch3D	ata REAL			
Sho	Show controller tags Show program tags w parameters from oth	ner program	n:	-	

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



6. The rung should look like the following.



Adding the Timer to the Logic

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



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

眙			
0		Motor_Start Motor_Stop <locat:2:idata.0> <locat:2:idata.1> Motor_Run <locat:0:d.data.0> E</locat:0:d.data.0></locat:2:idata.1></locat:2:idata.0>	Motor_Run <local:0:o data.0=""></local:0:o>
1	r r r r r r	Motor_Start Motor_Stop <local:2:idata.0> Motor_Run <local:0:d.data.0> E</local:0:d.data.0></local:2:idata.0>	Motor_Run <locat.0:o.data.0> ()</locat.0:o.data.0>
2			MOV- Source Locat81.Ch0Data 5.13663469e-003 ← Dest Locat7:0.Ch0Data 5.13663469e-003 ←
(End)		

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

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



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



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

围		abcd ab	<ab> e e→x e</ab>	A RR	¥¥	₩₩	**	*			
0	e e e e e e	Motor_Start <local:21.deta.0> Motor_Run <local:0:o.data.0></local:0:o.data.0></local:21.deta.0>	Motor_Stop <local:2:i.data.1> /</local:2:i.data.1>						Motor_Run <local:0:0.data.0></local:0:0.data.0>	TON- Timer On Delay Timer ? Preset ? Accum ?	N)
1	r r r r r r	Motor_Start <local:2:i.data.0>] [Motor_Run <local:0:0 data.0=""></local:0:0></local:2:i.data.0>	Motor_Stop <local:2:i.data.1> /</local:2:i.data.1>							Motor_Ru <local:0:o.da< td=""><td>n ta.0></td></local:0:o.da<>	n ta.0>

FYI

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

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

	Motor_Run <local:0:o.data.0> ()</local:0:o.data.0>	TON- er On Delay	
	New Tag		? -(DN)
Ж	Cu <u>t</u> Instruction	Ctrl+X	?
8	⊆opy Instruction	Ctrl+C	
ß	Paste	Ctrl+V	lotor_Run
	Delete Instruction	Del	al:0:0.Data.0>
	Add Ladder Element	Alt+Ins	
	Edit <u>M</u> ain Operand Description	n Ctrl+D	
	Save Instruction Defaults		
	Clear Instruction Defaults		
	R <u>e</u> move Force		Ch0Data
	<u>G</u> o To	Ctrl+G	69e-003 < Ch0Data
	Instruction <u>H</u> elp	F1	69e-003 ←

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

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

New Program	Parameter or Tag	×
Name:	Timer	Create
Description:	<u> </u>	Cancel
		Help
Usage:	Local Tag	
Parameter Connection:		
Туре:	Base Connection	
Alias For:		
Data Type:	TIMER	
Scope:	🕞 MainProgram 💌	
External Access:	Read/Write	
Style:	<u></u>	
Constant		
Sequencing		
Copen Configuration		
C Open Parameter Connections		
7. Verify that the tag has been created in the timer instruction as shown below:

0	Motor_Start <local:2:i.data.0> E Motor_Run <local:0:0.data.0></local:0:0.data.0></local:2:i.data.0>	Motor_Stop <local 21.data.1=""> /</local>	Motor_Run <locat.0:o.data.0> Timer On Delay Timer Timer Preset 0 Accum 0</locat.0:o.data.0>	
r r r r r r r	Motor_Start <local:2:i.data.0> Motor_Run <local:0:0.data.0></local:0:0.data.0></local:2:i.data.0>	Motor_Stop <local:2:i.data.1>]/ []</local:2:i.data.1>	Motor_Run ≪Locat0:0.Data.0> ⟨)	

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



9. Enter a value of 32767.

FYI

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

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



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

11. Click on the Finalize All Edits icon

問題	國田田 ••• \$*••• • • 『IIIX IX \$\$ \$\$ \$\$ \$\$	
0 i 1 1 1 1	Motor_Start Motor_Stop Motor_Run <locat 21data.0=""> <locat 21data.1=""> <locat 0.0="" data.0=""> Motor_Run <locat 0.0="" data.0=""></locat></locat></locat></locat>	Timer On Delay Timer Timer Preset 32767 Accum 0
1 r r r r r r	Motor_Start <local:2:idata.0> Motor_Run <local:0:odata.0> E</local:0:odata.0></local:2:idata.0>	Motor_Run <locat0:o.data.0> < ></locat0:o.data.0>
2 i i i i i i i i		MOV Source Locat:8:1.Ch0Data 5.13663469e-003 ← Dest Locat:7:0.Ch0Data 5.13663469e-003 ←

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

RSLogix 5000	X			
Finalize all edits in program 'MainProgram'.				
Finalize all 'i, 'i,' d', 1' 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. Dutputs 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.	r			
This operation cannot be undone.				
Finalize all edits in program?				
Yes No Help				

The ladder editor will now appear as follows:

田盛	小田政 KX XX	(*)	
0	Motor_Start Motor_Stop <local:21:data.0> <local:21:data.1> </local:21:data.1></local:21:data.0>	Motor_Run <locat0:0.data.0> < ></locat0:0.data.0>	Timer On Delay Timer Timer Preset 32767 ← (DN)- Accum 0 ←
1			MOW Source Local:8:I.Ch0Data 5:13663463e-003 ← Dest Local:7:O.Ch0Data 5:13663469e-003 ←
(End)			

Testing Your Logic

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

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

Section 7: Creating and Running a Trend

This lab section should take roughly 5 minutes to complete.

Objective:

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

In this Lab you will:

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

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

FYI

Trending

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

Basic Trending has these constraints: you can trend data elements of type BOOL, SINT, INT, DINT, and REAL, you are limited to sampling eight unique data elements in a single trend.

Creating and Running a Trend

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



The New Trend window appears.

New Trend - Ger	eral			×
Name:				
Description:			Ā	
Sample Period	10	Millisec	ond(s) 💌	



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

New Trend - Gene	eral	×
Name:	Timer_Trend	
Description:		
Sample Period:	10 Millisecond(s)	
Cancel	Kent Seck Next Sector Help	

3. Click Next.

The New Trend Add/Configure Tags window appears.

New Trend - Add/Configure Tags	×
Scope:	
10 Controller1	
AvailableT ags:	
Name	
∎ Elocal:0:C	
I I I coat:0:0	
I III.ocal:2:C	
1 ⊞-Local:2:I	-
Add 1	
Tags To Trend:	
Danua	
Hemove	
Cancel < Back Next > Finish	Help

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

New Trend - Add/Configure Tags				
Cooper				
AinProgram 🗾				
Controller1				
MainProgram	_			

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

Nev	v Trend - Add/Configure Tags	×
	Scope:	
	📑 MainProgram	
	AvailableTags:	
	Name	-
	Motor_Run	
	Motor_Start	
	Motor_Stop	
	<u></u> ±-1imer	
		-
	Add	
	Tags To Irend:	
		_
	<u> </u>	
	<u>R</u> emove	
_		
	Cancel < Back Next > Finish Help	,

- 5. Expand the *Timer* tag by clicking on the +.
- 6. Select *Timer.ACC* and then click the **Add** button. This will add the tag Timer.ACC to the *Tags To Trend* list.

New Trend - Add/Configure Tags	[×
Scope:		
🕞 MainProgram 🗸		
, AvailableTags:		
Name	-= -	
-Timer		
Timer.PRE		
-Timer.ACC		
Timer.EN		
	-	
Add		
Tags to Trend:		
MainProgram\Timer,ACC		
Bemove		
<u><u>H</u>einove</u>		
· · · · · · · · · · · · · · · · · · ·		
Cancel < <u>B</u> ack <u>N</u> ext⇒ Finish	Help	

7. Click on *Finish*

The Trend window will now appear.

Trend - Timer_Trend			
Run Stop Errors Log -	Logging Sto	pped Periodic 10 ms	Capture: 0 🗾 of 0
MainProgram\Timer.ACC 1	Timer_Trend	Wednesday, September 21, 2011	
<u></u> 1			
0			
0			
0			
0			
0			
1:30:34 PM 1:	30:34 1:3	0:34 1:30:35 1:3	0:35 1:30:36 PM

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

🗟 Trend - Timer_Trend		
Run Stop Errors Log -	Logging Stopped Periodic 10 ms	Capture: 🕛 📻 of O
MainProgram.Timer.ACC	Timer_Trend VVednesday, Septembe	r 25, 2013
	✓ Scroll	
	Active Value Bar	
	Undo Zoom/Pan Print Trend	
	Create Snapshot	
0		

The RSTrendX Properties window will now appear.

- 9. Click on the *X-Axis* tab.
- 10. Change the Chart time range Time span from Second(s) to Minute(s).
- 11. click **OK**

RSTrendX Properties	:
Name General Display Pets X-Axis Y-kis 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 date on scale 0 Minutes 4 Major grid lines 0 Minute grid lines	
OK Cancel Apply Help	

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

	Figure Trend				
C	Run Strip Errors Log -	Logging Sto	pped Periodic 10 ms	Capture: 0	of 0
	MainProgram\Timer.ACC 1	Timer_Trend	Wednesday, September 21, 2011		
	0				

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



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

- 15. Try pressing the **DI1** pushbutton and watch the trend.
- 16. When you are finished investigating the trend, click **Stop** and **close** the trend window.

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

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

This section should take about 10 minutes to complete.

Objective:

This lab section covers creating and using custom data structures.

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

Creating User Defined Types

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

FYI

What is a UDT and what is it good for?

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

Continue to use the project already open.

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



83 of 98

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*						
Name:	Gallons_to_Liters				Data Type Size: ??	Properties	- ₽
	I					Extended Propertie	s 🔻
Description:		Holds	; gallons and quivalent in			🖂 General	
		che e	liters I			Data Type Size	??
M						Description	Holds gallons an …
Members:	Data T	una Description				Name	Gallons_to_Liters
	Id Member	ype Description			A		
					-		
	\	ОК	Cancel	Apply	Help		
	· · · · ·						

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

Members:

	Name		Data Type	Description	
	Gallons	Τ		4	4
		-			

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

Membe	ers:				External A
- 24	Name	Data Type 🔻	Description		Name
*	Gallons	REAL]		A	
	🔆 Add Memi	ber			
1					

84 of 98

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



7. Click Apply.

The window should appear as shown.

107 010	Data Type	: Gallons_to	_Liters					_	
F	Name:	Gallons_to_Li	iters		Data Type Size:	8 bytes	Properties		- ₽,
L		I					Extended Propertie:	s	•
(Description:			Holds gallons and			🖃 General		
L				the equivalent Liters			Data Type	REAL	
L							Description		
1	Members:						External Access	Read/Write	•
	Nam Nam	э	Data Type	Description			Name	Liters	
	Gallo	ns	REAL			<u></u>	Style	Float	•
	Liters	5	REAL						
	* /	1dd Member							
						-			
Ľ					\frown				
				OK Cancel	Apply H	elp			

- 8. Click **OK**to close the window.
- 9. Double click on *Parameters and Local Tags* under the MainProgram as shown to open the tag window.



- 10. On the blank row, fill in "Gallons_to_Liters" for the tag Name.
- 11. On the same row, select "Gallons_to_Liters" for the Data Type as shown and click OK.

Sc	ope: 🕞 MainProgr	am 💌 Si	how: All Tags			• 7	Enter Name Filter			
	Name === 🛆	Usage	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style	4
	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	
	H Timer	Local			пмев		Read/Write	Γ		
	Gallons_to_Liters	Logal			Gallons_to_Liters 🚥		Read/Write		Decimal	
				Select Data Type	e	at .	×			
				Data Types:		-0				
				Gallons_to_Liters	s		ок			
					2400					
				FILTER_HIGH_F	-422 2422		Lancel			
				FILTER NOTCH			Help			
				FIVE_POS_MOD	E_SELECTOR					
				FLIP_FLOP_D						
				FLIP_FLOP_JK						
			(Gallons to Liters	FRATUR					
				- Array Dimension	ns					
				Dim 2	Dim 1	Dim 0				
				0 👻	0 🚊	0 🕂				
										_
	A			Show Data T	ypes by Groups					
4	Monitor Tags	<u>A Edit fags /</u>								

Add the UDT tag to an instruction

12. Double click on the MainRoutine.



- 13. Make sure the End rung is highlited. Click on the Highlited insert rung icon to create a new rung.
- 14. Find the Compute/Math tab on the instruction tool bar and click on the MUL instruction.

Compare	
Motor_Start Motor_Stop	ToN Timer On Delay Timer Timer Preset 32767 (DN) Accum 32767 (DN) MOV Source Locat 81 Ch0Data Second 4
2 e e e e e e e e e e e e e e e e e e e	B383/24 ← Dest Locat 7:0. ChOData 6.889254 ← MULL Multiply Source A ? Source B ? ?? Dest ? Dest ?? ??

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

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



- 16. Fill in 3.785 for source B (the conversion constant to convert gallons to liters).
- 17. Double click on the ? in the destination field and select the Gallons_to_Liters.Liters tag as shown.

間雪	瞬間間 eved st c.u ・ even 「UIIX IX See With With See See See See See See See See See Se	
2	Mut Sou Des	6.889254 ← Dest Local 7:0 Ch0Data 6.889254 ← Holds gallons and the equivalent Liters -MUL iply rce A Gallons_to_Liters.Gallons 0.0 ← rce B 3.785 t Gallons_to_Liters.Liters 0.0 ←
(End)		

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

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



Notice that the Liters value updates automatically.

Monitoring UDT Tags

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

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



FYI

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

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

Section 9: (Optional) Using Studio 5000 Help

This lab section should take roughly 15 minutes to complete.

Objective:

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

In this lab you will be viewing:

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

Instruction Help

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



The following window will appear.



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

Viewing I/O Module Wiring Diagrams

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



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

🔮 Logix Designer Online Help	the second s	×
Hide Back Print Options		
Contents Index Search Type in the word(s) to search for:	Wiring Diagram (<mark>1756-1A16</mark>)	
1756-IA16	All terminals with the same name are connected together on the module.	1
Select topic: Found: 6 Title Location Rank	For example, L2 can be connected to any terminal marked L2-0. When you daisy chain from	IN-1
Communication For Logix Desi 1 Wiring Diagrams (17 Module 2 Configure 1756 Digit Module 3	a group to another RTB, always connect the daisy chain to the terminal GI IS directly connected to the Group 0	IN-3 IN-5
Module Properties D., Module 4 Module-defined Dat., Module 5 Wring Diagram (175,, Module 6	IN-5 IN-4 R 77	IN-7
		L2-0
		IN-9
	Group 1	IN-13
	14 (13) IN-11 IN-10	IN-15
	16 15 ··································	
Search previous results Match similar words Search titles active	1+15 1+14 20 19	Daisy chain to
1 Search ares only		

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

Using Start Pages

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

🛐 Start Page			- +
Start Pa	Quick Start		LISTEN, THINK, SOLVE.
	Controller Projects		
Œ	▶ Get Started	Quick Start - My First Project	
	▶ Get Connected	Click on a Mv First Project topic and view an animated tutorial. The Mv First	
	My First Project	Project topics will help you learn how to create and configure your first controller project.	
	Create a New Project		
	 Modify the Main Task 		
	 Modify the Main Program Modify the Main Poutine 		
	Configure an Input		
	Configure an Output Module		
	➡ Create a Tag		N.S.
	Create a User-Defined Type		
	Routine Logic		
	Routine		
	Download a Project		
	🖝 Ga Online 🕞		

- 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!

96 of 98

Thank you for attending!

Notes :