Allen-Bradley

Logix5000 Controllers Design Considerations

1756 ControlLogix
1756 GuardLogix
1768 CompactLogix
1769 CompactLogix
1789 SoftLogix5800
1794 FlexLogix
PowerFlex 700S with DriveLogix

Reference Manual

Rockwell Automation

Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://literature.rockwellautomation.com) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

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

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

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING 	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION 	Identifies information about practices or circumstances that can lead to: personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
WARNING 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
BURN HAZARD 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

Logix5000 Controller Comparison

Characteristic	1756 ControlLogix	1756 GuardLogix	1768 CompactLogix	1769 CompactLogix	1789 SoftLogix5800	1794 FlexLogix	PowerFlex 700S2 DriveLogix
Controller tasks • Continuous • Periodic • Event	• 100 tasks • Event tasks: all event triggers	• 100 tasks • Event tasks: all event triggers	• 16 tasks • Event tasks: consumed tag, EVENT instruction, axis, and motion event triggers	• 1769-L35CR, -L35E: 8 tasks • 1769-L32C, -L32E: 6 tasks • 1769-L31: 4 tasks • Event tasks: consumed tag and EVENT instruction triggers	• 100 Tasks • Event tasks: all event triggers, plus outbound and Windows events	• 8 tasks • Event tasks: consumed tag and EVENT instruction triggers	• 8 tasks • Event tasks: axis and motion event triggers
User memory	1756-L55M12: 750 KB 1756-L55M13: 1.5 MB 1756-L55M14: 3.5 MB 1756-L55M16: 7.5 MB 1756-L55M22: 750 KB 1756-L55M23: 1.5 MB 1756-L55M24: 3.5 MB 1756-L60M03SE: 750 KB 1756-L61: 2 MB 1756-L62: 4 MB 1756-L63: 8 MB 1756-L64: 16 MB	1756-L61S: 2 MB Standard 1 MB Safety 1756-L61S: 4 MB Standard 1 MB Safety	1768-L43: 2 MB	1769-L31: 512 KB 1769-L32*x*: 750 KB 1769-L35*x*: 1.5 MB	1789-L10: 2 MB; 3 slots; No motion 1789-L30: 64 MB; 5 slots 1789-L60: 64 MB; 16 slots	1794-L34: 512 KB	256 KB 768 KB with memory expansion
Nonvolatile user memory	1756-L55M12: none 1756-L55M13: none 1756-L55M14: none 1756-L55M16: none 1756-L55M22: yes 1756-L55M23: yes 1756-L55M24: yes 1756-L6*x*: CompactFlash	CompactFlash	CompactFlash	CompactFlash	None	Yes	Yes (expansion memory)
Built-in communication ports	1 port RS-232 serial	1 port RS-232 serial	1 port RS-232 serial	• 1769-L31: 2 RS-232 ports • 1769-L32C, -L35CR: 1 ControlNet port and 1 RS-232 port • 1769-L32E, -L35E: 1 EtherNet/IP port and 1 RS-232 port	Depends on personal computer	• 1 RS-232 port • 2 slots for 1788 communication cards	• 1 RS-232 port • 1 slot for 1788 communication cards
Communication options	• EtherNet/IP • ControlNet • DeviceNet • Data Highway Plus • Remote I/O • SynchLink	• EtherNet/IP (standard; safety) • ControlNet (standard; safety) • DeviceNet (standard; safety) • Data Highway Plus • Remote I/O • SynchLink	• EtherNet/IP • ControlNet • DeviceNet	• EtherNet/IP • ControlNet • DeviceNet	• EtherNet/IP • ControlNet • DeviceNet	• EtherNet/IP • ControlNet • DeviceNet	• EtherNet/IP • ControlNet • DeviceNet
Serial port communication	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus	• ASCII • DF1 full/half duplex • DF1 radio modem • DH-485 • Modbus
Connections	• 100 ControlNet • 128 EtherNet/IP • 64 TCP/IP	• 48 ControlNet • 128 EtherNet/IP • 64 TCP/IP	• 48 ControlNet • 64 EtherNet/IP • 32 TCP/IP	• 32 ControlNet • 32 EtherNet/IP • 32 TCP/IP	• 48 ControlNet • EtherNet/IP limited by type/number cards	• 32 ControlNet • 32 EtherNet/IP • 64 TCP/IP	• 32 ControlNet • 32 EtherNet/IP • 64 TCP/IP
Controller redundancy	Full support	None	N/A	N/A	N/A	Backup via DeviceNet	N/A
Simple motion	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive	• Stepper • Servo via DeviceNet • Analog ac drive
Integrated motion	SERCOS interface Analog options: • Encoder input • LDT input • SSI input	SERCOS interface Analog options: • Encoder input • LDT input • SSI input	SERCOS interface	N/A	SERCOS interface Analog encoder input	N/A	• 1 full servo • 1 feedback axis
Programming languages	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC • External routines (developed in C/C++)	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC

Notes:

Summary of Changes

This version of the design guidelines corresponds to revision 16 of the Logix5000 controller firmware.

Change	Page
Continuous task execution time at least 1 ms	25
Output processing on event tasks	26
Configurable use for any unused part of the system overhead timeslice	28
Add-On Instructions	31
Addition of a LINT data type	41
Electronic keying	64
Control HART devices	68
Control FOUNDATION Fieldbus devices	69
Runtime addition of generic ControlNet and EtherNet/IP modules	73
Unicast produced and consumed tags over an EtherNet/IP network	77
DeviceNet tag generator tool	83
FactoryTalk Alarms and Events system	91
Controller-initiated firmware updates	117
Autoflash supports SERCOS drives	118

In addition, these modules are now available:

- 1768-L43 CompactLogix controller
- 1768-CNB, 1768-CNBR ControlNet modules
- 1768-EWEB EtherNet/IP web server module
- 1756-L64 ControlLogix controller
- 1756-CN2, 1756-CN2R ControlNet modules
- 1756-EN2T EtherNet/IP module

Notes:

Table of Contents

Notes:

Design Logix5000 Systems

Introduction

This reference manual provides guidelines you can follow to optimize your system. This manual also provides system information you need to make system design choices.

In addition to the controller-specific topics covered in each chapter, the back of this manual includes a:

- glossary of commonly used terms.
- list of related publications.

This manual is meant for experienced Logix-system programmers. The information in this manual is presented with the assumption that you understand how to implement the guidelines. The list of related publications web sites at the back of the manual identifies resources you can use for more details on how to implement the guidelines.

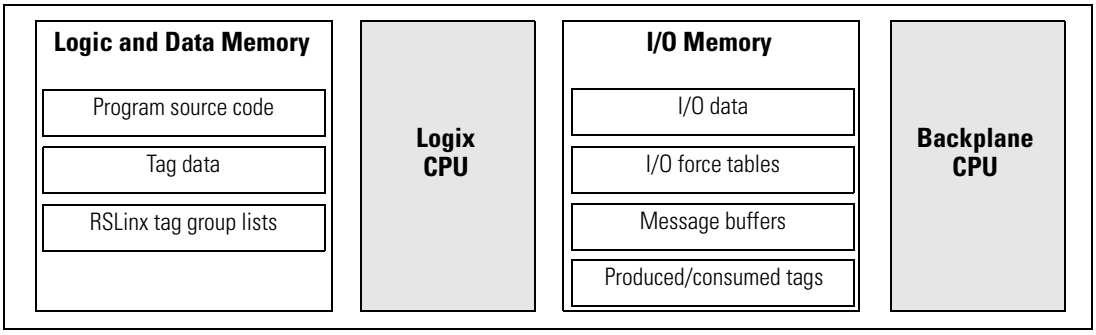
Notes:

Logix5000 Controller Resources

Introduction

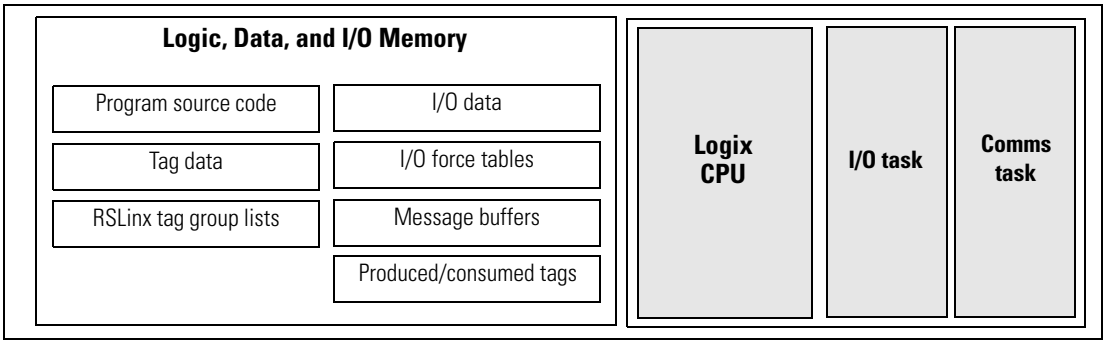
Depending on the controller, resources are divided differently.

1768 CompactLogix and ControlLogix controllers - Memory is separated into two, isolated sections.



- The Logix CPU executes application code and messages.
- The backplane CPU communicates with I/O and sends/receives data from the backplane. This CPU operates independently from the Logix CPU, so it sends and receives I/O information asynchronous to program execution.

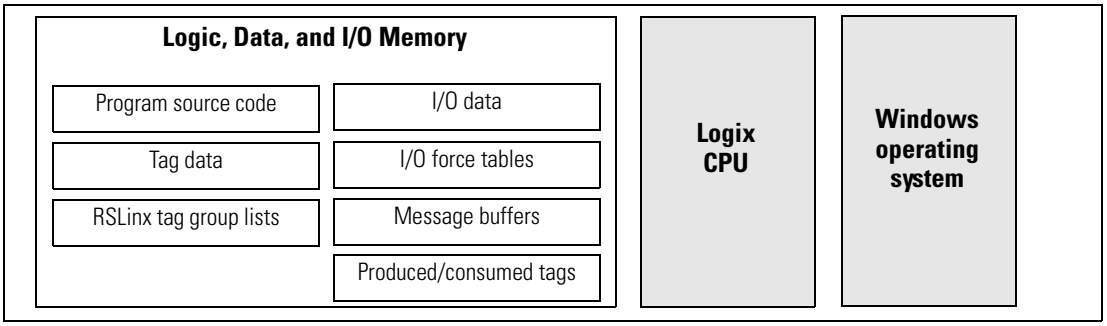
1769 CompactLogix, FlexLogix, and DriveLogix controllers - Memory is in one, contiguous section.



These controllers have a single CPU that performs all operations. Isolated tasks perform I/O and communication and interact with networks. These tasks simulate the backplane CPU.

Controller	I/O Task Priority	Communication Task Priority
1769 CompactLogix, FlexLogix, and DriveLogix	6	12

SoftLogix controllers - Memory is in one, contiguous section.



The SoftLogix controller has a single CPU that works in conjunction with the Windows operating system to perform all operations. Rather than using controller priority levels for I/O and communication tasks, the SoftLogix controller uses Windows priority levels for these tasks.

Controller	I/O Task Priority	Communication Task Priority
SoftLogix	Windows priority 16 (Idle)	Windows priority 16 (Idle)

For all controllers, memory is used at run time for:

- message processing to buffer incoming and outgoing messages.
- RSLinx data handling to store tag groups.
- online edits to store edit rungs.
- graphical trends to buffer data.

Estimate Memory Use

These equations provide an estimate of the memory needed for a controller.

Controller tasks	_____ * 4000	=	_____ bytes (minimum 1 needed)
Digital I/O points	_____ * 400	=	_____ bytes
Analog I/O points	_____ * 2600	=	_____ bytes
DeviceNet modules[1]	_____ * 7400	=	_____ bytes
Other communication modules[2]	_____ * 2000	=	_____ bytes
Motion axis	_____ * 8000	=	_____ bytes
FactoryTalk alarm instruction	_____ * 1000	=	_____ bytes (per alarm)
FactoryTalk subscriber	_____ * 10000	=	_____ bytes (per subscriber)
	Total	=	_____ bytes

[1]The first DeviceNet module is 7400 bytes. Additional DeviceNet modules are 5800 bytes each.

[2]Count all the communication modules in the system, not just those in the local chassis. This includes device connection modules, adapter modules, and ports on PanelView terminals.

Reserve 20...30% of the controller memory to accommodate growth.

RSLinx Software Use of Logix5000 Controller Memory

The amount of memory RSLinx software needs depends on the type of data RSLinx software reads. These equations provide a memory estimate.

RSLinx overhead (per connection)	_____	* 1345 =	_____	bytes (4 connections by default)
Individual tags	_____	* 45 =	_____	bytes
Arrays / structures	_____	* 7 =	_____	bytes
		Total =	_____	bytes

Consolidating tags into an array or a structure reduces the communication overhead and the number of connections needed to obtain the data.

Compare PLC/SLC MEMORY

The Logix5000 controllers use compiled instructions to provide faster execution times than PLC or SLC processors. The compiled instructions use more memory when compared to the instructions in PLC and SLC processors.

If you have a PLC/SLC program, you can estimate the number of bytes it will take in a Logix5000 controller by:

number PLC/SLC words * 18 = *number of Logix5000 bytes*

Controller Connections

A Logix5000 controller uses a connection to establish a communication link between two devices. Connections can be:

- controller to local I/O modules or local communication modules.
- controller to remote I/O or remote communication modules.
- controller to remote I/O (rack optimized) modules.

  For more information on connections for I/O, see Chapter 5, Communicate with I/O.
- produced and consumed tags.

  For more information, see Chapter 4, Share Tag Data with Other Controllers (produced and consumed tags).
- messages.

  For more information, see Chapter 7, Communicate with Other Devices.
- access to RSLogix 5000 programming software.
- RSLinx software access for HMI or other software applications.

The controllers have different communication limits.

Communication Attribute	ControlLogix and SoftLogix	CompactLogix	FlexLogix and DriveLogix
Connections	250	100	100
Cached messages[1]	32 for messages and block-transfers combined	32 for messages and block-transfers combined	32 for messages and block-transfers combined
Unconnected receive buffers	3	3	3
Unconnected transmit buffers	10 (can be increased to 40)	10 (can be increased to 40)	10 (can be increased to 40)

(1) See Chapter 7 Communicate with Other Devices for more information about messages and buffers.

The limit of connections may ultimately reside in the communication module you use for the connection. If a message path routes through a communication module, the connection related to the message also counts towards the connection limit of that communication module.

Controller	Communication Device	Supported Connections
ControlLogix	1756-CN2, 1756-CN2R	100 CIP connections (any combination of scheduled and message connections)
	1756-CNB,1756 -CNBR	64 CIP connections depending on RPI, recommend using only 48 connections (any combination of scheduled and message connections)
	1756-EN2T, 1756-ENBT 1756-EWEB	128 CIP connections 64 TCP/IP connections
1768 CompactLogix	1768-ENBT 1768-EWEB	64 CIP connections 32 TCP/IP connections
1769 CompactLogix	1769-L32C, 1769-L35CR	32 CIP connections depending on RPI, as many as 22 connections can be scheduled The remaining connections (or all 32, if you have no scheduled connections) can be used for message connections
	1769-L32E, 1769-L35E	32 CIP connections 64 TCP/IP connections
FlexLogix PowerFlex 700S with DriveLogix	1788-CN*x*, 1788-CN*x*R	32 CIP connections depending on RPI, as many as 22 connections can be scheduled The remaining connections (or all 32, if you have no scheduled connections) can be used for message connections
	1788-ENBT	32 CIP connections 64 TCP/IP connections
SoftLogix5800	1784-PCICS	128 CIP connections 127 of which can be scheduled connections

Determine Total Connection Requirements

The total connections for a Logix5000 controller include both local and remote connections. Tallying local connections is not an issue for FlexLogix or CompactLogix controllers because both support the maximum number of modules allowed in their systems. The ControlLogix and SoftLogix controllers support more communication modules than the other controllers, so you must tally local connections to make sure you stay within the 250 connection limit. Use this table to tally **local** connections.

Connection Type	Device Quantity	x	Connections per Module	=	Total Connections
Local I/O module (always a direct connection)		x	1	=	
Motion servo module		x	3	=	
ControlNet communication module		x	0	=	
EtherNet/IP communication module		x	0	=	
DeviceNet communication module		x	2	=	
DH+/Remote I/O communication module		x	1	=	
RSLogix 5000 programming software access to controller		x	1	=	
Total					

The communication modules you select determine how many remote connections are available. Use this table to tally **remote** connections.

Connection Type	Device Quantity	x	Connections per Module	=	Total Connections
Remote ControlNet communication module Configured as a direct (none) connection Configured as a rack-optimized connection		x	 0 **or** 1	=	
Distributed I/O module over a ControlNet network (direct connection)		x	1	=	
Remote EtherNet/IP communication module Configured as a direct (none) connection Configured as a rack-optimized connection		x	 0 **or** 1	=	
Distributed I/O module over an EtherNet/IP network (direct connection)		x	1	=	
Remote device over a DeviceNet network (accounted for in rack-optimized connection for local DeviceNet module)		x	0	=	
Other remote communication adapter		x	1	=	
Produced tag and first consumer Each additional consumer		x	1 1	=	
Consumed tag		x	1	=	
Connected message (CIP Data Table Read/Write and DH+)		x	1	=	
Block-transfer message		x	1	=	
RSLinx software access for HMI or other software applications		x	4	=	
RSLinx Enterprise software for HMI or other software applications		x	5	=	
Total					

Notes:

Divide Logic into Tasks, Programs, Routines, and Add-On Instructions

Introduction

The controller operating system is a preemptive multitasking system that is IEC 61131-3 compliant.

Tasks to configure controller execution

A task provides scheduling and priority information for a set of one or more programs. You can configure tasks as either continuous, periodic, or event.

Programs to group data and logic

A task contains programs, each with its own routines and program-scoped tags. Once a task is triggered (activated), all the programs assigned to the task execute in the order in which they are listed in the Controller Organizer.

Programs are useful for projects developed by multiple programmers. During development, the code in one program that makes use of program-scoped tags, can be duplicated in a second program and minimize the possibility of tag names colliding.

With firmware revision 15, tasks can contain programs and equipment phases.

Routines to encapsulate executable code written in a single programming language

Routines contain the executable code. Each program has a main routine that is the first routine to execute within a program. Use logic, such as the Jump to Subroutine (JSR) instruction, to call other routines. You can also specify an optional program fault routine.

See Develop Application Code in Routines on page 29 for information on selecting programming languages and how the controller prescans and postscans logic.

Add-On Instructions to encapsulate executable code into user-defined instructions

An Add-On Instruction is a user-created instruction that encapsulates logic. Add-On Instructions help divide a controller project into smaller, more manageable pieces.

Decide When to Use Tasks, Programs, and Routines

Use these considerations to determine when to use a task, program, or routine.

Comparison	Task	Program and Equipment Phase	Routine	Add-On Instruction
Quantity available	Varies by controller (4, 6, 8, or 32)	32 program and equipment phases (combined) per task (100 for ControlLogix and SoftLogix controllers)	Unlimited number of routines per program	Unlimited number of Add-On Instructions in a project
Function	Determines how and when code will be executed	Organizes groups of routines that need to share a common data area	Contains executable code (relay ladder, function block diagram, sequential function chart, or structured text)	Contains executable code (relay ladder, function block diagram, or structured text)
Use	• Most code should reside in a continuous task • Use a periodic task for slower processes or when time-based operation is critical • Use an event task for operations that require synchronization to a specific event	• Put major equipment pieces or plant cells into isolated programs • Use programs to isolate different programmers or create reusable code • Configurable execution order within a task • Isolate individual batch phases or discrete machine operations	• Isolate machine or cell functions in a routine • Use the appropriate language for the process • Modularize code into subroutines that can be called multiple times	• Standardize modules of code • Very specific or focused operations • Extensions to the base instruction set • Encapsulate an instruction from one language for use in another language • Instance based monitoring of logic and data
Considerations	• A high number of tasks can be difficult to debug • May need to disable output processing on some tasks to improve performance • Tasks can be inhibited to prevent execution	• Data spanning multiple programs must go into controller-scoped area • Listed in the Controller Organizer in the order of execution	• Subroutines with multiple calls can be difficult to debug • Data can be referenced from program-scoped and controller-scoped areas • Calling a large number of routines impacts scan time • Listed in the Controller Organizer as Main, Fault, and then alphabetically	• If you have a lot of parameters or specialized options, consider multiple Add-On Instructions • Calling a large number of Add-On Instructions impacts scan time • Must use cross-reference or find to locate calls to an Add-On Instruction • Can edit offline only • Supports only some data types. • Changes to data values must be made for each instance

For more information about equipment phases, see Chapter 11, Develop Equipment Phases for Batch Control.

Specify Task Priorities

Each task in the controller has a priority level. A higher priority task (such as 1) interrupts any lower priority task (such as 15). The continuous task has the lowest priority and is always interrupted by a periodic or event task.

This Logix5000 controller	Supports this many user tasks	And has this many priority levels
ControlLogix	32	15
1768-L43 CompactLogix	16	15
1769-L35CR, 1769-L35E CompactLogix	8	15
1769-L32C,1769-L32E CompactLogix	6	15
1769-L31 CompactLogix	4	15
FlexLogix	8	15
PowerFlex 700S with DriveLogix	8	15
SoftLogix5800	32	3

The Logix5000 controller has these types of tasks.

Priority	User Task	Description
Highest	N/A	CPU overhead - serial port and general CPU operations
	N/A	Motion planner - executed at coarse update rate
	N/A	Redundancy task - communication to 1757-SRM in redundant systems
	N/A	Trend data collection - high-speed collection of trend data values
	Priority 1 Event/Periodic	N/A
	Priority 2 Event/Periodic	N/A
	Priority 3 Event/Periodic	N/A
	Priority 4 Event/Periodic	N/A
	Priority 5 Event/Periodic	N/A
	Priority 6 Event/Periodic	1769 CompactLogix, FlexLogix, and DriveLogix controllers process I/O as a periodic task based on the chassis RPI setting
	Priority 7 Event/Periodic	N/A
	Priority 8 Event/Periodic	N/A
	Priority 9 Event/Periodic	N/A
	Priority 10 Event/Periodic	N/A
	Priority 11 Event/Periodic	N/A
	Priority 12 Event/Periodic	DriveLogix communication to drives CompactLogix and FlexLogix communication and scheduled connection maintenance
Lowest	Priority 13 Event/Periodic	N/A
	Priority 14 Event/Periodic	N/A
	Priority 15 Event/Periodic	N/A
	Continuous	Message handler - based on system overhead timeslice

If a periodic or event task is executing when another is triggered and both tasks are at the same priority level, the tasks timeslice execution time in 1 ms increments until one of the tasks completes execution.

Manage User Tasks

You can configure these user tasks.

If you want logic to execute	Then use a	Description
All of the time	Continuous task	The continuous task runs in the background. Any CPU time not allocated to other operations or tasks is used to execute the continuous task. • The continuous task runs all the time. When the continuous task completes a full scan, it restarts immediately. • A project does not require a continuous task. If used, there can be only one continuous task.
• At a constant period (e.g., every 100 ms) • Multiple times within the scan of your other logic	Periodic task	A periodic task performs a function at a specific time interval. Whenever the time for the periodic task expires, the periodic task: • interrupts any lower priority tasks. • executes one time. • returns control to where the previous task left off.
Immediately when an event occurs	Event task	An event task performs a function only when a specific event (trigger) occurs. Whenever the trigger for the event task occurs, the event task: • interrupts any lower priority tasks. • executes one time. • returns control to where the previous task left off. See Configure an Event Task on page 25 for the triggers for an event task. Some Logix5000 controllers do not support all triggers.

The user tasks you create appear in the Tasks folder of the controller. These pre-defined system tasks do not appear in the Tasks folder and they do not count toward the task limit of the controller:

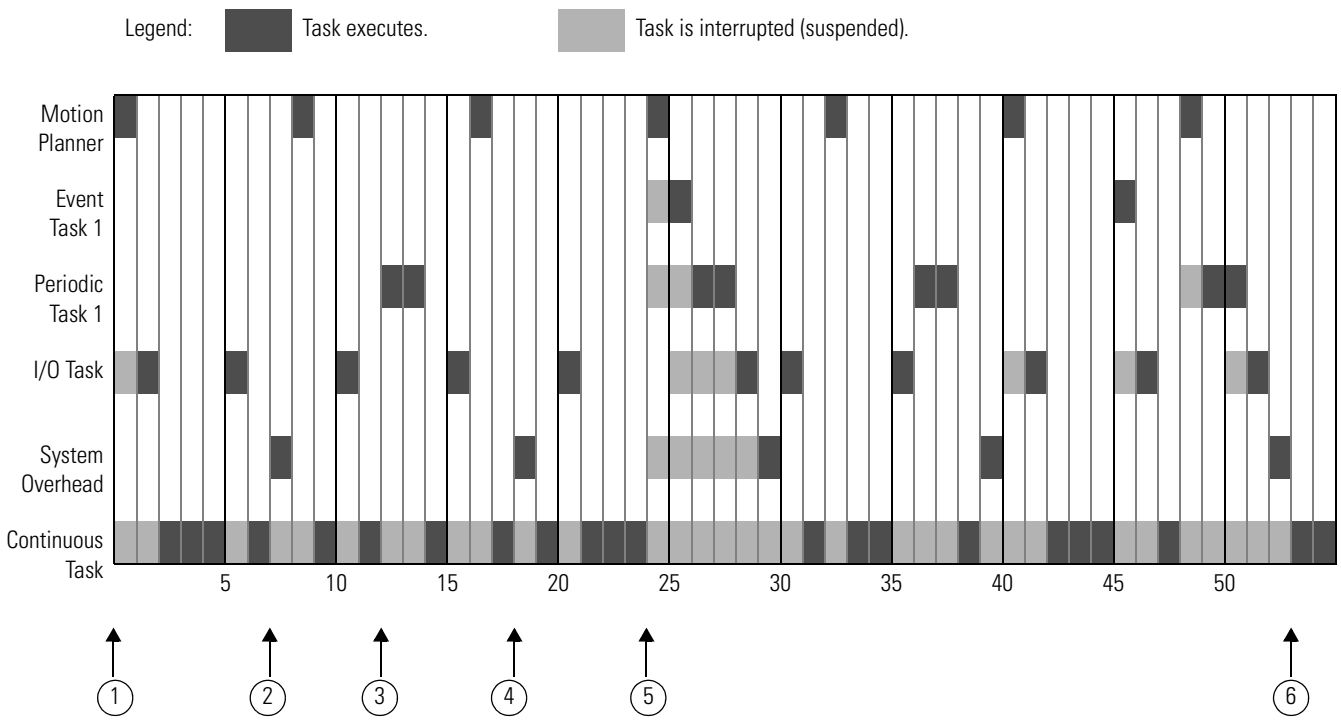
- Motion planner
- I/O processing
- System overhead
- Output processing

Considerations that Affect Task Execution

Consideration	Description
Motion planner See also Optimize an Application for Motion Control on page 99.	The motion planner interrupts all other tasks, regardless of their priority. • The number of axes and coarse update period for the motion group affect how long and how often the motion planner executes. • If the motion planner is executing when a task is triggered, the task waits until the motion planner is done. • If the coarse update period occurs while a task is executing, the task pauses to let the motion planner execute.
I/O processing	CompactLogix, FlexLogix, DriveLogix, and SoftLogix controllers use a dedicated periodic task to process I/O data. This I/O task: • for CompactLogix, FlexLogix, and DriveLogix controllers, operates at priority 6. For SoftLogix controllers, operates at Windows priority 16 (Idle). • higher-priority tasks take precedence over the I/O task and can impact processing. • executes at the fastest RPI you have scheduled for the system. • executes for as long as it takes to scan the configured I/O modules. • for local I/O, updates also occur at the end of each task.
System overhead See also Select a System Overhead Percentage on page 27.	System overhead is the time that the controller spends on message communication and background tasks. • Message communication is any communication that you do **not** configure through the I/O configuration folder of the project, such as MSG instructions. • Message communication occurs only when a periodic or event task is not running. If you use multiple tasks, make sure that their scan times and execution intervals leave enough time for message communication. • System overhead interrupts only the continuous task. • The system overhead timeslice specifies the percentage of time (excluding the time for periodic or event tasks) that the controller devotes to message communication. • The controller performs message communication for up to 1 ms at a time and then resumes the continuous task. • Adjust the update rates of the tasks as needed to get the best trade-off between executing your logic and servicing message communication. • Unused time allocated to system overhead is used to increase the scan rate of the continuous task, which can lead to scan time variability. To reduce this variability, reserve the time for communication via controller property settings.
Output processing	At the end of a task, the controller performs output processing for the output modules in your system. This output processing may effect the update of the I/O modules in your system.
Too many tasks	If you have too many tasks, then: • the continuous task may take too long to complete. • other tasks may experience overlaps. If a task is interrupted too frequently or too long, it may not complete its execution before it is triggered again. • controller communication might be slower. • if your application is designed for data collection, try to avoid multiple tasks. Switching between multiple tasks limits communication bandwidth.

This example depicts the execution of a project with these tasks.

Task	Priority	Period	Execution Time	Duration
Motion planner	N/A	8 ms (course update rate)	1 ms	1 ms
Event task 1	1	N/A	1 ms	1...2 ms
Periodic task 1	2	12 ms	2 ms	2...4 ms
I/O task—N/A to ControlLogix and SoftLogix controllers	7	5 ms (fastest RPI)	1 ms	1...5 ms
System overhead	N/A	Timeslice = 20%	1 ms	1...6 ms
Continuous task	N/A	N/A	20 ms	48 ms



Description	
1	Initially, the controller executes the motion planner and the I/O task (if one exists).
2	After executing the continuous task for 4 ms, the controller triggers the system overhead.
3	The period for periodic task 1 expires (12 ms), so the task interrupts the continuous task.
4	After executing the continuous task again for 4 ms, the controller triggers the system overhead.
5	The triggers occur for event task 1. Event task 1 waits until the motion planner is done. Lower priority tasks experience longer delays.
6	The continuous task automatically restarts.

Configure a Continuous Task

The continuous task is created automatically when you open an RSLogix 5000 project. A continuous task is similar to how logic executes on PLC-5 and SLC 500 processors. A Logix5000 controller supports one continuous task, but a continuous task is not required. You can configure whether the task updates output modules at the end of the continuous task. You can change the continuous task to either a periodic or event task.

The CPU timeslices between the continuous task and system overhead. Each task switch between user task and system overhead takes additional CPU time to load and restore task information.

RSLogix 5000 software, version 16, forces at least 1 ms of execution time for the continuous task, regardless of the system overhead timeslice. This more efficiently uses system resources because allowing shorter execution times of the continuous task means switching tasks more frequently.

Configure a Periodic Task

A periodic task executes automatically based on a preconfigured interval. This task is similar to selectable timed interrupts in PLC-5 and SLC 500 processors. You can configure whether the task updates output modules at the end of the periodic task. After the task executes, it does not execute again until the configured time interval has elapsed.

If your application has a lot of communication (such as message instructions or RSLinx communication), use a periodic task rather than a continuous task. This avoids the overhead associated with task switching, which can improve performance.

Configure an Event Task

An event task executes automatically based on a trigger event occurring or if a trigger event does not occur in a specific time interval. You configure whether the task updates output modules at the end of the task. After the task executes, it does not execute again until the event occurs again. Each event task requires a specific trigger.

Trigger	Description
Module Input Data State Change	The input module (digital or analog) triggers the event task based on the change of state (COS) configuration for the module. Enable COS for only one point on the module. If you enable COS for multiple points, a task overlap of the event task may occur. The ControlLogix sequence of events modules (1756-IB16ISOE, 1756-IH16ISOE) use the Enable CST Capture feature instead of COS.
Consumed Tag	Only one consumed tag can trigger a specific event task. Use an IOT instruction in the producing controller to signal the production of new data.
Axis Registration 1 or 2	A registration input triggers the event task.
Axis Watch	A watch position triggers the event task.
Motion Group Execution	The coarse update period for the motion group triggers the execution of both the motion planner and the event task. Because the motion planner interrupts all other tasks, it executes first.
EVENT Instruction	Multiple EVENT instructions can trigger the same task.

For more information on event tasks, see:

- Logix5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.
- Using Event Tasks with Logix5000 Controllers, LOGIX-WP003.

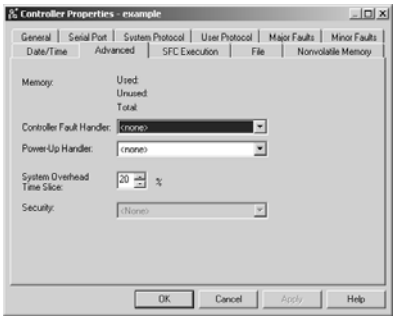
Guidelines to Configure an Event Task

Guideline	Description
Place the I/O module being used to trigger an event in the same chassis as the controller.	Placing the I/O module in a remote chassis adds additional network communication and processing to the response time.
Limit events on digital inputs to a single input bit on a module.	All inputs on a module trigger a single event, so using multiple bits increases the chance of a task overlap. Configure the module to detect change-of-state on the trigger input and turn off the other bits.
Set the priority of the event task as the highest priority on the controller.	If the priority of the event task is lower than a periodic task, the event task will have to wait for the periodic task to complete execution
Limit the number of event tasks.	Increasing the number of event tasks reduces the available CPU bandwidth and increases the chances of task overlap.

Additional Considerations for Event Tasks

Consideration	Description
Amount of code in the event task	Each logic element (for example, rung, instruction, or structured text construct) adds to scan time.
Task priority	If the event task is not the highest priority task, a higher priority task may delay or interrupt the execution of the event task.
CPS and UID instructions	If one of these instructions are active, the event task cannot interrupt the currently executing task. (The task with the CPS or UID.)
Communication interrupts	Serial port communication interrupts a task, regardless of the priority of the task.
Motion planner	The motion planner takes precedence over an event task.
Trends	Trend data collection takes precedence over an event task.
Output processing	You can disable output processing at the end of a task to reduce the amount of task processing time. With RSLogix 5000 software, version 16, the Controller Organizer displays whether outputs processing is disabled.

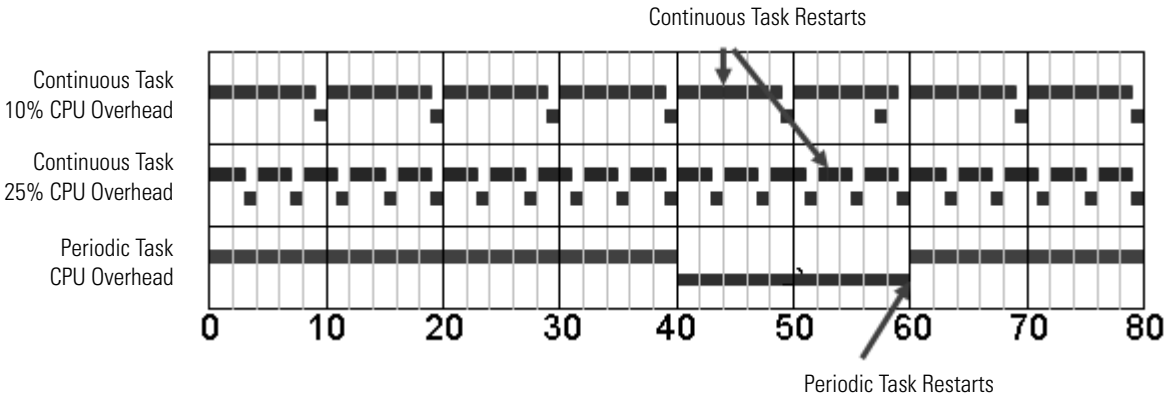
Select a System Overhead Percentage



The system overhead timeslice specifies the percentage of continuous task execution time that is devoted to communication and background redundancy functions. System overhead functions include the following:

- Communicating with programming and HMI devices (such as RSLogix 5000 software)
- Responding to messages
- Sending messages
- Serial port message and instruction processing

The controller performs system overhead functions for up to 1 ms at a time. If the controller completes the overhead functions in less than 1 ms, it resumes the continuous task. The following chart compares a continuous and periodic task.



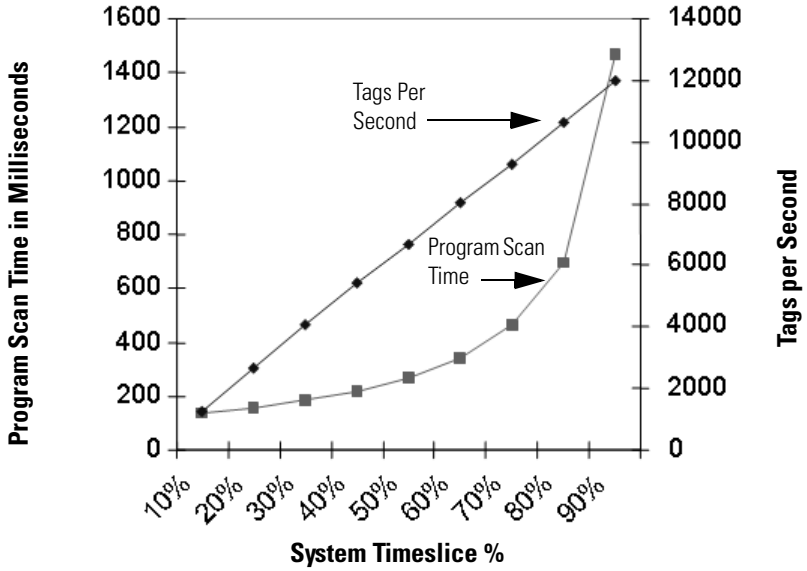
Example	Description
Continuous task 10% CPU overhead	In the top example, the system overhead timeslice is set to 10%. Given 40 ms of code to execute, the continuous task completes the execution in 44 ms. During a 60 ms timespan, the controller is able to spend 5 ms on communication processing.
Continuous task 25% CPU overhead	By increasing the system overhead timeslice to 25%, the controller completes the continuous task scan in 57 ms and spends 15 ms of a 60 ms timespan on communication processing.
Periodic task	Placing the same code in a periodic task yields even more time for communication processing. The bottom example assumes the code is in a 60 ms periodic task. The code executes to completion and then goes dormant until the 60 ms, time-based trigger occurs. While the task is dormant, all CPU bandwidth can focus on communication. Since the code takes only 40 ms to execute, the controller can spend 20 ms on communication processing. Depending on the amount of communication to process during this 20 ms window, it can be delayed as it waits for other modules in the system to process all the data that was communicated.

The Logix5000 CPU timeslices between the continuous task and system overhead. Each task switch between user task and system overhead takes additional CPU time to load and restore task information. You can calculate the continuous task interval as:

```
ContinuousTime=(100/SystemOverheadTimeSlice%) - 1
```

Manage the System Overhead Timeslice Percentage

As the system overhead timeslice percentage increases, time allocated to executing the continuous task decreases. If there are no communication for the controller to manage, the controller uses the communication time to execute the continuous task.

Consideration	Description
Impact on communication and scan time	Increasing the system overhead timeslice percentage decreases execution time for the continuous task while it increases communication performance. Increasing the system overhead timeslice percentage also increases the amount of time it takes to execute a continuous task - increasing overall scan time. 
Unused portion of system overhead timeslice	With RSLogix 5000 software, version 16, you can configure any unused portion of the system overhead timeslice to: • run the continuous task, which results in faster execution of application code and increases the variability of the program scan. • process communication, which results in more predictable and deterministic scan time for the continuous task.

Individual applications may differ, but the overall impact on communication and scan time remains the same. The above data is based on a ControlLogix5555 controller running a continuous task with 5000 tags (no arrays or user-defined structures).

Develop Application Code in Routines

Each routine contains logic in one programming language. Choose a programming language based on the application.

In general, if a section of your code represents	Then use this language
Continuous or parallel execution of multiple operations (not sequenced)	Relay ladder logic (LD)
Boolean or bit-based operations	
Complex logical operations	
Message and communication processing	
Machine interlocking	
Operations that service or maintenance personnel may have to interpret in order to troubleshoot the machine or process.	
Servo motion control	
Continuous process and drive control	Function block diagram (FBD)
Loop control	
Calculations in circuit flow	
High-level management of multiple operations	Sequential function chart (SFC)
Repetitive sequences of operations	
Batch process	
Motion control sequencing (via sequential function chart with embedded structure text)	
State machine operations	
Complex mathematical operations	Structured text (ST)
Specialized array or table loop processing	
ASCII string handling or protocol processing	

Comparison of Programming Languages

Comparison	Relay Ladder Logic	Function Block Diagram	Sequential Function Chart	Structured Text
Instruction categories	• Boolean • General and trig math • Timers and counters • Array management • Diagnostic • Serial port and messaging • ASCII manipulation • Specialty CPU control • Motion control	• General and trig math • Timers and counters • Bitwise logical • Advanced process • Advanced drive	• Step/action with embedded structured text • Transition with structure text comparisons • Simultaneous and selection branches • Stop element	• General and trig math • Timers and counters • Bitwise logical • Array management • Diagnostic • Serial port and messaging • ASCII manipulation • Specialty CPU control • Motion control • Advanced process • Advanced drive
Editor style	• Graphical rungs • Unlimited rungs	• Graphical, free-form drawing • Unlimited sheets	• Graphical, free-form drawing • Unlimited grid space	• Textual • Unlimited lines
Monitoring	• Rung animation • Data value animation • Force status	• Output and input pin data value animation	• Active steps animation • Auto display scroll • Branch/transition force status	• Tag watch pane • Context coloring
Comments	• Tag • Rung	• Tag • Text box	• Tag • Text box • Embedded structured text comments stored in CPU	• Multi-line • End if line • Comments stored in CPU

Guidelines to Pass Parameters to/from Subroutines

Guideline	Description
Input and Return parameters depend on the subroutine logic.	If the subroutine needs to know the previous state of any Return parameters (the values are used elsewhere in the project), these values should also be Input parameters: • If the subroutine contains latch/unlatch logic (holding circuits), intended outputs of the subroutine should be passed into and returned from the subroutine. • If the subroutine does not contain latch/unlatch logic, intended outputs of the subroutine only need to be returned from the subroutine.
Pass complete timers in and out of subroutines.	If a subroutine needs a timer, pass the complete timer tag to the subroutine as an input and return the complete timer tag as an output. Store the timer in a buffer tag outside of the subroutine.

Guidelines for User-defined Add-On Instructions

A user-defined Add-On Instruction is a user-created instruction that encapsulates code and local data.

Guideline	Description
Use Add-On Instructions to create standardized modules of code for reuse across a project.	Use Add-On Instructions to: • encapsulate specific or focused operations, such as a Motor or Valve action. A Conveyor or Tank action is better managed as a routine. • create extensions to the base controller instructions. For example, create an Add-On Instruction to execute an SLC 500 or PLC controller instruction not available in the Logix5000 controllers. • encapsulate an instruction from one language for use in another language. For example, create a function block PIDE instruction for use in relay ladder.
Create Add-On Instructions in relay ladder, function block diagram, or structured text languages.	Supports all Add-On Instructions and most built-in instructions. Excludes JSR/SBR/RET, JXR, FOR/BRK (relay ladder), SFR, SFP, SAR, IOT and EVENT) instructions. GSV/SSV instructions in an Add-On Instruction cannot reference the Module, Message, Axis, Motion Group, or Coordinate System class names. Add-On Instructions support function block, relay ladder, and structured text programming languages. Each of the Add-On Instruction logic areas can be any language. For example, the main logic can be function block and the prescan logic can be relay ladder. You can nest Add-On Instructions seven levels deep.

Guideline	Description
An Add-On Instruction supports parameters: • Input (copied in) • Output (copied out) • InOut (passed by reference)	• Limited to 512 total: Input parameter + Output parameter + local tags (no limit on the number of InOut parameters) • 2 MB maximum data instance (parameters and locals) • Alarm, axis, axis group, coordinate system, message, motion group, and produced/consumed tags must exist at the program or controller scope and passed as an InOut parameter • Can include references to controller-scoped tags, program-scoped tags, and immediate values. • Input and Output parameters are limited to atomic (BOOL, SINT, INT, DINT, REAL) data types. Use the InOut parameter for LINT, user-defined, and structure data types. • DINT data types provide optimal execution. • Default values of parameters and local tags are used to initialize the data structure when a tag is created of the instruction's data type. When an existing parameter or local tag's default value is modified, the existing tag instances for that instruction are not updated. When a parameter or local tag is added to the instruction definition, the tag's default value is used in the existing tags.
Create and modify offline only.	Online operation supports monitoring. Modifications to Add-On Instructions are made offline. Make changes once to the Add-On Instruction definition to affect all instances.
An Add-On Instruction executes like a routine.	A task with a higher execution priority can interrupt an Add-On Instruction. Use a UID/UIE instruction pair to block a task switch to check that an Add-On Instruction completes execution before switching to another task.
The code within an Add-On Instruction can access data specified only via parameters or defined as local.	Copy the local data to a parameter if you want to programmatically access it outside of an Add-On Instruction.
Use optional Scan mode logic to setup, initialize, or reset the Add-On Instruction code.	An Add-On Instruction can have logic in addition to the main logic for the instruction. • Prescan logic executes on controller startup. • Postscan logic executes on SFC Automatic reset. • EnableInFalse logic executes when rung condition is false.

Comparison of Subroutines and Add-On Instructions

Comparison	Subroutine	Add-On Instructions
Accessibility	Within program (multiple copies)	Anywhere in controller (single copy)
Parameters	Pass by value	Pass by value or reference via InOut
Numeric parameters	No conversion, user must manage	Automatic data type conversion for Input and Output parameters InOut parameters must match declared type exactly
Parameters data types	Atomic, arrays, structures	• Atomic data types as In or Out parameters • LINT, user-defined, and structure data types as InOut parameters
Parameter checking	None, user must manage	Verification checks
Data encapsulation	All data at program or controller scope (accessible to anything)	Local data is isolated (only accessible within instruction)
Monitor/debug	Logic animated with mixed data from multiple calls	Logic animated with data from a single calling instance
Supported programming languages	FBD, LD, SFC, ST	FBD, LD, ST
Callable from	FBD, LD, SFC, ST	FBD, LD, SFC, ST
Protection	Locked and View Only	Locked and View Only
Documentation	Routine, rung, textbox, line	Instruction description, revision information, vendor, rung, textbox, line, extended help
Execution performance	• JSR/SBR/RTN add overhead • All data is copied	• Call is more efficient • InOut passed by reference
Memory use	Very compact	• Call requires more memory • All references need additional memory
Edit	Both code and data can be modified offline and online in a running controller	Code modifications limited to offline in the project file and require a new download Data values associated can be modified online and offline
Import/export	All routines are imported/exported in the full project .L5K file (protected routines may be excluded or encrypted) Individual LD rungs and references and tags/UDTs can be imported/exported via the .L5X file	All Add-On Instructions are imported/exported in the full project .L5K file (protected instructions may be excluded or encrypted) Individual Add-On Instruction definitions and code are imported/exported via the .L5X file

Programming Methods

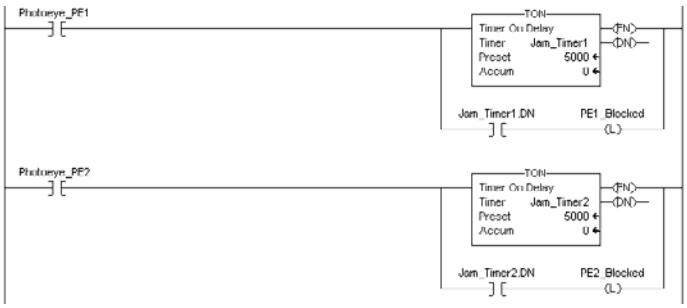
The capabilities of the Logix5000 controllers make different programming methods possible. There are tradeoffs to consider when selecting a programming method.

Inline Duplication

Benefits

- Uses more memory
- Fastest execution time because all tag references are defined before run time
- Easiest to maintain because rung animation matches tag values
- Requires more time to create and modify

Write multiple copies of the code with different tag references.

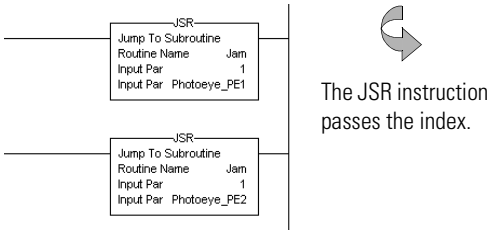


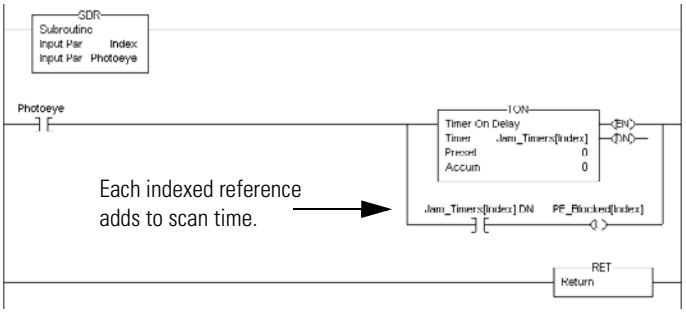
Indexed Routine

Benefits

- One copy of code is faster to develop
- Slowest execution time because all tag references are calculated at run time
- Can be difficult to maintain because the data monitor is not synchronized to execution

Write one copy of code and use indexed references to data stored in arrays.



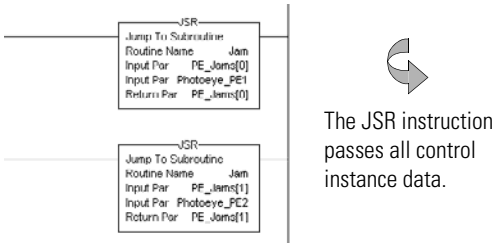


Buffered Routine

Benefits

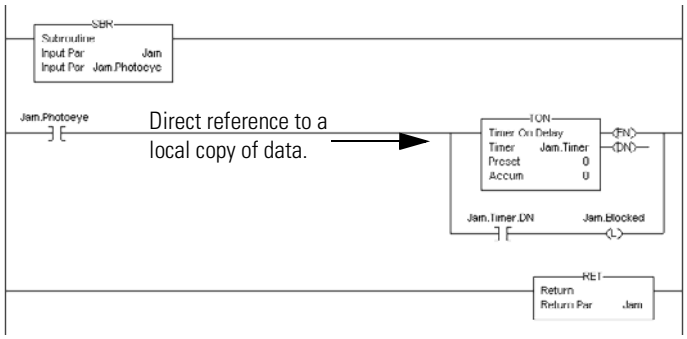
- One copy operation can occur faster than multiple index offsets
- Eliminates the need to calculate array offsets at run time
- The amount of code increases, but so do the benefits
- Can be difficult to maintain because the data monitor is not synchronized to execution

Copy the values of an array into tags and reference these buffer tags directly.



The JSR instruction passes all control instance data.

A user-defined structure consolidates control data.



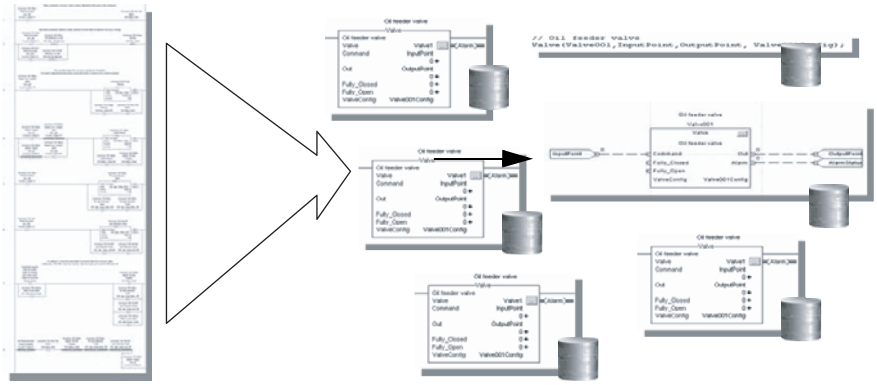
Add-On Instruction

Benefits

- User-defined instruction that is re-usable in one or multiple projects
- Encapsulate code like a routine that can be instantiated multiple times
- Each instance of the instruction has its own backing data

An Add-On Instruction encapsulates logic into a reusable user-define instruction.

Reuse Add-On Instructions throughout the project.



Controller Prescan of Logic

On power-up, the controller prescans logic to initialize instructions. The controller resets all state-based instructions, such as outputs (OTE) and timers (TON). Some instructions also perform operations during prescan. For example, the ONSR instructions turns off the storage bit. For information on prescan, see:

- Logix5000 Controllers General Instructions Reference Manual, publication 1756-RM003.
- Logix5000 Controllers Process Control and Drives Instructions Reference Manual, publication 1756-RM006.

During prescan, input values are not current and outputs are not written.

Prescan Affects	Description
Relay ladder logic	The controller resets non-retentive I/O and internal values.
Function block diagram logic	In addition to resetting non-retentive I/O and internal values, the controller clears the EnableIn parameter for every function block diagram.
Structured text logic	The controller resets bit tags and forces numeric tags to zero (0). Use the bracketed assignment operator ([:=]) to force a value to be reset during prescan. If you want a tag left in its last state, use the non-bracketed assignment operator (:=).
Sequential function chart logic	Embedded structured text follows the same rules as listed above.
Array indexed values	Array index values can fault the controller during prescan. If an array index value is larger than the dimension of the array, the controller will detect a major fault during prescan. To avoid this, make sure the index is always set properly or use a fault routine to handle this error during prescan. See Prescan of an Array Index on page 46.

Prescan differs from first scan in that the controller does not execute logic during prescan. The controller executes logic during first scan. The controller sets S:FS for one scan:

- during the first scan that follows prescan.
- during the first scan of a program when it has been uninhibited.
- each time a step is first scanned (when step.FS is set). You can view the S:FS bit being set only from the logic contained in actions that execute during the first scan of their parent step (N, L, P, and P1).

Add-On Instruction Prescan Logic

An Add-On Instruction prescan logic executes after the main logic executes in Prescan mode. Use the prescan logic to initialize tag values prior to execution. For example, set a PID instruction to Manual mode with a 0% output prior to its first execution.

When an Add-On Instruction executes in Prescan mode, any required parameters have their data passed.

- Values are passed to Input parameters from their arguments in the instruction call.
- Values are passed out of Output parameters to their arguments defined in the instruction call.

Controller Postscan of SFC Logic

SFCs support an automatic reset option that performs a postscan of the actions associated with a step once a transition indicates that the step is completed. Also, every Jump to Subroutine (JSR) instruction causes the controller to postscan the called routine. During this postscan:

- output energize (OTE) instructions are turned off and non-retentive timers are reset.
- in structured text code, use the bracketed assignment operator ([:=]) to have tags reset.
- in structured text code, use the non-bracketed assignment operator (:=) to have tags left in their last state.

Add-On Instruction Postscan Logic

When an Add-On Instruction is called by logic in an SFC Action and the Automatic Reset option is set, the Add-On Instruction executes in Postscan mode. An Add-On Instruction postscan routine executes after the main logic executes in Postscan mode. Use the postscan logic to reset internal states and status values or to disable instruction outputs when the SFC action completes.

Timer Execution

Timers in the PLC, SLC, and Logix5000 controllers all store off a portion of the real-time clock each time they are scanned. The next time through, they compare this stored value against the current clock and then adjust the ACC value by the difference.

PLC/SLC Controller	Logix5000 Controller
In a PLC/SLC controller, the timers stores 8 bits at 10 ms/bit. This allows 2.56 seconds (2**8 / 100) of padding before a timer overlaps.	A Logix5000 controller uses native 32-bit data, so there is more space to store the time. The timer stores 22 bits at 1 ms/bit, which equates to 69.905 minutes (2**22 / 1000 ms per second / 60 seconds per minute).
If program execution skips timers, it appears as if the timers pause. Actually, the timers are overrunning themselves. Depending on when the timer logic next executes, the lost time varies between 0 and 2.56 seconds.	If program execution skips timers, it takes longer than in PLC/SLC controllers to overrun the timers. This results in a larger jump in lapsed time when the timer code next executes.

Program execution can skip executing timers due to:

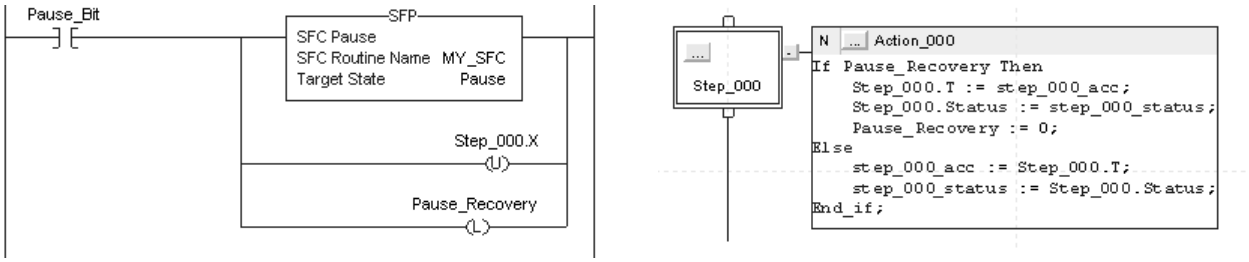
- a subroutine not being called.
- jumping over code.
- an SFC action.
- an inactive SFC step.
- an event or periodic task not executing.
- an equipment phase state routines.

SFC Step Timer Execution

An SFC step timer stores the clock time each time the step executes. On subsequent scans of the step, the controller compares the current clock time with the last scan and updates the step timer's ACC by the difference.

When you pause an SFC and then release the SFC, the step timer jumps forward by the duration of the pause. If you want a step timer to remain at its position during a pause:

- latch a recovery bit when the chart pause is released.
- add an action to the step to store the step timer's .ACC value and restore that value when the pause recovery bit is set.



Edit an SFC Online

Firmware revision 13 adds support for editing SFCs online. When you edit an SFC online, the software initially makes the changes in the offline project. When you accept the changes, they are downloaded to the controller. If you transition the controller to test or un-test edits, the controller resets the SFC and starts execution at the initial step. If you edit an SFC online:

- plan when you test or untest edits to coincide with the SFC executing the initial step.
- place structured text logic in subroutines to minimize the impact of online edits.
- use an SFR instruction to programmatically shift SFC execution to the desired step.

In some cases, this can result in the SFC being out of sync with the equipment. Program logic in the initial step to check the last state and use an SFR instruction to change to the appropriate step, if needed. One method is to set an index number in an action of each step. Then when the restart occurs, use the SFR instruction to jump to appropriate step based on the index value.

Notes:

Address Data

Introduction

Logix5000 controllers support IEC 61131-3 atomic data types, such as BOOL, SINT, INT, DINT, LINT, and REAL. The controllers also support compound data types, such as arrays, predefined structures (such as counters and timers) and user-defined structures.

Data Type	Description	
Atomic data type (BOOL, SINT, INT, DINT, REAL)	**Benefit**	**Consideration**
	• Individual names • No limit to the number of tags • Tag Editor and Data Monitor can filter individual tags and display any references • Always listed alphabetically in the Tag Editor and Data Monitor • Full alias tag support (both the base tag and its bits) • Can be added when programming online • Supported as In or Out parameter in an Add-On Instruction	• Each tag uses 32 bits of memory • Require more communication overhead and, potentially, more controller memory than compound data types • Can only change a tag's data type when programming offline • The root tag is listed alphabetically in the Tag Editor and Data Monitor, but the structure members are listed in the order in which they were defined in the structure
Special-use atomic data type (LINT)	**Benefit**	**Consideration**
	• 64-bit integer value to store date and time values • Data monitor display radix for Date and Time lets you display a LINT value as year, month, day, hours, minutes, seconds, microseconds	• Limited instruction support: GSV, SSV, ALMD, ALMA, COP, and CPS • For math operations or comparisons, copy the LINT value into a pair of DINTs and then manipulate through code • Limited to InOut parameter in an Add-On Instruction
Compound data type (array, structure)	**Benefit**	**Consideration**
	• Allow for specific names and user-defined organization • Consolidates information in controller memory • Optimizes communication time and memory impact • Arrays can be dynamically indexed • Can create new arrays when programming online • Alias support for user-defined structures, members of an array, and bits of a member	• 2 MB data limit per user-defined structure or array • User-defined structures are padded to enforce 32-bit data alignment • Alias tags cannot point to the root tag of an array • Tag Editor and Data Monitor filtering limited • Can only create or change a user-defined structure when programming offline • Can only change an array when programming offline • Limited to InOut parameter in an Add-On Instruction

The Logix CPU reads and manipulates 32-bit data values. All data starts at 32-bit offsets, so the minimum memory allocation for a tag is 4 bytes. When you create a standalone tag that stores data that is less than 4 bytes, the controller allocates 4 bytes, but the data only fills the part it needs.

Data Type	Byte 3	Byte 2	Byte 1	Byte 0					
	31...0	31...0	31...0	31	16	15	8	7 1	0
BOOL	Not used	Not used	Not used	Not used					0 or 1
SINT	Not used	Not used	Not used	Not used				-128...127	
INT	Not used	Not used	Not used	Not used		-32,768...32,767			
DINT	Not used	Not used	Not used	-2,147,483,648...2,147,483,647					
REAL	Not used	Not used	Not used	$-3.40282347E^{38}$...$-1.17549435E^{-38}$ (negative values) 0 $1.17549435E^{-38}$...$3.40282347E^{38}$ (positive values)					
LINT	Valid Date/Time range is from 1/1/1970 12:00:00 AM coordinated universal time (UTC) to 1/1/3000 12:00:00 AM UTC								

To manipulate SINT or INT data, the controller converts the values to DINT values, performs the programmed manipulation, and then returns the result to a SINT or INT value. This requires additional memory and execution time when compared to using DINT values for the same operation.

Guidelines for Data Types

Guideline	Description
Use DINT data types whenever possible	The Logix5000 controllers perform DINT (32 bit) and REAL (32 bit) math operations. DINT data types use less memory and execute faster than other data types. Use: • DINT for most numeric values and array indexes. • REAL for manipulating floating-point, analog values. • SINT (8 bit) and INT (16 bit) primarily in user-defined structures or when communicating with an external device that does not support DINT values.

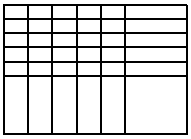
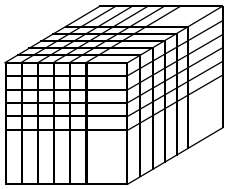
	SINT	INT	DINT	REAL
Memory reserved for a standalone tag	4 bytes	4 bytes	4 bytes	4 bytes
Memory reserved for data in a user-defined structure	1 byte (8-bit aligned)	2 bytes (16-bit aligned)	4 bytes (32-bit aligned)	4 bytes (32-bit aligned)
Memory used to access a tag in an ADD instruction	236 bytes	260 bytes	28 bytes	44 bytes
Execution time on a 1756-L63 controller required to perform an ADD instruction	3.31 μs	3.49 μs	0.26 μs	1.45 μs

Group BOOL values into arrays	When using BOOL values, group them into DINT arrays to best use controller memory and to make the bits accessible via FBC or DDT instructions.

Arrays

An array allocates a contiguous block of memory to store a specific data type as a table of values.

- Tags support arrays in one, two, or three dimensions.
- User-defined structures can contain a single-dimension array as a member of the structure.

This array	Stores data like	For Example				
One dimension		Tag name	Type	Dimension 0	Dimension 1	Dimension 2
		one_d_array	DINT[7]	7	--	--
		Total number of elements = 7 Valid subscript range DINT[*x*] where *x*=0...6				
Two dimension		Tag name	Type	Dimension 0	Dimension 1	Dimension 2
		two_d_array	DINT[4,5]	4	5	--
		Total number of elements = 4 * 5 = 20 Valid subscript range DINT[*x*,*y*] where *x*=0...3; *y*=0...4				
Three dimension		Tag name	Type	Dimension 0	Dimension 1	Dimension 2
		three_d_array	DINT[2,3,4]	2	3	4
		Total number of elements = 2 * 3 * 4 = 24 Valid subscript range DINT[*x*,*y*,*z*] where *x*=0...1; *y*=0...2, *z*=0...3				

The data type you select for an array determines how the contiguous block of memory gets used.

BOOL[96] = 12 bytes

BOOL arrays use 32-bit increments of memory

31	30	29	27	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64

SINT[10] = 12 bytes of memory (2 bytes unused)

SINT arrays are padded to use any left over bytes

3	2	1	0
7	6	5	4
Unused	Unused	9	8

INT[5] = 12 bytes of memory (2 bytes unused)

INT arrays are padded to use any left over bytes

1	0
3	2
Unused	4

DINT[3] = 12 bytes and REAL[3] = 12 bytes

DINT and REAL arrays use 4-byte increments of memory

0
1
2

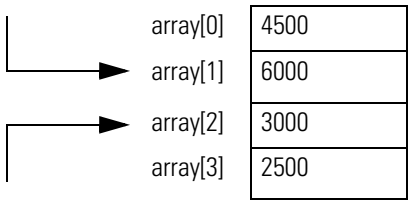
Guidelines for Arrays

<table>
<tr><th colspan="2">Guideline</th><th colspan="2">Description</th></tr>
<tr><td colspan="2">You can create arrays of most data types, except for ALARM, AXIS, COORDINATE_SYSTEM, MOTION_GROUP, and MESSAGE data types.</td><td colspan="2">A subscript identifies an individual element within the array. A subscript starts at 0 and extends to the number of elements minus 1 (zero based).
• Single-dimension arrays take less memory and execute faster than two-dimension or three-dimension arrays.
• Direct references to array elements execute faster than indexed references.
• An array can be as large as 2 MB.
• If you create an array of structures, the memory for each element is allocated based on the structure definition.</td></tr>
<tr><td></td><td>Type of Array</td><td>Benefit</td><td>Considerations</td></tr>
<tr><td></td><td>Single (1) dimension</td><td>• Better support by native file instructions
• Fully supported in user-defined structures and arrays
• Smallest impact (execution time and memory) for indexed references
• Can create new arrays when programming online</td><td>• Multiple arrays cannot be indirectly referenced like in PLC or SLC processors (such as, N[N7:0]:5)
• BOOL arrays not directly supported by file instructions
• Can be changed only when programming offline</td></tr>
<tr><td></td><td>Double (2) dimension
and
Triple (3) dimension</td><td>• Can provide a more accurate data representation for a physical system
• Can emulate PLC file/word indirection with a two-dimension array
• Can create new arrays when programming online</td><td>• Larger impact (execution time and memory) for indexed references
• File manipulation requires extra code in addition to file instructions
• Can only be changed when programming offline</td></tr>
<tr><td colspan="2">Nest arrays.</td><td colspan="2">The file instructions offer limited support for arrays. To work with array data, create a user-defined structure with one array as a member of the structure. Then create an array tag by using the user-defined structure as its data type.</td></tr>
<tr><td colspan="2">Select the data type of the array based on the data, as well as the instructions that manipulate that data.</td><td colspan="2">While SINT and INT arrays can compact more values into a given memory area, they require additional memory and execution time for each instruction that references the array.</td></tr>
<tr><td colspan="2">Limit arrays to 2 MB of data.</td><td colspan="2">The maximum array size is 2 MB. The software displays a warning if you try to create an array that is too large. The software also displays a warning if an array is 1.5...2 MB in size, even though these sizes are valid.</td></tr>
<tr><td colspan="2">Edit arrays online and offline.</td><td colspan="2">You can create new arrays when online or offline. However, you can modify only the size or data type of an existing array when offline.</td></tr>
</table>

Indirect Addresses of Arrays

If you want an instruction to access different elements in an array, use a tag in the subscript of the array (an indirect address). By changing the value of the tag, you change the element of the array that your logic references.

When *index* equals 1, *array[index]* points here.



When *index* equals 2, *array[index]* points here.

Directly referencing an element in an array (such as MyArray[20]), uses less memory and executes faster than an indirect reference (MyArray[MyIndex]). You can also indirectly address bits in a tag (MyDint.[Index]).

If you use indirect addresses, use DINT tags because other data types require conversion and execute slower. For each indexed access to data, the controller recalculates the array index. If you access a specific array element multiple times, copy the data out of the array into a fixed tag and use that tag in subsequent logic.

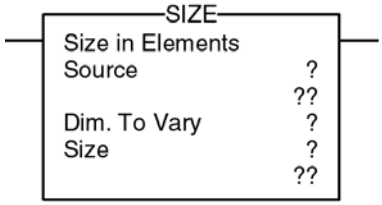
You can also use an expression to specify the index value. For example: MyArray[10 + MyIndex].

- An expression uses operators to calculate a value.
- The controller computes the result of the expression and uses it as the index.
- These are valid operators.

Operator	Description	Optimal
+	Add	DINT, REAL
-	Subtract/negate	DINT, REAL
*	Multiply	DINT, REAL
/	Divide	DINT, REAL
**	Exponent (x to y)	DINT, REAL
ABS	Absolute value	DINT, REAL
ACS	Arc cosine	REAL
AND	Bitwise AND	DINT
ASN	Arc sine	REAL
ATN	Arc tangent	REAL
COS	Cosine	REAL
DEG	Radians to degrees	DINT, REAL
FRD	BCD to integer	DINT

Operator	Description	Optimal
LN	Natural log	REAL
LOG	Log base 10	REAL
MOD	Modulo divide	DINT, REAL
NOT	Bitwise complement	DINT
OR	Bitwise OR	DINT
RAD	Degrees to radians	DINT, REAL
SIN	Sine	REAL
SQR	Square root	DINT, REAL
TAN	Tangent	REAL
TOD	Integer to BCD	DINT
TRN	Truncate	DINT, REAL
XOR	Bitwise exclusive OR	DINT

Guidelines for Array Indexes

Guideline	Description
Use the SIZE instruction to determine the number of elements in an array.	By determining the number of elements in an array at run time, you can write reusable code that adjusts itself to meet each instance where it is used.  The SIZE instruction returns the number of elements. Arrays are zero-based, so subtract 1 from the result to determine the last element position.
Use immediate values to reference array elements.	Immediate value references to array elements are quicker to process and execute faster than indexed references.
Use DINT tags for array indexes.	DINT tags execute the fastest. SINT, INT, and REAL tags require conversion code that can add additional scan time to an operation.
Avoid using array elements as indexes.	The Logix5000 controller does not directly support the use of an array element as the index to look up a value in another array. To work around this, you can create an alias to the element and then use this as the index. Or copy the element to a base tag and use that base tag as the index.

Prescan of an Array Index

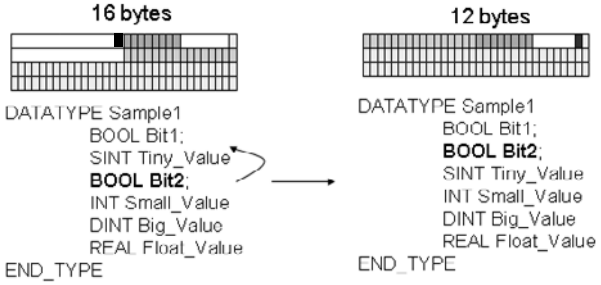
During prescan, the controller resets state based on instructions such as outputs and timers. If you use calculated array indexes based on program execution, an Indexed address out of range error occurs because the program has not executed and the index was not initialized. Use a fault handler routine.

- Place an unconditional rung with an OTE instruction referencing an internal bit in the first program of the first task. During prescan, the prescan bit will be turned off. During normal scan, the prescan bit will be on at all times.
- Indexed address out of range error occurs and the prescan bit is off, reset the error and continue.

See the Logix5000 Controllers Common Procedures Programming Manual, publication 1756-PM001, for information and sample code to handle faults.

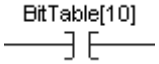
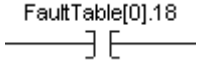
IMPORTANT This prescan condition no longer exists in controllers with firmware revision 13 and later. You do not need to program a fault handler routine to handle indexed address out-of-range errors.

Guidelines for User-defined Structures

Guideline	Description
Group members of the same data type within a structure.	You can create members of most data types, except for ALARM, AXIS, COORDINATE_SYSTEM, MOTION_GROUP, and MESSAGE data types. Place members that use the same data type in sequence.  A Logix5000 controller aligns every data type along an 8-bit boundary for SINTs, a 16-bit boundary for INTS, or a 32-bit boundary for DINTs and REALs. BOOLs also align on 8-bit boundaries, but if they are placed adjacent to each other in a user-defined structure, they are mapped so that they share the same byte.
Arrays within structures can only be 1-dimension.	If you include an array as a member, limit the array to a single dimension. Multidimension arrays are not permitted in a user-defined structure.
I/O data used in structure must be copied into the members.	If you include members that represent I/O devices, you must use logic to copy the data into the members of the structure from the corresponding I/O tags. Make sure the data type of the structure member matches the I/O data type to avoid data type conversion.
Limit user-defined structures to 500 members.	Logix5000 controllers limit user-defined structures to 500 members. If you need more, consider nesting structures within the main structure.
Limit user-defined structures to 2 MB of data.	The maximum UDT size is 2 MB. The software displays a warning if you try to create an UDT that is too large. The software also displays a warning if the UDT is 1.5...2 MB in size, even though these sizes are valid.
Limit the size of user-defined structures if they are to be communicated.	Produced and consumed tags are limited to 500 bytes over the backplane and 480 bytes if over a network. RSLinx software can optimize user-defined structures that are less than 480 bytes.
Use the appropriate instruction to load data into a structure.	Load input values into the user-defined structure at the beginning of the program and copy output values from the user-defined structure at the end of the program. • Single bit - Examine On (XIC) and Output Energize (OTE) instructions • Contiguous bits - Bit Field Distribute (BTD) instruction • Single value - Copy (COP) instruction • Multiple contiguous values - Synchronize Copy (CPS) instruction
Use structure descriptions to automatically create tag descriptions.	Enable the Use Pass-through Description workstation option (Tools > Options > Display) to display the descriptions you add to the members of structures for each tag that uses that structure data type.
Online and offline editing.	You can create new user-defined structures when online or offline. However, you can modify only an existing structure when offline.

Select a Data Type for Bit Tags

Bits in a Logix5000 controller can exist as: BOOL tags, bits in a BOOL array, bits in elements of a SINT, INT, DINT array, members of a user-defined structure, or as bits in a SINT, INT, DINT member of a user-defined structure.

Tag Type	Description	
BOOL tag	Each tag accesses a specific bit. Each tag uses 4 bytes.	
	Benefits	**Considerations**
MyBit:BOOL 	• Each bit has a specific tag	• Requires extra bandwidth to communication • Uses more memory; 32-bits for each tag • Cannot use FBC/DDT bit file instructions
BOOL array	A BOOL array combines multiple bits into adjacent words (32-bit words).	
	Benefits	**Considerations**
BitTable:BOOL[32] 	• Consolidates multiple bits into a single word • Better use of memory • Can address all bits in an array using indirect addressing	• BOOL data type only supported by bit instructions • Cannot use file instructions, copy instructions, or DDT/FBC instructions
DINT array	A DINT combines multiple bits into adjacent words.	
	Benefits	**Considerations**
FaultTable:DINT[3] 	• Consolidates multiple bits into a single word • File instructions, copy instructions, and DDT/FBC instructions support DINT arrays • Lets you access the bits by element (word) and bit number	• Requires extra planning to indirectly address bits • Difficult to address bits in the array using indirect addressing
User-defined structure	A user-defined structure combines multiple bits into adjacent, individually-named words.	
	Benefits	**Considerations**
BitStructure Bit1:BOOL Bit2:BOOL Fault:BitStructure 	• Object based • Consolidates multiple bits into a single word	• Structures are not directly supported by 3rd party MMI/EOI products (RSView does support 32-bit tags and structures) • Cannot use FBC/DDT bit file instructions

Serial Bit Addresses

The BOOL B data table in the PLC-5 and SLC 500 processors supports two address modes that can address the same bit.

Address Mode	Description
Serial bit In PLC-5 or SLC software, this addressing mode is represented as /Bit	Serial bit addressing provides the ability to reference all bits as a contiguous list (array) of bits. For example, if you want to reference the third bit in the second word of a B file, specify B3/18. This method is similar to a BOOL array in a Logix5000 controller where you would specify FaultBit[18].
Word bit In PLC-5 or SLC software, this addressing mode is represented as Word/Bit	Word bit addressing identifies a bit within a specific word. For example, B3:1/2 is the same as B3/18 from the serial bit example. This method is similar to accessing the bits of a SINT, INT, DINT array in a Logix5000 controller where you would specify FaultTable[1].2.

The Logix5000 controller supports both of these addressing modes, but you cannot use both to reference bits in the same array due to conformance with the IEC 61131-3 standard. Choose the method that best meets your application needs. You can copy data between arrays using both methods.

You can also use an expression to indirectly reference a bit in a DINT array using a serialized bit number. For example:

```
Tag
       MyBits : DINT[10]
       BitRef : DINT
EndTag

MOV(34, BitRef)
XIC(MyBits[BitRef / 32].[BitRef AND 31])
```

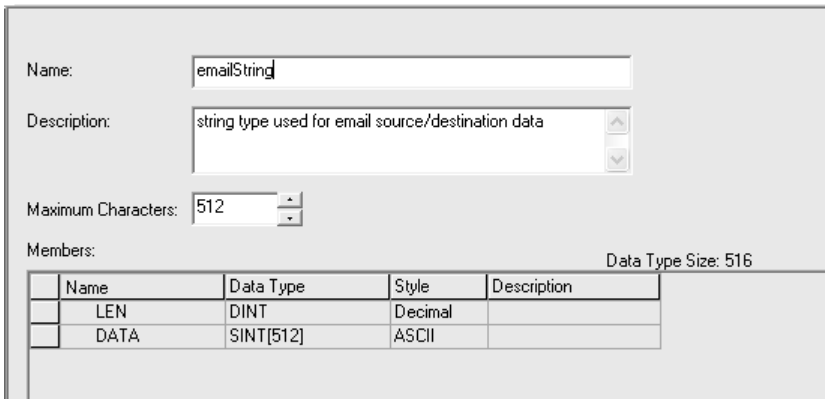
where:

This expression	Calculates the
`[BitRef / 32]`	Element in the DINT array
If the tag MyBits is an INT or SINT, the divisor would be 16 or 8, respectively.	
`[BitRef AND 31]`	Bit within the element
If the tag MyBits is an INT or SINT, the mask value would be 15 or 7, respectively.	

The Diagnostic Detect (DDT) and File Bit Compare (FBC) instructions provide a bit number as a result of their operation. These instructions are limited to DINT arrays so you can use them to locate the bit number returned from the example above.

Guidelines for String Data Types

String data types are structures that hold ASCII characters. The first member of the structure defines the length of the string; the second member is an array that holds the actual ASCII characters.



Guideline	Description
You can create a string data type that is longer or shorter than the default string data type.	The default string data type can contain as many as 82 characters, but you can create custom-length string data types to hold as many characters as needed.
Only some instructions support string data types.	These comparison instructions support string tags: EQU, NEQ, GRT, GEG, LES, LEQ, CMP. These serial port instructions support string tags: ARD, ARL, AWA, AWT. These string-handling instructions support string tags: STOD, DTOS, STOR, RTOS, CONCAT, MID, FIND, DELETE, INSERT, UPPER, LOWER, SIZE. These file instructions support string arrays: FAL, FFL, FFU, LFL, LFU, COP, CPS, FSC.
Use the SIZE instruction to determine the number of characters in a string,	By determining the number of characters in a string at run time, you can write reusable code that adjusts itself to meet each instance where it is used.
Using the DTOS, RTOS, and CONCAT instructions, you can embed tag values within a string.	The SLC 500 processor supports the ability to embed a data-table reference address within a string (inline indirection). The SLC 500 AWA and AWT instructions can then look up the data value and place an ASCII representation into the outgoing string. The Logix5000 controller does not directly support this ability. Use the DTOS or RTOS instructions to convert a value to a string and the CONCAT instruction to merge characters with another string.
Set the LEN field to indicate the number characters that are present.	The LEN field in the string structure indicates how many characters are in the string. RSLogix 5000 software and the controller instructions that manipulate strings use the LEN value to determine how many positions in the string DATA array contain valid characters. Both RSLogix 5000 software and the instructions stop processing the DATA array once they reach the LEN value.

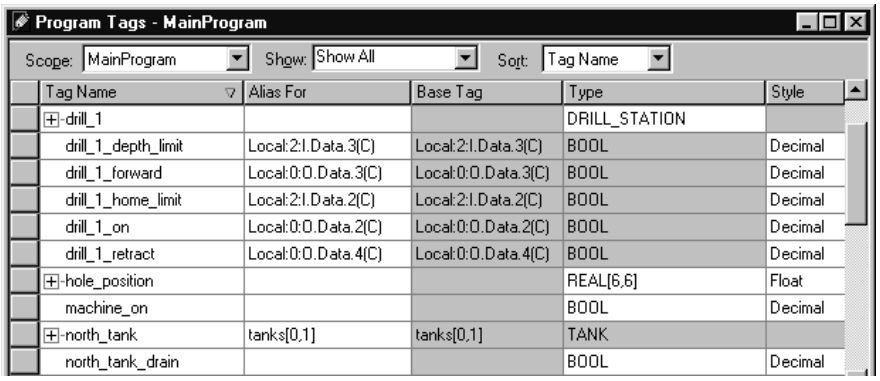
PLC-5/SLC 500 Access of Strings

The ASCII A data table in the PLC-5 and SLC 500 processors uses a string format that is similar to the Logix string data type. The main difference is that the LEN field (length) in a PLC-5/SLC 500 processor is a 16-bit, INT value whereas the LEN field in a Logix5000 controller is a 32-bit, DINT field. This difference can impact converted logic and data communication. The Logix5000 controller will convert the LEN field to the appropriate value and size when a PLC-5/SLC 500 message format is used to read or write a string.

Configure Tags

A tag is a text-based name for an area of the controller's memory where data is stored. Tags are the basic mechanism for allocating memory, referencing data from logic, and monitoring data.

If you want the tag to	Then choose this type
Store a value for use by logic within the project	Base
Use a different name for an existing tag's data (can help simplify long, pre-determined tag names, such as for I/O data or user-defined structures)	Alias
Send (broadcast) data to another controller	Produced
Receive data from another controller	Consumed



Tag Name	Alias For	Base Tag	Type	Style
+-drill_1			DRILL_STATION	
drill_1_depth_limit	Local:2:I.Data.3(C)	Local:2:I.Data.3(C)	BOOL	Decimal
drill_1_forward	Local:0:O.Data.3(C)	Local:0:O.Data.3(C)	BOOL	Decimal
drill_1_home_limit	Local:2:I.Data.2(C)	Local:2:I.Data.2(C)	BOOL	Decimal
drill_1_on	Local:0:O.Data.2(C)	Local:0:O.Data.2(C)	BOOL	Decimal
drill_1_retract	Local:0:O.Data.4(C)	Local:0:O.Data.4(C)	BOOL	Decimal
+-hole_position			REAL[6,6]	Float
machine_on			BOOL	Decimal
+-north_tank	tanks[0,1]	tanks[0,1]	TANK	
north_tank_drain			BOOL	Decimal

For more information on I/O tags, see Chapter 5 Communicate with I/O.

Guidelines for Base Tags

Guideline	Description
Create standalone atomic tags.	The controller supports pre-defined, standalone tags. • Atomic tags are listed directly in the Tag Editor and Data Monitor and can be easily located by browsing the alphabetical list. • Atomic tags can be created online, but the data type can be only modified offline. Using only atomic tags can impact HMI communication performance as more information must be passed and acted on.
Create user-defined structures	User-defined structures (data types) let you organize your data to match your machine or process. • One tag contains all the data related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type. • Each piece of data (member) gets a descriptive name. • You can use the structure to create multiple tags with the same data layout. • User-defined structure can only be modified offline. RSLinx software optimizes user-defined structures more than standalone tags.
Use arrays like files to quickly create a group of similar tags.	An array creates multiple instances of a data type under a common tag name. • Arrays let you organize a block of tags that use the same data type and perform a similar function. • You organize the data in one, two, or three dimensions to match what the data represents. • Arrays can be only modified offline. • RSLinx software optimizes array data types more than standalone tags. Minimize the use of BOOL arrays. Many array instructions **do not** operate on BOOL arrays. This makes it more difficult to initialize and clear an array of BOOL data.
Take advantage of program-scoped tags.	If you want multiple tags with the same name, define each tag at the program scope (program tags) for a different program. This lets you reuse both logic and tag names in multiple programs. Avoid using the same name for both a controller tag and a program tag. Within a program, you cannot reference a controller tag if a tag of the same name exists as a program tag for that program.
Use mixed case and the underscore characters.	Although tags are not case sensitive (upper case **A** is the same as lower case **a**), mixed case is easier to read. For example, Tank_1 can be easier to read than tank1.
Consider alphabetical order.	RSLogix 5000 software displays tags of the same scope in alphabetical order. To make it easier to monitor related tags, use similar starting characters for tags that you want to keep together. For example, consider using Tank_North and Tank_South rather than North_Tank and South_Tank.
Use leading zeroes (0) when numbers are part of tag names	RSLogix 5000 software uses a simple sort to alphabetize tag names in the Tag Editor and Data Monitor. This means if you have Tag1, Tag2, Tag11, and Tag12, the software displays them in order as Tag1, Tag11, Tag12, and then Tag2. If you want to keep them in numerical order, name them Tag01, Tag02, Tag11, and Tag12.

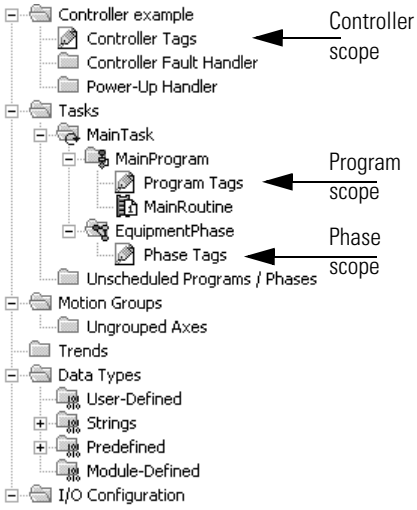
Create Alias Tags

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

- Both tags share the same value as defined by the base tag.
- When the value of a base tag changes, all references (aliases) to the base tag reflect the change.

Guideline	Description
An alias tag references a base tag.	When assigning aliases, avoid: • nesting aliases (you cannot have an alias of an alias). • using multiple aliases to the same tag. On upload, the software decompiles the program and uses the physical memory addresses to determine which tags are referenced in the code. All references to a base tag reverts to an alias if one exists. If multiple aliases point to the same tag, RSLogix 5000 software uses the first alias tag (alphabetically) that it finds.
Alias tags do not affect controller execution.	During download, the program is compiled into machine executable code and physical memory addresses. While the existence of an alias requires controller memory to store the name, the program performs the same operation for a reference with an alias or its associated base tag.
Access alias tags from RSLinx software.	Because an alias tag appears as a standalone tag to RSLinx software, an alias tag that references a compound array or structure might require additional communication time. When referencing tags from RSLinx software or other HMI, it might be fastest to reference base tags directly.

Guidelines for Data Scope



Data scope defines where you can access tags. Controller-scoped tags are accessible by all programs. Program-scoped tags are accessible only by the code within a specific program; phase-scoped tags are accessible only by the code within a specific equipment phase.

If you want to	Then assign this scope
Use a tag in more than one program in the same project	Controller scope (controller tags)
Use a tag in a message (MSG) instruction	
Produce or consume data	
Use motion tags	
Communicate with a PanelView terminal	
Reuse the same tag name multiple times for different parts or processes within a controller	Program scope (program tags) Phase scope (phase tags)
Have multiple programmers working on logic and you want to merge logic into one project	

Isolate portions of a machine or different stations into separate programs or equipment phases and use program-scoped or phase-scoped tags. This:

- provides isolation between programs and equipment phases.
- prevents tag name collisions.
- improves the ability to reuse code.

Guidelines for Tag Names

Guideline	Description
Create descriptive names but keep them short.	Tag names can be from 1...40 characters long. • Each character of the tag name uses 1 byte of controller memory, rounded to a 4-byte boundary. For example, a tag name with 1...4 characters uses 4 bytes. A tag name with 5 characters uses 8 bytes. • Tag names are stored in the controller. • Use structures to reduce the number and size of tags needed. Program upload preserves tag names.
Create a naming convention.	Develop a tag-naming convention on electrical drawings or machine design. For example, Conv1_Full_PE101 combines the sensor function with the photoeye number.
Use correct characters in tag names.	Logix5000 tag names follow the IEC 61131-3 standard. You can use: • letters A through Z. • numbers 0...9. • underscore character (_). Tags must start with a letter (this avoids confusion with logical expressions). The remaining characters can be any of the supported characters.
Pad names to improve sort order.	RSLogix 5000 software displays tags in alphabetical order. If you use numbers in your tag names, pad the number with leading zeros so the names sort in the proper order. For example, tag names: TS1, TS2, TS3, TS10, TS15, TS20, TS30 display as: TS1, TS10, TS15, TS2, TS20,TS3, and TS30. Pad the numbers with zero so they display as: TS01, TS02, TS03, TS10, TS15, TS20, TS30.

Tag Descriptions

RSLogix 5000 software searches a tag's origin to locate the first available description. This reduces the number of descriptions you need to enter. This also verifies that tag references display associated descriptions.

<table>
<tr><th>Guideline</th><th>Description</th></tr>
<tr><td>Tag descriptions display in RSLogix 5000 software according to the tag's origin.</td><td>
<table>
<tr><th>Type of Tag</th><th>Description Display in RSLogix 5000 Software</th></tr>
<tr><td>Atomic</td><td>For a BOOL, SINT, INT, DINT, or REAL tag, the description associated with the tag is the only description available for display.</td></tr>
<tr><td>Alias</td><td>First the alias tag description, then the base tag description.</td></tr>
<tr><td>User-defined structure and Add-On Instruction</td><td>All the members use the description for tag, unless you define a specific description for a member.
For example, MyTimer.DN uses the description for MyTimer if there is no description for MyTimer.DN.</td></tr>
<tr><td>Atomic array</td><td>• All references into an array use the description for the array, unless you define a description for an element of the array.
For example, MyTable[10] uses the description for MyTable if there is no description for MyTable[10].
• All indexed references into an array use the description for the array.
For example, MyTable[Index] uses the description for MyTable.</td></tr>
<tr><td>Structure array</td><td>All references to a member of a structure in an array default to the array definition, unless you define a description for the structure member of the array.
For example, Table[0].Field1 uses the description for Table if there is no description for the specific field.</td></tr>
</table>
</td></tr>
</table>

For more information, see Create Tag Descriptions Automatically with User-Defined Data Types White Paper, LOGIX-WP004.

Notes:

Share Tag Data with Other Controllers (produced and consumed tags)

Introduction

Logix5000 controllers support the ability to produce (broadcast) and consume (receive) system-shared tags.

For two controllers to share produced or consumed tags, both controllers must be attached to the same control network (such as a ControlNet or Ethernet/IP network). You cannot bridge produced and consumed tags over two networks.

Logix5000 controllers can produce and consume tags over these networks (as long as they support communication over these networks):

- The ControlLogix backplane
- A ControlNet network
- An EtherNet/IP network

If there are no other connections, the controller supports these tags.

As a	The controller support
Producer	(*number of produced tags*) ≤ 127
Consumer	(*number of consumed tags*) ≤ 250 (or controller maximum)

The total combined number of consumed and produced tags that a controller supports is:

(*produced tags*) + (*consumed tags*) + (*other connections*) ≤ 250 (or controller maximum)

IMPORTANT The actual number of produced and consumed tags that you can configure in a project depends on the connection limits of the communication module through which you produce or consume the tags.

Guidelines for Produced and Consumed Tags

Guideline	Description
You cannot bridge produced and consumed tags over different networks.	For two controllers to share produced or consumed tags, both controllers must be attached to the same network. You can produce and consume tags over ControlNet or EtherNet/IP networks.
Create the tag at controller scope.	You can only produce and consume (share) controller-scoped tags.
Limit the size of the tag to ≤ 500 bytes.	If you transfer a tag with more than 500 bytes, create logic to transfer the data in packets. If you consume a tag over a ControlNet hop, the tag must be ≤ 480 bytes. This is a limitation of the ControlNet network, not the controller.
Combine data that goes to the same controller.	If you are producing several tags for the same controller: • group the data into one or more user-defined structures. This uses less connections than producing each tag separately. • group the data according to similar update intervals. To conserve network bandwidth, use a greater RPI for less critical data.
Use one of these data types: • DINT • REAL • Array of DINTs or REALs • User-defined structure	To share data types other than DINT or REAL, create a user-defined structure to contain the required data. Use the same data type for the produced tag and the corresponding consumed tag or tags.
Use a user-defined structure to produce or consume INT or SINT data.	To produce or consume INT or SINT data, create a user-defined structure with INT or SINT members. The members can be individual INTs or SINTs or the members can be INT or SINT arrays. The resulting user-defined structure can then be produced or consumed.
The data type in the producer and the consumer must match.	The data type for a produced or consumed tag must be the same in both the producer and the consumer.
Produce tags based on user-defined structures to non-Logix devices.	The controller produces tags in 32-bit words. For devices that communicate in other word boundaries, such as 16-bit words, the resulting data in the target device can be misaligned. To help avoid misalignment, structure the produced data in a user-defined structure.
Use a programmatic handshake to ensure data is exchanged.	Produced tags continually transmit based on the RPI, so it can be difficult to know when new data arrives. You can set a bit or increment a counter embedded in the produced tag to identify to the consumer that new data is present. You can also provide a return handshake via a reverse produced/consumed tag so that the original producer knows that the tag was received and processed by the consumer.
Use a CPS instruction to buffer produced and consumed data.	Use the CPS instruction to copy the data to the outgoing tag on the producer side. Then use another CPS instruction to copy the data into a buffer tag on the consumer side. The CPS instructions provides data integrity for data structures greater than 32 bits. **Important:** The controller inhibits all interrupts while it executes a CPS instruction.
Use unicast EtherNet/IP communication to reduce broadcast network traffic.	To reduce bandwidth use and preserve network integrity, some facilities block multicast Ethernet packets. With RSLogix 5000 software, version 16, you can configure a produced/consumed tag to use multicast or unicast connections. Unicast connections: • reduce network bandwidth. • simplify Ethernet switch configuration.

Guidelines to Specify an RPI Rate for Produced and Consumed Tags

When configuring produced and consumed tags, you specify an Requested Packet Interval (RPI) rate. The RPI value is the rate at which the controller attempts to communicate with the module.

Guideline	Description
Make sure the RPI is equal to or greater than the NUT.	You use RSNetWorx for ControlNet software to select the network update time (NUT) and the software schedules the network connections. RSNetWorx software cannot schedule a ControlNet network if a module and/or produced/consumed tag on the network has an RPI that is faster than the network update time.
The smallest (fastest) consumer RPI determines the RPI for the produced tag.	If multiple consumers request the same tag, the smallest (fastest) request determines the rate at which the tag is produced for all the consumers.

Guidelines to Manage Connections for Produced and Consumed Tags

Guideline	Description
Minimize the use of produced and consumed tags.	To reduce network traffic, minimize the size of produced and consumed tags. Also, minimize the use of produced and consumed tags to high-speed, deterministic data, such as interlocks.
Use arrays or user-defined structures.	When sending multiple tags to the same controller, use an array or user-defined structure to consolidate the data. The byte limit of ≤ 500 bytes per produced and consumed tag still applies.
Configure the number of consumers accurately.	Make sure the number of consumers configured for a produced tag is the actual number of controllers that will consume the tag. If you set the number higher than the actual number of controllers, you unnecessarily use up connections. The default is two consumers per produced tag.
Multiple produced/consumed connections are linked.	If there are multiple produced and consumed connections between two controllers and one connection fails, all the produced and consumed connections fail. Consider combining all produced and consumed data into one structure or array so that you only need one connection between the controllers.

Configure an Event Task Based on a Consumed Tag

An event task executes automatically based on a preconfigured event occurring. One such event can be the arrival of a consumed tag.

- Only one consumed tag can trigger a specific event task.
- Typically, use an IOT instruction in the producing controller to signal the production of new data.
- When a consumed tag triggers an event task, the event task waits for all the data to arrive before the event task executes.

For information on configuring an event task, see Chapter 2, Divide Logic into Tasks, Programs, Routines, and Add-On Instructions.

Compare Messages and Produced/Consumed Tags

Method	Benefits	Considerations
Read/Write Message	• Programmatically initiated • Communication and network resources only used when needed • Support automatic fragmentation and reassembly of large data packets, up to as many as 32,767 elements • Some connections can be cached to improve re-transmission time • Generic CIP message useful for third-party devices	• Delay may occur if resources are not available when needed • MSG instruction and processing impacts controller scan (system overhead timeslice) • Data arrives asynchronous to program scan (use a programmatic handshake or an UID/UIE instruction pair to reduce impact, no event task support)
Produced/Consumed Tag	• Configured once and sent automatically based on requested packet interval (RPI) • Multiple consumers can simultaneously receive the same data from a single produced tag • Can trigger an event task when consumed data arrives • ControlNet resources are reserved up front • Does not impact the scan of the controller	• Support limited to Logix5000 and PLC-5 controllers, and the 1784-KTCS I/O Linx and select third party devices • Limited to 500 bytes over the backplane and 480 bytes over a network • Must be scheduled when using ControlNet • Data arrives asynchronous to program scan (use a programmatic handshake or CPS instruction and event tasks to synchronize) • Connection status must be obtained separately • On an EtherNet/IP network, you can configure produced/consumed tags to use multicats or unicast connections.

Communicate with I/O

Introduction

In Logix5000 controllers, I/O values update at a period, requested packet interval (RPI), that you configure via Module Property dialog in the I/O configuration folder of the project. The values update asynchronously to the execution of logic.

The module sends input values to the controller at the specified RPI. Because this transfer is asynchronous to the execution of logic, an I/O value in the controller can change in the middle of a scan.

Buffer I/O Data

If you reference an I/O tag multiple times and the application could be impacted if the value changes during a program scan, you must copy the I/O value into a buffer tag prior to the first reference of that tag in your code. In your code, reference the buffer tag rather than the I/O tag.

IMPORTANT Use the synchronous copy (CPS) instruction to buffer I/O data. While the CPS instruction copies data, no I/O updates or other tasks can change the data. Tasks that attempt to interrupt a CPS instruction are delayed until the instruction is done. Overuse of the CPS instruction can impact controller performance by keeping the communication task from executing.

Buffer I/O data to:

- prevent an input or output value from changing during the execution of a program. (I/O updates asynchronous to the execution of logic.)
- copy an input or output tag to a member of a structure or element of an array.
- prevent produced or consumed data from changing during the execution of a program.
- ensure all produced and consumed data arrives or is sent as a group (not mixed from multiple transfers)
- only use the CPS instruction if the I/O data that you want to buffer is greater than 32 bits (or 4 bytes) in size

Overuse of the CPS instruction can greatly impact CPU use.

If you have a user-defined structure with members that represent I/O devices, you must use logic to copy the data into the members of the structure from the corresponding I/O tags.

Guidelines to Specify an RPI Rate for I/O Modules

Configure an RPI rate per module (ControlLogix and SoftLogix) or an RPI rate per controller (CompactLogix and FlexLogix). The RPI value is the rate at which the controller attempts to communicate with the module.

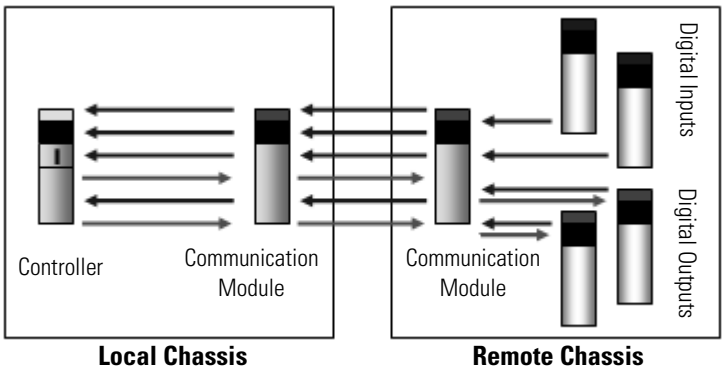
Guideline	Description
Specify an RPI at 50% of the rate you actually need.	Setting the RPI faster (specifying a smaller number) than what your application needs wastes network resources, such as ControlNet schedule bandwidth, network processing time, and CPU processing time. For example, if you need information every 80 ms, set the RPI at 40 ms. The data is asynchronous to the controller scan, so you sample data twice as often (but no faster) than you need it to make sure you have the most current data.
Group devices with similar performance needs onto the same module.	By grouping devices with similar performance needs on the same module, you consolidate data transmission to one module rather than multiple modules. This conserves network bandwidth.
Set the ControlNet network update time (NUT) equal to or less than the fastest RPI.	When configuring a ControlNet network, set the network update time (NUT) equal to or less than the fastest RPI of the I/O modules and produced/consumed tags in the system. For example, if your fastest RPI is 10 ms, set the NUT to 5 ms for more flexibility in scheduling the network.
In an ControlNet system, the RPI should be an even multiple of the NUT.	Set the RPI to a binary multiple of the NUT. For example, if the NUT is 10 ms, select an RPI such as 10, 20, 40, 80, 160 ms, and so on.
In a ControlNet system, isolate I/O communication.	If you use unscheduled ControlNet communication or want to be able to add ControlNet I/O at runtime (see page 72), dedicate one ControlNet network to I/O communication only. On the dedicated I/O network, make sure there is: • no HMI traffic. • no MSG traffic. • no programming workstations. • no peer-to-peer interlocking in a multi-processor system architectures.
In an EtherNet/IP system, module change-of-state is limited to 1/4 of the RPI.	If you configure change of state communication for a module in a remote chassis connected via an EtherNet/IP network, the module can send data only as fast as the module RPI. Initially, the module sends its data immediately. However, when an input changes, the module data is held at the adapter until 1/4 of the RPI is reached to avoid overloading the EtherNet/IP network with the module communication.
Data transmission depends on the controller.	The type of controller determines the data transmission rate. • ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module. • CompactLogix and FlexLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, or 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms.

Communication Formats for I/O Modules

The communication format determines whether the controller connects to the I/O module via a direct or a rack-optimized connection. The communication format also determines the type and quantity of information that the module will provide or use.

Direct connection

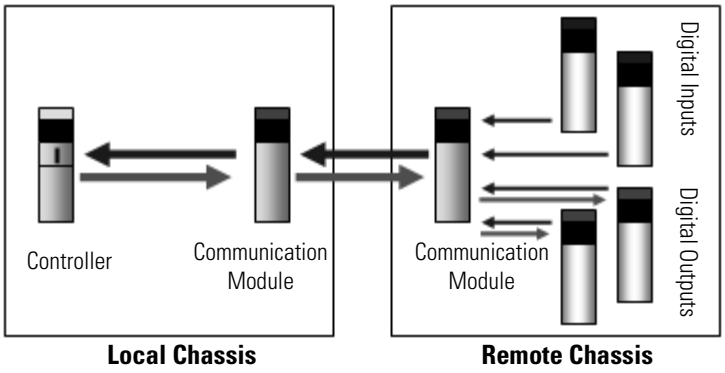
Each module passes its data to/from the controller individually. Communication modules bridge data across networks.



Benefits	Considerations
• Each module can determine its own rate (RPI) • More data can be sent per module, such as diagnostic and analog data • Supports event task communication	• Requires additional connections and network resources • This is the only method supported in the local chassis • I/O data presented as individual tags

Rack-optimized connection

The communication module in a remote chassis consolidates data from multiple modules into a single packet and transmits that packet as a single connection to the controller.



.

Benefits	Considerations
• One connection can service a full chassis of digital modules • Reduces network resources and loading	• All modules are sent at the same rate • Unused slots are still communicated • Still need a direct connection for analog and diagnostic data • Limited to remote chassis • I/O data presented as arrays with alias tags for each module

The rack-optimized format limits data to a single 32-bit input word per module in a chassis. If you place a diagnostic module in a chassis, the rack-optimized format eliminates the value that the diagnostic module offers. In this case, it's better to use a direct connection so that all of the module's diagnostic information is passed to the controller.

Electronic Keying

When you configure a module, you select an electronic keying configuration to determine how you can place a different module in the same chassis slot or position.

Option	Description
Exact Match	Any future module you place in this chassis slot or position must be the exact same physical module as the original module and have the same vendor code, catalog number, major revision, and where appropriate, minor revision.
Compatible Match	Any future module you place in this chassis slot or position must be the same physical module as the original module and have the same vendor code, catalog number, and major revision. In addition, the minor revision must be equal to or greater than the original. Compatible match also lets modules replace other modules that they can emulate. For example, you can replace a 1756-ENET module (configured with compatible match) with a 1756-ENET, 1756-ENBT, or 1756-EN2T module.
Disable Keying	No electronic keying configuration is required. You can place any module in the same chassis slot or position.

WARNING

Changing the RPI and electronic keying selections may cause the connection to the module to be broken and may result in a loss of data.

Be extremely cautious when using the Disable Keying option. If used incorrectly, this option can lead to personal injury or death, property damag, or economic loss.

Guidelines to Manage I/O Connections

Guideline	Description
The type of I/O module can determine the type of connection.	Analog modules always use direct connections, except for 1771 analog modules that use connected messaging. Digital modules can use direct or rack-optimized connections. Communication formats that include optimization in the title are rack-optimized connections; all other connection options are direct connections.
Select the communication format for a remote adapter based on the remote I/O modules.	Select one of these formats for a remote adapter.

Select	If
None	The remote chassis contains only analog modules, diagnostic digital modules, fused output modules, or communication modules. On a ControlNet network, use None to add a new chassis to the network while the controller is running.
Rack-Optimized	The remote chassis only contains standard, digital input, and output modules (no diagnostic modules or fused output modules). A ControlNet network supports rack-optimized for scheduled communication only. At runtime (controller is online), you can add new digital modules to an existing rack-optimized connection, but new rack-optimized connections can only be added when offline. An EtherNet/IP network supports new rack optimized connections both offline and at runtime (online). For more information, see page 72.
Listen Only Rack-Optimized	You want to receive I/O module and chassis slot information from a rack-optimized remote chassis owned by another controller. The runtime capability for listen only rack-optimized connections is the same as for rack-optimized connections.

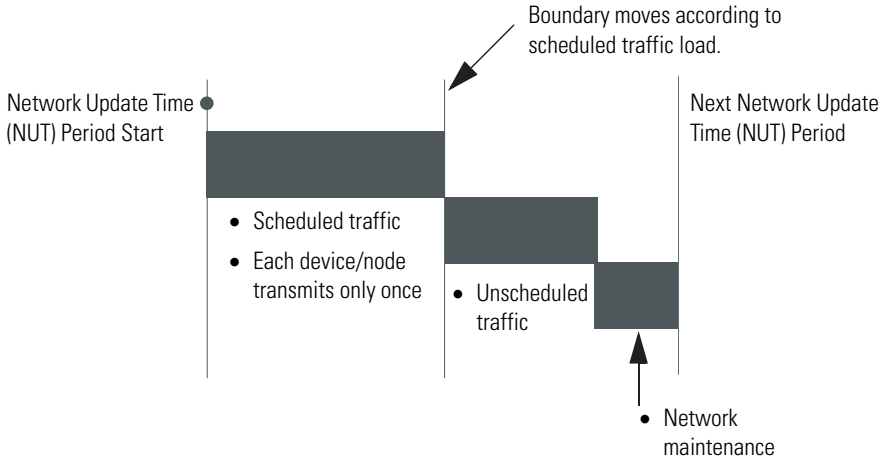
Guideline	Description
Use rack-optimized connections to conserve connection use.	If you are trying to limit the number of controller and network connections, rack-optimized connections can help.
In some cases, all direct connections work best.	For a remote adapter module configured for rack-optimized connections, there is always data sent for each slot in the chassis, even if a slot is empty or contains a direct connection module. There are 12 bytes of data transferred for rack-optimized overhead between the controller and the remote adapter module. In addition, the remote adapter module sends 8 bytes per slot to the controller; the controller sends 4 bytes per slot to the remote adapter. For a small number of digital modules in a large chassis, it might be better to use direct connections because transferring the full chassis information might require more system bandwidth than direct connections to a few modules.

Example	Description
Remote 17-slot chassis Slot 0: 1756-CNBR/D Slots 1...15: analog modules Slot 16: standard digital module	**Option 1:** Select Rack Optimization for remote adapter's communication format. This example uses 16 controller connections (15 for analog modules and 1 for the rack-optimized connection). This example also transfers: • 12 bytes for rack-optimized overhead. • 12 bytes for the digital module. • 12 bytes for each of the 15 analog modules, for a total of 180 bytes. **Option 2:** Select None for the remote adapter's communication format. This example also uses 16 controller connections (1 direct connection to each I/O module). There is no rack-optimized overhead data to transfer. **Recommendation:** Option 2 is recommended because it avoids unnecessary network traffic, and thus improves network performance.
Remote 17-slot chassis Slot 0: 1756-CNBR/D Slots 1...8: analog modules Slots 9...16: digital modules	**Option 1:** Select Rack Optimization for the remote adapter's communication format. This example uses nine controller connections (eight for analog modules and one for the rack-optimized connection). This example also transfers: • 12 bytes for rack-optimized overhead. • 12 bytes for each of the 8 digital modules, for a total of bytes 96 bytes. • 12 bytes for each of the 8 analog modules, for a total of 96 bytes. **Option 2:** Select None for remote adapter's communication format. This example uses 16 controller connections (1 direct connection to each I/O module). There is no rack-optimized overhead data to transfer. **Recommendation:** The best option for this example depends on the type of digital I/O modules in the system and other controller connections. If the total system has many analog modules, diagnostic modules, fused output modules, or produced/consumed tags, select Option 1 to conserve controller connections. If there are plenty of controller connections available, select Option 2 to reduce unnecessary network traffic.

Control 1771 I/O Modules

The Logix5000 controllers support:

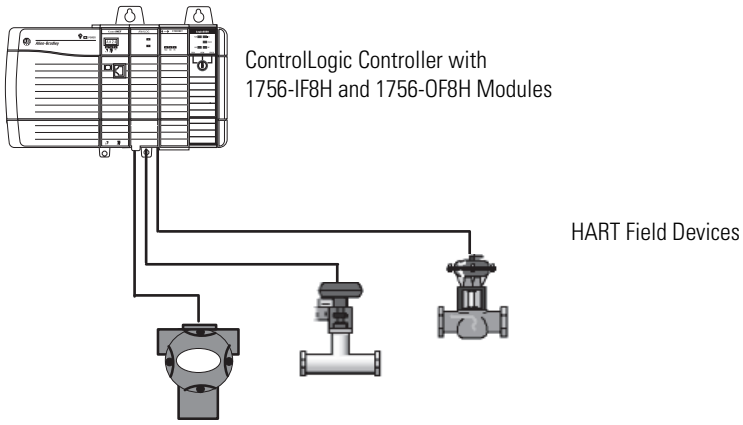
- remote I/O communication to 1771 digital and analog I/O modules.
- ControlNet communication to 1771 digital I/O modules.
- block-transfer message instructions via a ControlNet network to 1771 analog and intelligent I/O modules.

Guideline	Description
Distribute 1771 analog I/O modules.	Spread 1771 I/O analog I/O modules across multiple chassis to reduce the number of block-transfers a single 1771-ACN15, 1771-ACNR15, or 1771-ASB adapter manages. Isolate different 1771 chassis on different networks to diversify the communication so no single network communication module has to manage all the block-transfer messages.
For block-transfers over a ControlNet network, increase the amount of ControlNet unscheduled bandwidth.	The amount of time available for unscheduled communication is determined by the traffic load of scheduled communication.  Increase the controller system overhead to allocate more CPU time to message and block-transfer processing.
Program block-transfers.	Unscheduled data is limited to 510 bytes/node per ControlNet NUT. The 1756-CNB is limited to 128 words per transfer. If needed, data will be sent in multiple packets. The data transfer occurs asynchronous to the program scan. For more information on block-transfers, see page 90.

Communicate with HART Devices

HART (Highway Addressable Remote Transmitter) is an open protocol designed to connect analog devices in industrial process-measurement applications. The protocol uses the standard 4...20 mA current loop widely used for such measurements.

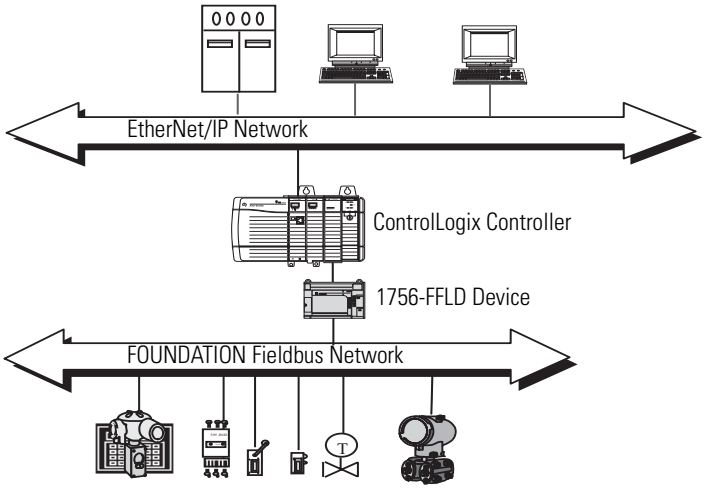
The 1756-IF8H and 1756-OF8H modules offer analog and HART connectivity in one module. You can place modules local to the controller or remote over ControlNet or EtherNet/IP networks. You do not need external hardware to access the HART signal.



Guideline	Description
Enable HART support on only those channels that need the support.	All the channels share the HART modem, so HART response time is better if you enable only the needed HART channels.
The update rate for the HART part of a tag is slower than for the analog part.	The update rate varies, depending on HART network traffic. If all eight channels have HART enabled, update rates are in the range of 10 s. Be sure to consider this response time in your control strategy. Also, check the data quality indications provided with the HART data.
The Device Variable Status (PVStatus, SVStatus, TVStatus, FVStatus) is a relatively new feature in HART systems.	If your HART device does not support Device Variable Status, the 1756-IF8H and 1756-OF8H module synthesizes a status value based on the communication status with the HART field device. The Dynamic Variables do not update as fast as the Analog Signal. The actual rate depends on the number of channels configured for HART, the number of Pass Through commands, the presence of handheld communicators or other secondary masters, and the response speed of the field device.
The 1756 HART modules support asset management software.	HART must be enabled before any asset management access is possible, including scanning for multiplexors. RSLinx Professional software, RSlinx Gateway software, and RSLinx OEM software let asset management software communicate through NetLinx networks and 1756 backplane. Endress+Hauser FieldCare asset management software is a Field Device Tool (FDT) frame application. The frame application runs the Device Type Manager (DTM) files. The DTM files are executable files provided by control and device vendors. There are communication DTMs and device DTMs. Rockwell Automation provides one communication DTM for RSLinx software and the 1756 backplane and two other communication DTMs for the 1756 HART modules. Companies like Endress+Hauser provide device DTMs for their instruments and valves. The device DTMs provide visualization of the parameters needed to configure, monitor, and maintain the devices.

Communicate with FOUNDATION Fieldbus Devices

The 1757-FFLD linking device bridges an EtherNet/IP network to FOUNDATION Fieldbus links.



Guideline	Description
Each 1757-FFLD device supports 16 Logix exchange function blocks.	Each Logix exchange function block supports eight analog inputs and eight analog outputs. Each function block uses one CIP connection. One 1756-ENBT communication module supports 125 CIP connections, which in turn supports seven 1757-FFLD devices that each have 16 function blocks. Add another 1756-ENBT module to support additional 1757-FFLD devices.
The type of device affects the maximum number of FOUNDATION Fieldbus devices per H1 segment.	Each 1757-FFLD device supports two or four H1 segments. • 8...10 instruments (16 maximum) per each H1 segment. • For valves, 4...6 per segment because valves have more data.
Do not exceed the maximum number of virtual communication relationships (VCRs) on each H1 segment.	A VCR is a channel that provides for the transfer of data between FOUNDATION Fieldbus devices. The number of VCRs required to send data or receive data depends on the device and type of data. The 1757-FFLD supports a maximum of 16 publisher and 16 subscriber VCRs for each H1 segment. Each parameter you pass to or from the Logix5000 controller uses a VCR. Some devices, such as valves, use more VCRs than transmitters.
Make sure you have the correct device description (DD) for each FOUNDATION Fieldbus device.	DDs are like EDS files for DeviceNet devices. You can find DDs on vendor/organization websites or on media that ships with the device. A host with DD services can interoperate with all parameters defined in the DD for a field device.
Use the right wiring and connection products.	Always use a tree or modified tree topology. Never daisy chain devices. Noise is the most frequent problem, due to: • wrong wiring. • improper grounding. • bad connectors.
To get the best implementation, understand the details of a FOUNDATION Fieldbus system.	See these references: • FOUNDATION Fieldbus Technical Overview at http://www.fieldbus.org • RelcomWiring Guide at http://www.relcominc.com

Create Tags for I/O Data

Each I/O tag is automatically created when you configure the I/O module through the programming software. Each tag name follows this format.

Location:SlotNumber:Type.MemberName.SubMemberName.Bit

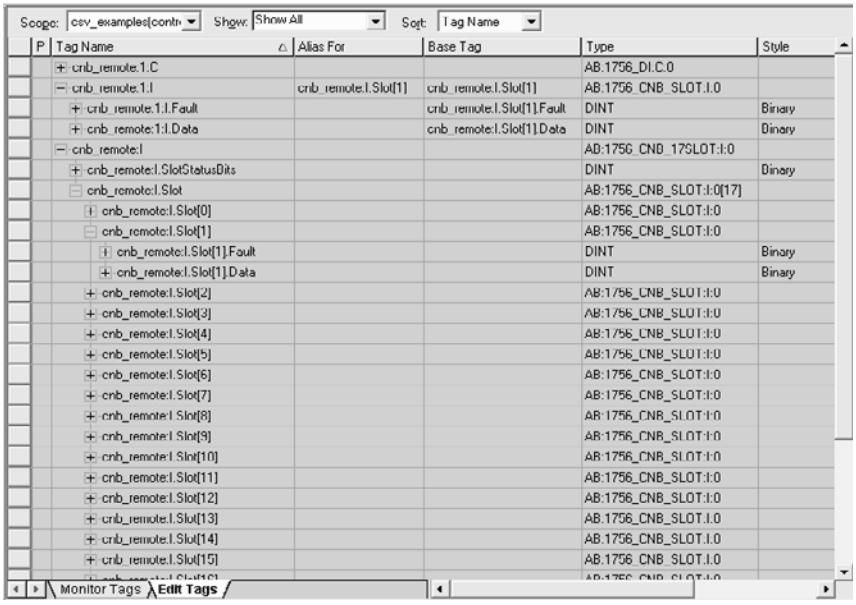
This address variable	Is
Location	Identifies network location LOCAL = local chassis or DIN rail ADAPTER_NAME = identifies remote adapter or bridge
SlotNumber	Slot number of I/O module in its chassis
Type	Type of data: I = input C = configuration O = output S = status
MemberName	Specific data from the I/O module, such as Data and Fault; depends on the module
SubMemberName	Specific data related to a MemberName
Bit (optional)	Specific point on the I/O module; depends on the size of the I/O module (0...31 for a 32-point module)

If you configure a rack-optimized connection, the software creates a rack-object tag for the remote communication module. You can reference the rack-optimized I/O module individually, or by its element within the rack-object tag.

For example, a remote ControlNet communication module (remote_cnb) has an I/O module in slot 1.

This is the individual tag created for the I/O module in remote slot 1.

This is the entry in the rack-object tag for the remote communication module that identifies the I/O module in remote slot 1.



Controller Ownership

When you choose a communication format, you have to choose whether to establish an owner or listen-only relationship with the module.

Mode	Description
Owner	The owner controller writes configuration data and can establish a connection to the module.
Listen-only	A controller using a listen-only connection only monitors the module. It does not write configuration data and can only maintain a connection to the I/O module when the owner controller is actively controlling the I/O module.

There is a noted difference in the ownership of input modules versus the ownership of output modules.

Controlling	This Ownership	Description
Input modules	Owner	An input module is configured by a controller that establishes a connection as an owner. This configuring controller is the first controller to establish an owner connection. Once an input module has been configured (and owned by a controller), other controllers can establish owner connections to that module. This allows additional owners to continue to receive multicast data if the original owner controller breaks its connection to the module. All other additional owners must have the identical configuration data and identical communication format that the original owner controller has, otherwise the connection attempt is rejected.
	Listen-only	Once an input module has been configured (and owned by a controller), other controllers can establish a listen-only connection to that module. These controllers can receive multicast data while another controller owns the module. If all owner controllers break their connections to the input module, all controllers with listen-only connections no longer receive multicast data.
Output modules	Owner	An output module is configured by a controller that establishes a connection as an owner. Only one owner connection is allowed for an output module. If another controller attempts to establish an owner connection, the connection attempt is rejected.
	Listen-only	Once an output module has been configured (and owned by one controller), other controllers can establish listen-only connections to that module. These controllers can receive multicast data while another controller owns the module. If the owner controller breaks its connection to the output module, all controllers with listen-only connections no longer receive multicast data.

Runtime/Online Addition of I/O Modules

With RSLogix 5000 programming software, version 15, you can add 1756 I/O modules to the Controller Organizer when the controller is in Run mode.

- You can only add 1756 I/O modules at runtime.
- You can add the 1756 I/O modules to the local chassis, remotely via the unscheduled portion of a ControlNet network, and remotely via an EtherNet/IP network.

Network	Considerations
ControlNet network	The ControlNet I/O modules you add at runtime can be added to existing rack-optimized connections or as direct connections (you cannot create rack-optimized connections when adding ControlNet I/O modules at runtime). You can use: • 1756-CN2, 1756-CN2R any series modules. • 1756-CNB, 1756-CNBR series D or later communication modules. Disable the change of state (COS) feature on digital input modules because it can cause inputs to be sent faster than the RPI. Dedicate one ControlNet network to I/O communication only. On the dedicated I/O network, make sure there is: • no HMI traffic. • no MSG traffic. • no programming workstations. RPIs faster than 25 ms for unscheduled modules will overload the 1756-CNB, 1756-CNBR communication module. Also: • use a NUT of 10 ms or more. • keep the SMAX and UMAX values as small as possible. You can add I/O modules until you reach: • 75% utilization of the 1756-CNB, 1756-CNBR communication module. Depending on the RPI, utilization increases 1...4% for each I/O module you add. • 48 connections on the 1756-CNB, 1756-CNBR communication module. • < 350,000 bytes as the remaining unscheduled bandwidth on the ControlNet network.
EtherNet/IP network	Adding EtherNet/IP I/O at runtime follows the same guidelines as adding EtherNet/IP I/O when offline. EtherNet/IP I/O communicates immediately based on the module RPI. There is no required scheduling, so EtherNet/IP communication can be thought of as unscheduled. The EtherNet/IP I/O modules you add at runtime can be added to existing rack-optimized connections, added to new rack-optimized connections, or added as direct connections (you can create new rack-optimized connections when adding EtherNet/IP I/O modules at runtime). You can add I/O modules until you reach the limits of the communication module.

1756-EN2T Module	1756-ENBT Module	1756-ENET/B Module
10,000 pps	5000 pps	900 pps
128 TCP connections	64 TCP connections	64 TCP connections
256 CIP connected messages	128 CIP connected messages	160 CIP connected messages

Add I/O Modules at Runtime/Online

1756 Module Type and Connection Method	In Local Chassis		In Remote Chassis via a ControlNet Network				In Remote Chassis via an EtherNet/IP Network	
	Offline	Runtime[1]	Offline		Runtime[1]		Offline	Runtime[1]
			Scheduled	Unscheduled	Scheduled	Unscheduled		
Motion - direct	Yes	No	N/A	N/A	N/A	N/A	N/A	N/A
Digital - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Digital - rack-optimized	N/A	N/A	Yes	No	Yes	No	Yes	Yes
Analog - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Generic third party - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
1756-DNB	Yes	No	Yes	No	No	No	Yes	Yes
1756-DHRIO	Yes	No	Yes	No	No	No	Yes	Yes
1756-CN*x* - no connection	Yes	Yes	Yes	Yes	No	Yes	N/A	N/A
1756-CN*x* - rack-optimized	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A
Generic ControlNet 3rd party - direct	N/A	N/A	Yes	Yes	No	Yes	N/A	N/A
1756-EN2T, 1756-ENBT, 1756-ENET - no connection	Yes	Yes	N/A	N/A	N/A	N/A	Yes	Yes
1756-EN2T, 1756-ENBT, 1756-ENET - rack-optimized	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes
Generic EtherNet/IP third party - direct	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes

(1) Support added with RSLogix 5000 programming software, version 15.

Design Considerations for Runtime/online Addition of I/O

When you design your network, address these considerations if you to add I/O modules at runtime.

Design Issue	Considerations
I/O modules	Currently, you can only add 1756 I/O modules at runtime. Leave space in the local chassis, remote chassis on a ControlNet network, or remote chassis on an EtherNet/IP network for the I/O modules you want to add.
Input transmission rate	Make sure the RPIs work for the data you want to send and receive. Make sure the added I/O does not depend on change-of-state data.
Network topology	On a ControlNet network, install spare taps so you can add 1756 I/O modules at runtime without disrupting the network. Each tap must be terminated so as to not ground out the system. Check ControlNet system requirements to determine how many spare taps your network can support. • In a ControlNet network with redundant cabling, you can break the trunk and add a new tap, but redundant cabling is lost during the module installation. • In a ControlNet ring, add a new drop off the ring or add new nodes off the coax and disrupt only part of the network. • You could remove a single existing node and add a repeater off of that drop. Then re-add the existing node and add any new nodes off of the new segment. On EtherNet/IP, reserve some connection points on the switch so that you can connect additional nodes or switches in the future.
Network configuration	On a ControlNet network, plan which communication can be scheduled or can be unscheduled. On an EtherNet/IP network, all communication is immediate and occurs based on a module's RPI (also referred to as unscheduled). If you know that you will need a new chassis with digital modules in the future, configure the network and add it to the I/O configuration tree as rack optimized. Then inhibit the communication adapter until you need the chassis.
Network performance	You can I/O modules at runtime until you impact the capacity of the communication module. Make sure you have sufficient communication modules for the connections you plan to add.

For more information, see the Runtime/Online Addition of ControlLogix (1756) I/O Over ControlNet and EtherNet/IP White Paper, publication LOGIX-WP006.

Determine the Appropriate Network

Introduction

NetLinx Open Network Architecture is the Rockwell Automation strategy of using open networking technology for seamless, top-floor to shop-floor integration. The networks in the NetLinx architecture — DeviceNet, ControlNet, and EtherNet/IP — share a universal set of communication services. These are the recommended networks for Logix control systems.

Select a Network

Comparison	EtherNet/IP Network	ControlNet Network	DeviceNet Network
Control I/O	Better	Best	Low density
Configuration devices	Best	Best	Best
Collect data	Best	Better	Good
Peer interlocking	Better	Best	Good
Devices	Better	Better	Best
Topologies	Star Requires switches	Trunkline/dropline Star with repeaters	Trunkline/dropline
Number of nodes	Many nodes	99 nodes	63 nodes
Performance	Best	Best	Good
Runtime/online addition of I/O modules	Supported(1)	Supported(1)	Coming soon(2)

(1) For information, see Runtime/Online Addition of I/O Modules on page 72.

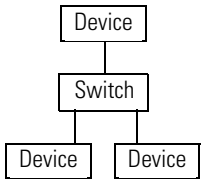
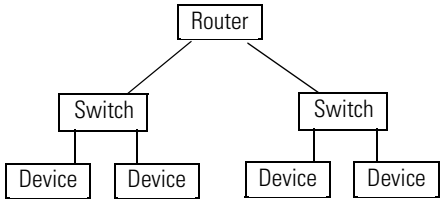
(2) With 1756-DNB firmware, revision 7 and later, and its associated RSNetWorx for DeviceNet software, you can add nodes to a running network. Until this firmware is available, you can only add nodes to a DeviceNet network when the scanner is in Idle mode.

Follow these guidelines when planning a network.

Design Issue	Considerations
Network topology	Plan for future connections. Plan for additional controllers and/or communication modules to handle future I/O modules.
Network configuration	On a ControlNet network, plan which communication can be scheduled or can be unscheduled. On an EtherNet/IP, network all I/O communication is based on a module's RPI (also referred to as unscheduled). If you know that you will need a new chassis with digital modules in the future, configure the network and add it to the I/O configuration tree as rack optimized. Then inhibit the communication adapter until you need the chassis.
Network performance	Make sure you have sufficient communication modules for the connections you plan to use.
Chassis	Consolidate communication connections for multiple modules to a single network node. Group digital I/O modules into a rack-optimized connection to reduce the amount of communication and network bandwidth.
Input transmission rate	Make sure the RPIs work for the data you want to send and receive. Make sure that I/O added at runtime does not depend on change-of-state data.

For more information about planning for adding I/O modules at runtime/online, see page 72.

EtherNet/IP Network Topology

EtherNet/IP Network	Topology
• An EtherNet/IP network supports messaging, produced/consumed tags, and distributed I/O • An EtherNet/IP network supports half/full duplex 10 Mbps or 100 Mbps operation • An EtherNet/IP network requires no network scheduling and no routing tables • There are several methods available to configure EtherNet/IP network parameters for devices. Not all methods are available at all times. These methods are device and configuration dependent: – DHCP – Rockwell Automation BOOTP/DHCP utility – RSLinx software – RSLogix 5000 software – RSNetWorx for EtherNet/IP software **Application Ideas** • Connect many computers • Default gateway to business systems • Star topology best for few nodes and short distances	**Example 1**  **Example 2** 

Guidelines for EtherNet/IP Networks

<table>
<tr><th>Guideline</th><th>Description</th></tr>
<tr><td>Use these publications.</td><td>Use these publications when installing an EtherNet/IP network:
EtherNet/IP Modules in Logix5000 Control Systems User Manual, publication ENET-UM001
EtherNet/IP Web Server Module User Manual, publication ENET-UM527
EtherNet/IP Media Planning and Installation Guide, ENET-IN001</td></tr>
<tr><td>Make sure the switch has the required features.</td><td>For EtherNet/IP control, use an industrial-grade switch.<table><tr><th>Required or Recommended</th><th>Switch Feature</th></tr><tr><td>Required</td><td>Full-duplex capability on all ports</td></tr><tr><td>Recommended</td><td>• VLAN
• Autonegotiation and manually configurable speed/duplex
• Wire-speed switching fabric
• SNMP</td></tr><tr><td>Recommended in systems that require managed switches
(see Guidelines for Switches in EtherNet/IP Systems on page 78)</td><td>• IGMP snooping
– Constrains multicast traffic to ports associated with a specific IP multicast group
– Most switches require a router for IGMP snooping
– In a standalone network, make sure the switch supports IGMP snooping without a router present
• Port diagnostics
• Port mirroring (required for troubleshooting)</td></tr></table></td></tr>
<tr><td>Consider using switches from Encompass partners.</td><td>These Encompass partners have switches that meet the required features: Cisco, Hirschmann, and N-Tron.</td></tr>
<tr><td>Data transmission depends on the controller.</td><td>The type of Logix5000 controller determines the data transmission rate.
ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module.
CompactLogix, FlexLogix, and DriveLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, or 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms.</td></tr>
<tr><td>You can add I/O modules at runtime.</td><td>With Logix controller firmware, revision 15, you can add 1756 I/O modules to remote chassis connected via an EtherNet/IP network to a running controller. You can configure direct or rack-optimized connections. For more information see page 72.</td></tr>
<tr><td>Data transmission rate depends on the RPI.</td><td>An EtherNet/IP network broadcasts I/O information to the controller based on the RPI setting. With change of state (COS) enabled and:
no data changes, the EtherNet/IP module produces data every RPI.
data changes, the EtherNet/IP module produces data at a maximum rate of RPI/4.</td></tr>
<tr><td>Use unicast EtherNet/IP communication to reduce broadcast network traffic.</td><td>To reduce bandwidth use and preserve network integrity, some facilities block multicast Ethernet packets. With RSLogix 5000 software, version 16, you can configure a produced/consumed tag to use multicast or unicast connections. Unicast connections:
reduce network bandwidth.
simplify Ethernet switch configuration.</td></tr>
</table>

Guidelines for Switches in EtherNet/IP Systems

Use a Managed Switch	Use an Unmanaged Switch
• The EtherNet/IP control system is directly connected to the business system via a switch or router. Proper segregation of the control and business network is always a good design practice. • The system has non-Rockwell Automation EtherNet/IP devices connected on the network (except for personal computers). These devices might not be able to properly handle the multicast traffic generated by the I/O devices. • If the system needs to perform troubleshooting. For troubleshooting, you need port mirroring, which is only supported with a managed switch.	**In I/O Systems** • In an isolated EtherNet/IP architecture. The control system is not directly connected to the business system. Or the control system is connected to the business system via a ControlLogix gateway (for example, a ControlLogix chassis contains two 1756-ENBT modules; one is connected to the control system and the other is connected to the business system). • The EtherNet/IP control system contains only Rockwell Automation devices (with the exception of personal computers). • Traffic loading through each device (in packets/sec) is less than the capacity of each device. **In Non-I/O Systems** • The EtherNet/IP traffic on the network consists of messaging only (MSG instructions, HMI, program upload/download). In this case, an unmanaged switch is acceptable regardless of network architecture.

If you use an unmanaged switch, you give up these features:

- Switch port diagnostics
- Port mirroring
- Forced duplex speed
- SNMP
- IGMP snooping
- Web browser for viewing configuration and diagnostics

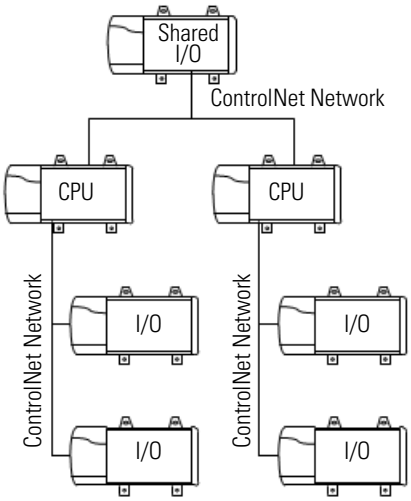
Determine Whether Your System Operates Properly

Rockwell Automation EtherNet/IP devices (such as the 1756-ENBT scanner) have embedded diagnostic web pages.

On this web page	Look for
Ethernet Statistics page • All media counters • In Error and Out Error counters • Rejected Packets counter	These numbers should be near zero (0) and not incrementing.
Diagnostic Overview page	• There should be no connection timeouts. • The packets/sec counts should be within each device's capacity. • The MISSED counter under I/O Packet Counter Statistics should be zero (0).

If connections frequently break or if HMIs appear to update slowly, reduce traffic loading. If the situation is multicast-related, it might also help to use managed switches with IGMP snooping.

ControlNet Network Topology

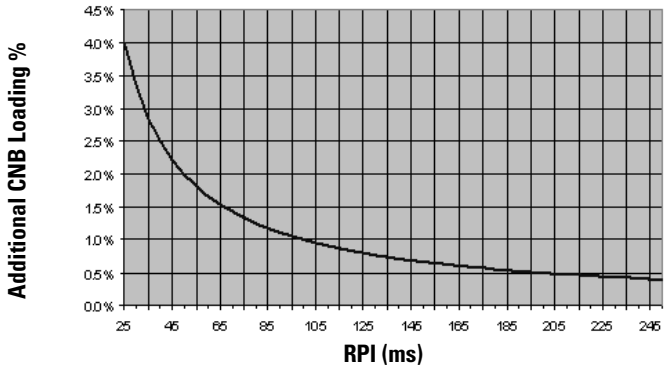
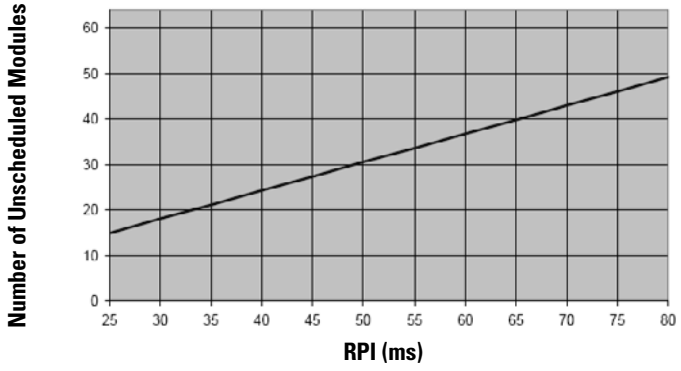
ControlNet Network	Topology
• A ControlNet network allows both I/O and messaging on the same wire. • Multiple controllers and their respective I/O can also be placed on the same ControlNet wire. • When new I/O is added or an existing I/O module's communication structure is changed, you must use RSNetWorx for ControlNet software to reschedule the network. • If the network timing changes, every device with scheduled traffic on the network is affected. • To reduce the impact of changes, place each CPU and its respective I/O on isolated ControlNet networks. • Place shared I/O and produced/consumed tags on a common network available to each CPU that needs the information. **Application Ideas** • Default Logix network. • Best replacement for Universal Remote I/O. • Backbone to multiple distributed DeviceNet networks. • Peer interlocking network. • Common devices include: Logix5000 controllers, PanelView terminals, I/O modules, and drives.	

Guidelines for ControlNet Networks

Guideline	Description
Use these publications.	Use these publications when installing a ControlNet network: • ControlNet Coax Media Planning and Installation Guide, publication CNET-IN002 • ControlNet Fiber Media Planning and Installation Guide, publication CNET-IN001 • ControlNet Modules in Logix5000 Control Systems User Manual, publication CNET-UM001
Limit the number of nodes per ControlNet network to 40.	The ControlNet network was designed with a limit of 99 nodes per network, but this number of nodes decreases network performance. A maximum of 40 nodes per network results in better performance and leaves bandwidth for other communication.
Adjust the default RSNetWorx for ControlNet settings.	Change these settings in the RSNetWorx for ControlNet software: • UMAX (highest unscheduled node on the network) – Default is 99 – The network takes the time to process the total number of nodes specified in this setting, even if there are not that many devices on the network – Change to a reasonable level to accommodate the active devices on the network and any additional devices that might be connected • SMAX (highest scheduled node on the network) – Default is 1 – This must be changed for all systems – Set SMAX < UMAX

Guideline	Description
Design for at least 400 KB of available, unscheduled network bandwidth.	Leaving too little memory for unscheduled network bandwidth results in poor message throughput and slower workstation response. Unscheduled data transfers on ControlNet occur asynchronous to the program scan and support a maximum of 510 bytes/node per ControlNet NUT. The 1756-CNB, 1756-CNBR communication modules support a maximum of 128 words per data transfer. If necessary, unscheduled data is sent in multiple packets.
Place DeviceNet (1756-DNB) and serial (1756-MVI) communication modules in the local chassis.	DeviceNet (1756-DNB) and serial (1756-MVI) communication modules have multiple, 500-byte data packets that will impact scheduled bandwidth. Placing these modules in the same chassis as the controller avoids this data being scheduled over the ControlNet network. If you must place these communication devices in remote chassis, configure the input and output sizes to match the data configured in RSNetWorx for DeviceNet software. This reduces the amount of data that must be transmitted.
Limit 1756-CNB, 1756-CNBR connections.	For best performance, limit the 1756-CNB, 1756-CNBR to 40...48 connections. Add additional modules in the same chassis if you need more connections. Adding more modules and splitting connections among the modules can improve system performance. If the chassis that contains the CNB module also contains multiple digital I/O modules, configure the CNB module's communication format for Rack Optimization. Otherwise, use None. Use 1756-CNB, 1756-CNBR modules in chassis that contain only I/O modules. Use the 1756-CN2, 1756-CN2R communication modules in the same chassis as the controller.
For additional connections, consider the 1756-CN2, 1756-CN2R modules.	The 1756-CN2, 1756-CN2R communication modules each support 100 connections. In addition to the 100 connections, the modules also have higher performance. Use 1756-CN2, 1756-CN2R communication modules in the same chassis as the controller. Use 1756-CNB, 1756-CNBR modules in chassis that contain only I/O modules.
If you change network settings, resave each controller's project.	Any time you use RSNetWorx for ControlNet software and you save or merge your edits, attach to each controller in the system with their respective RSLogix 5000 project file and perform a save. This copies the ControlNet settings into the offline, database file and makes sure that future downloads of the controller permit it to go online without having to run RSNetWorx for ControlNet software.
You can add I/O modules at runtime.	With Logix controller firmware, revision 15, you can add 1756 I/O modules to remote chassis connected via EtherNet/IP to a running controller. You can configure direct or rack-optimized connections.
Data transmission depends on the controller.	The type of Logix5000 controller determines the data transmission rate. • ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module. • CompactLogix, FlexLogix, and DriveLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, and 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms.

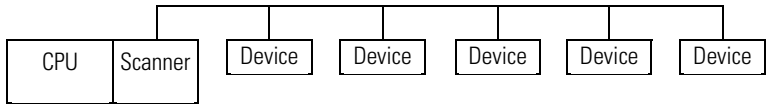
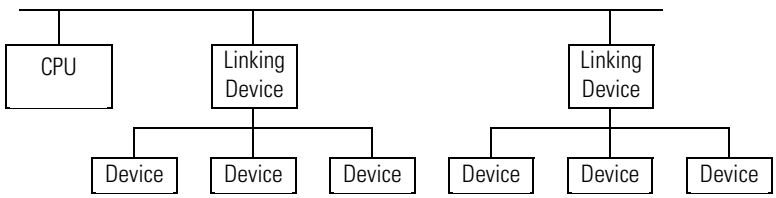
Guidelines for Unscheduled ControlNet Networks

Guideline	Description
You can run an entire ControlNet network as unscheduled.	An unscheduled ControlNet network: • provides for easier network configuration. • is useful if your I/O updates needs are slower. You still need to run RSNetWorx for ControlNet software to configure NUT, SMAX, UMAX, and media configuration settings.
Plan appropriately if you place I/O on an unscheduled ControlNet network.	Follows these recommendations for I/O on an unscheduled ControlNet network: • You must use a 1756-CNB, 1756-CNBR series D or later. • Disable the Change of State (COS) feature on digital input modules because it can cause inputs to be sent faster than the RPI. • Dedicate a ControlNet network to I/O only. • Do not exceed 75% utilization of the 1756-CNB, 1756-CNBR communication module. • Have no more than 48 connections on the 1756-CNB, 1756-CNBR communication module. • Use a NUT of 10 ms or more. • Keep the SMAX and UMAX values as small as possible.
Set the RPI at 25 ms or slower.	Use RPIs of 25 ms or slower for unscheduled modules to avoid overload on the 1756-CNB, 1756-CNBR communication module. Depending on the RPI, the communication module loading increases 1...4% for each I/O module added. **Additional 1756-CNB, 1756-CNBR Loading** 
The RPI affects how many I/O modules you can have.	This chart shows the number of modules and associated RPIs so that you do not exceed 75% utilization of the 1756-CNB, 1756-CNBR communication module. **Maximum Number of I/O Modules in an Unscheduled Network** 

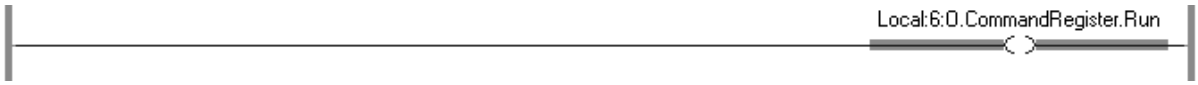
Compare Scheduled and Unscheduled ControlNet Communication

Scheduled ControlNet Communication	Unscheduled ControlNet Communication
Deterministic	Less deterministic than scheduled communication Allows for simpler ControlNet installations when scheduled networks are not required
To add scheduled I/O on ControlNet you must: • add the I/O to an offline RSLogix5000 project. • download the RSLogix5000 project to the controller. • run RSNetWorx to schedule the network requires network to be scheduled (must stop the network and put the controller in Program mode to schedule a network). • save the RSLogix 5000 project.	Can be changed online without impacting schedule New modules can impact other modules communicating via unscheduled bandwidth
RPI and NUT determine module communication rates	RPI determines module communication rates
MSG and HMI traffic can occur on the same network because they are isolated in unscheduled traffic MSG and HMI traffic can affect I/O communication	Should dedicate a ControlNet network for I/O only MSG and HMI traffic can affect I/O communication
Direct and rack-optimized connections to I/O	Only direct connections to I/O (results in being able to use fewer total I/O modules because of the connection limits controllers and communication modules)
Supports any firmware revision of a ControlNet communication module	You can use any 1756-CN2, 1756-CN2R communication module If you use a 1756-CNB, 1756-CNBR communication module, it must be series D or later
Supports any I/O platform that can communicate via a ControlNet network	With RSLogix software, version 15, supports only 1756 I/O modules

DeviceNet Network Topology

DeviceNet Network	Topology
• You need a DeviceNet scanner to connect the controller to DeviceNet devices. • You must use RSNetWorx for DeviceNet software to configure devices and create the scanlist for the scanner. • You can configure the network baud rate as 125 Kbps (default and a good starting point), 250 Kbps, or 500 Kbps. • If each device on the network (except the scanner) sends ≤ 4 bytes of input data and receives ≤ 4 bytes of output data, you can use the AutoScan feature on the scanner to configure the network. **Application Ideas** • Distributed devices • Drives network • Diagnostic information	**Single Network**  **Several Smaller Distributed Networks (Subnets)** 

Guidelines for DeviceNet Networks

Guideline	Description
Use these publications.	Use these publications when installing a DeviceNet network: • DeviceNet Cable System Manual, publication DN-UM072 • DeviceNet Modules in Logix5000 Control Systems User Manual, publication DNET-UM001
Use the DeviceNet Tag Generator tool.	RSLogix 5000 software, version 16, includes a DeviceNet tag generator tool that creates device-specific structured tags and logic based on the network configuration in RSNetWorx for DeviceNet software. The logic copies data to/from the DNB data array tags to the device tags so that data is presented synchronously to program scan.
Place DeviceNet (DNB) communication modules in the local chassis.	Placing DNB modules in the local chassis maximizes performance, especially in ControlLogix systems. Size the input and output image for the DNB modules to the actual devices that are connected plus 20% for future growth. If you have to place DNB modules in remote chassis, sizing the input and output images is critical for best performance.
Verify the total network data does not exceed the maximum DNB data table size.	A DNB supports: • 124, 32-bit input words. • 123, 32-bit output words. • 32, 32-bit status words. You can use RSNetWorx for DeviceNet software offline to estimate network data. Use a second DNB if there is more network data than one module can support.
Set up slaves first.	Configure a device's parameters before adding that device to the scanlist. You cannot change the configuration of many devices once they are already in the scanlist. If you configure the scanner first, there is a chance that the scanner configuration will not match the current configuration for a device. If the configuration does not match, the device will not show up when you browse the network.
Leave node address 63 open to add nodes.	Devices default to node 63 out-of-the-box. Leave node address 63 unused so you can add a new devices to the network. Then change the address of the new device.
Leave node address 62 open to connect a computer.	Always leave at least one open node number to let a computer be attached to the network if needed for troubleshooting or configuration.
Don't forget to set the scanner run bit.	For the scanner to be in Run mode, the controller must be in Run mode and the logic in the controller must set the scanner's run bit.
	
Make sure you have the most current EDS files for your devices.	RSNetWorx for DeviceNet software uses EDS file to recognize devices. If the software is not properly recognizing a device, you are missing the correct EDS files) For some devices, you can create an EDS file by uploading information from the device. Or you can get EDS files from: http://www.ab.com/networks/eds.

Notes:

Communicate with Other Devices

Introduction

The MSG instruction asynchronously reads or writes a block of data to another device.

If the target device is a	Select one of these message types
Logix5000 controller	CIP Data Table Read
	CIP Data Table Write
I/O module that you configure using RSLogix 5000 software	Module Reconfigure
	CIP Generic
SERCOS drive	SERCOS IDN Read
	SERCOS IDN Write
PLC-5 controller	PLC5 Typed Read
	PLC5 Typed Write
	PLC5 Word Range Read
	PLC5 Word Range Write
SLC controller MicroLogix controller	SLC Typed Read
	SLC Typed Write
Block-transfer module	Block-Transfer Read
	Block-Transfer Write
PLC-3 processor	PLC3 typed read
	PLC3 typed write
	PLC3 word range read
	PLC3 word range write
PLC-2 processor	PLC2 unprotected read
	PLC2 unprotected write

Cache Messages

Some types of messages use a connection to send or receive data. Some also give you the option of either leaving the connection open (cache) or closing the connection when the message is done transmitting. This table shows which messages use a connection and whether or not you can cache the connection.

This type of message	Using this communication method	Uses a connection	Which you can cache
CIP data table read or write	CIP	X	X
PLC2, PLC3, PLC5, or SLC (all types)	CIP		
	CIP with Source ID		
	DH+	X	X
CIP generic	N/A	Your option(1)	Your option(1)
Block-transfer read or write	N/A	X	X

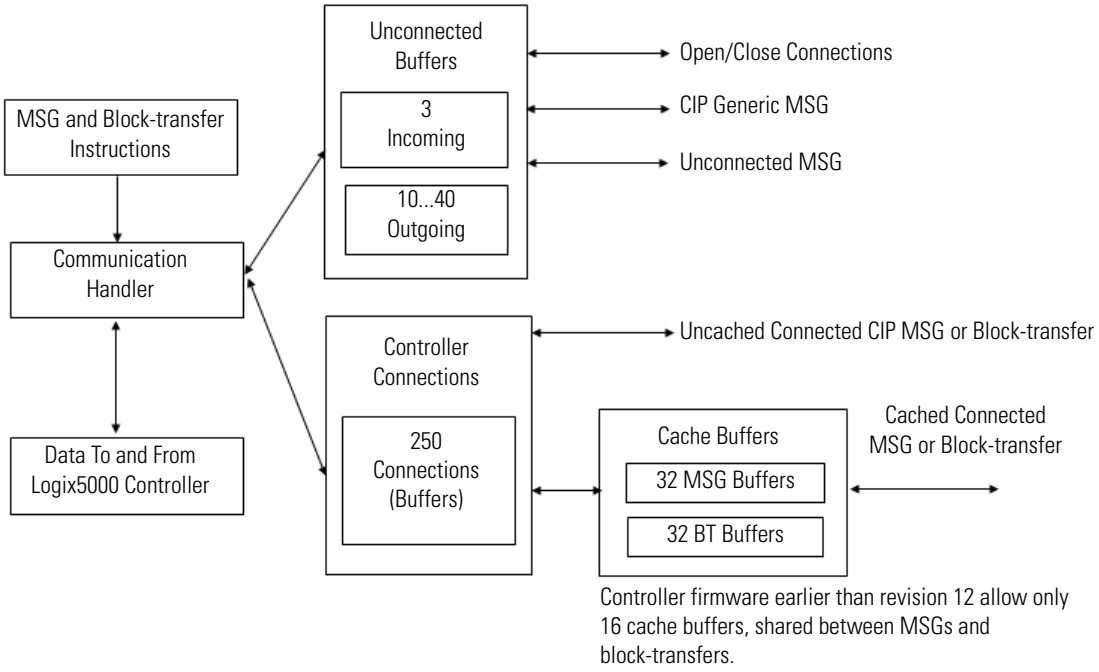
(1) You can connect CIP generic messages, but for most applications we recommend you leave CIP generic messages unconnected.

A cached connection remains open until one of the following occurs:

- The controller goes to Program mode.
- You rerun the message as uncached.
- Another message is initiated and a cached buffer is needed.
- An intermediate node in the connection goes down.

Message Buffers

A Logix5000 controller has buffers for unconnected messages and for cached messages. Buffers store incoming and outgoing message data until the controller can process the data.



Buffer	Description
10 outgoing unconnected buffers You can increase this to 40 by using a CIP Generic message instruction. Each buffer you add uses approximately 1.2 KB of I/O memory. See the MSG section in the Logix 5000 Controllers General Instructions Reference Manual, publication 1756-RM003	The outgoing unconnected buffers are for: • establishing I/O connections to local I/O modules and remote devices on ControlNet, EtherNet/IP, and remote I/O networks. • executing unconnected PLC2, PLC3, PLC5, or SLC (all types) messages over Ethernet or ControlNet (CIP and CIP with Source ID) networks. • initiation of messaging over a DH+ network (uses 2 buffers, one to open the connection and one to transfer data). • initiation of uncached block transfers. • initiation of uncached CIP read/write message instructions. • initiation of cached block transfers. • initiation of cached CIP read/write messages instructions. • CIP Generic message instructions.
3 incoming unconnected buffers	The incoming unconnected buffers are for: • initial receiving of a cached CIP message instruction. • receiving an uncached CIP message instruction. • receiving a message over a DH+ network. • receiving a CIP Generic message instruction. • receiving a read or write request from a ControlNet PanelView terminal (unconnected messaging). • initial receiving of a read request from an EtherNet/IP PanelView terminal (connected messaging). • receiving a write request from an EtherNet/IP PanelView terminal (unconnected messaging). • receiving a initial request from RSLogix 5000 software to go online. • initial receiving of RSLinx connections.

Buffer	Description
Cached buffers **Revision 12 and later firmware:** 32 cached buffers for any combination of messages and block-transfers **Revision 11 and earlier firmware** 16 cached buffers for messages only and 16 cached buffers for block-transfers only	The cached buffers are outgoing buffers for messages and block-transfers. A cached connection helps message performance because the connection is left open and does not need to be reestablished next time it is executed. A cached connection counts towards the total limit of connections for a controller. A cached connection is refreshed at the connection RPI. All cached entries are closed when the controller transitions to Program mode. With revision 12 and later firmware, you can cache 32 messages and block-transfers (any combination). Previous revisions of controller firmware let you cache 16 messages only and 16 block-transfers only. The first time a cached message is executed, it uses one of the 10 out going unconnected buffers. When the connection is established it will then move into the cached buffer area. For optimum performance, do not cache more messages or block-transfers than there are cached buffers. If you cache more than the available cached buffers, the controller looks for a connection that has been inactive for the longest time, closes that connection, and allows a new connection take its place. The controller will close a cached message or block-transfer, depending on which has been inactive the longest. If all 32 cached connections are in use, the message uses one of the unconnected out going buffers. If all the unconnected buffers are in use, the message instruction generate error code 301 (No Buffer Memory) or 302 (Bandwidth Not Available). You can multiplex cached connections. If a connection is inactive and a message instruction executes that has the same target and path, it uses that inactive connection For example, if you have a block-transfer read and write to the same module, interlock the read and write so that only one is active at a time. Then when they are cached, they will use the same cached connection.

Outgoing Unconnected Buffers

Buffers	Use
1...10	The first 10 buffers (default) are shared for unconnected messaging, initiating connected messaging, establishing I/O connections, and establishing produced/consumed connections.
11	The 11th buffer is dedicated to establishing I/O and produced/consumed connections.
12...40	The 12th to the 40th buffers are used only for initiating connected messages and executing unconnected messages. To increase the outgoing buffers to a value higher than 11, execute a CIP generic message to configure that change each time you transition from Program mode to Run mode.

Guidelines for Messages

Guideline	Description
Message tags must exist as controller-scoped, base tags.	The information in a message tag is accessed by the operating system asynchronously to the program scan. In addition to the visible fields within the message tag, there are hidden attributes only referenced by the background operating system.
You can have more than 32 messages in a program.	The controller supports 32 active, cached messages at a time. If you determine that there are more than 32 messages, you cannot cache all of the messages. You need extra programming to ensure that no more than 32 messages are active at the same time. Prior to controller revision 12, the controller supported 16 active, cached messages at a time.
You can use a message to send a large amount of data.	Even though there are network packet limitations (such as 500 bytes on ControlNet and 244 bytes on DH+), the controller can send a large amount of data from a single MSG instruction. When configuring the message, select an array as the source/destination tags and select the number of elements (as many as 32,767 elements) you want send. The controller automatically breaks the array into small fragments and sends all the fragments to the destination. On the receiving side, the data appears in fragments, so some application code may be required to detect the arrival of the last piece.

Guidelines to Manage Message Connections

Guideline	Description
Create user-defined structures or arrays.	User-defined structures let you organize your data to match your machine or process. • One tag contains all the data related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type. • Each individual piece of data (member) gets a descriptive name. This automatically creates an initial level of documentation for your logic. • You can use the structure to create multiple tags with the same data layout. • RSLinx can optimize user-defined structures more than standalone tags.
Cache connections when appropriate.	If a message executes repeatedly, cache the connection. This keeps the connection open and optimizes execution time. Opening a connection each time the message executes increases execution time. If a message executes infrequently, do not cache the connection. This closes the connection upon completion of the message, which frees up that connection for other uses.

The system overhead timeslice percentage you configure for the controller determines the percentage of controller time (excluding the time for periodic and event tasks) that is devoted to communication and background functions. This includes sending and receiving messages.

For information on specifying a system overhead percentage, see Select a System Overhead Percentage on page 27.

Guidelines for Block-transfer Messages

Guideline	Description
Distribute 1771 analog modules across multiple chassis.	Distributing 1771 analog modules across multiple chassis reduces the number of block-transfers that a single 1771-ACN or 1771-ASB module needs to manage.
Isolate different 1771 chassis on different networks.	Isolating different chassis onto different networks diversifies the communication so that no single network or communication module has to deal with all of the communication.
Increase ControlNet unscheduled bandwidth.	If communicating over a ControlNet network, increase the amount of ControlNet unscheduled bandwidth to permit additional time on the network for data exchange. See page 82 for more information about unscheduled bandwidth on a ControlNet network.
Increase the system overhead timeslice percentage.	Increase the Logix5000 controller's system overhead timeslice to allocate more CPU time to communication processing from the continuous task.
Interlock block-transfer read and write messages to the same module.	Programmatically interlock block-transfer read and write messages to the same module so that both operations cannot be active at the same time.
Use the 1757-ABRIO module for systems with a high number of block-transfer modules.	The 1757-ABRIO module provides connectivity from a ControlLogix chassis to 1771 I/O and other modules connected via remote I/O. The 1757-ABRIO module off-loads the burden of performing block-transfers from the controller and increases the number of block-transfer operations that can be performed.

Map Tags

A Logix5000 controller stores tag names on the controller so that other devices can read or write data without having to know physical memory locations. Many products only understand PLC/SLC data tables, so the Logix5000 controller offers a PLC/SLC mapping function that lets you map Logix tag names to memory locations.

- You only have to map the file numbers that are used in messages; the other file numbers do not need to be mapped.
- The mapping table is loaded into the controller and is used whenever a logical address accesses data.
- You can only access controller-scoped tags (global data).

Follow these guidelines when mapping tags.

- Do not use file numbers 0, 1, and 2. These files are reserved for Output, Input, and Status files in a PLC-5 processor.
- Use PLC-5 mapping only for tag arrays of data type INT, DINT, or REAL. Attempting to map elements of system structures may produce undesirable effects.
- Use these file types and identifiers.

For this Logix5000 array type	Use this PLC file identifier
INT array	N or B
DINT array	L
REAL array	F

FactoryTalk Alarms and Events System

Introduction

The FactoryTalk Alarms and Events system integrates alarming between FactoryTalk View SE applications and Logix5000 controllers by embedding an alarming engine in Logix5000 controllers. You need:

- RSLogix 5000 software, version 16 or later.
- FactoryTalk View SE, version 5.0 or later.
- Logix5000 controllers:

Firmware	Revision
ControlLogix non-redundant controllers	16.20 or later
ControLogix redundanct controllers	16.60 or later
CompactLogix controllers	16.20 or later
DriveLogix systems	16.20 or later
SoftLogix controllers	16.40 or later

Two Logix-based alarm instructions are available in relay ladder, structured text, and function block diagram.

- The Digital Alarm (ALMD) instruction detects alarms based on Boolean (true/false) conditions.
- The Analog Alarm (ALMA) instruction detects alarms based on the level or rate of change of analog value.

Guidelines for Logix-based Alarm Instructions

<table>
<tr><th>Guideline</th><th>Description</th></tr>
<tr><td>Estimate increased controller memory use for each alarm.</td><td>The alarm instructions use new alarm data types that contain state information and timestamps for each alarm. Estimate this memory use in the controller:
100 KB per FactoryTalk Alarms and Events subscriber that receives alarms from the controller
There is a maximum of 16 subscribers per controller. Most applications only require one subscriber to a controller to provide data to many FactoryTalk View SE clients.
1 KB per alarm</td></tr>
<tr><td>Alarm instructions increase total controller scan time.</td><td>These execution times show how ALMD instructions and ALMA instructions affect total scan time.<table><tr><th>Rung State</th><th colspan="2">Execution Times</th></tr><tr><th></th><th>ALMD Instruction</th><th>ALMA Instruction</th></tr><tr><td>False</td><td>N/A (ALMD uses rung state as its input condition)</td><td>17 μs</td></tr><tr><td>Scan true with no alarm state change</td><td>7 μs</td><td>65 μs</td></tr><tr><td>Scan true with alarm state change</td><td>25 μs</td><td>118 μs</td></tr></table>An alarm state change is any event that changes the condition of the alarm, such as acknowledging or suppressing the alarm. Minimize the potential for a large number of alarms changing state simultaneously (alarm bursts) by creating dependencies on related alarms. Large alarm bursts can have a significant impact on application code scan time.</td></tr>
<tr><td>You can edit or add an alarm instruction online.</td><td>Online edits of new and existing alarms are automatically sent to the subscribers. You do not have to re-subscribe to receive the updated information. Online changes automatically propagate from the controller alarm structure to the rest of the architecture.</td></tr>
<tr><td>In relay ladder, how you define the alarm values on the instruction faceplate determine whether you can access those values programmatically through the alarm structure.</td><td>When you create an alarm instruction, you also create an alarm data type for that alarm. For example, MyDigitalAlarm of data type DigitalAlarm. In relay ladder, these values are also available on the instruction faceplate:
ProgAck
ProgReset
ProgDisable
ProgEnable
In relay ladder, if you enter a value or assign a tag to these faceplate parameters (such as AckSection1All), the value or tag value is automatically written to the alarm structure each time the instruction is scanned.
In relay ladder, if you want to programmatically access the alarm structure, you must assign the structure tag to the faceplate. For example, to use MyAnalogAlarm.ProgAck in logic, assign the tag MyAnalogAlarm.ProgAck on the faceplate to the ProgAck parameter.</td></tr>
<tr><td>Test alarm behavior from within RSLogix 5000 software.</td><td>On the Status tab of the alarm dialog, monitor the alarm condition, acknowledge an alarm, disable an alarm, suppress an alarm, or reset an alarm. Use the dialog selections to see how an alarm behaves, without needing an operational HMI.</td></tr>
</table>

Configure Logix-based Alarm Instructions

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Message string</td><td>The message string (maximum of 255 characters, including embedded text) contains the information to display to the operator regarding the alarm. In addition to entering text, you can also embed variable information. In the alarm message editor, select the variable you want and add it anywhere in the message string.
<table>
<tr><th>Variable</th><th>Description</th></tr>
<tr><td>Alarm name</td><td>Tag name of the alarm.
<code>/*S:0 %AlarmName*/</code></td></tr>
<tr><td>Condition name</td><td>State of the alarm (such as, true, false, high-high, or low).
<code>/*S:0 %ConditionName*/</code></td></tr>
<tr><td>Input value</td><td>True, false, or surrent value of the analog input value.
<code>/*N:5 %InputValue NOFILL DP:0*/</code></td></tr>
<tr><td>Limit value</td><td>Limit or condition that caused the alarm.
<code>/*N:5 %LimitValue NOFILL DP:0*/</code></td></tr>
<tr><td>Severity</td><td>The assigned importance of the alarm.
<code>/*N:5 %Severity NOFILL DP:0*/</code></td></tr>
<tr><td>Values of associated tags</td><td>Values of the selected tags delivered with the alarm.
<code>/*N:5 %Tag1 NOFILL DP:0*/</code></td></tr>
</table>
This information is always sent with the alarm, viewable by the operator, and entered in the history log, regardless of whether you embed it in the message string.
You cannot programmatically access the alarm message string from the alarm tag. To change the alarm message based on specific events, configure one of the associated tags as a string data type and embed that associated tag in the message.
You can have multiple language versions of messages. You enter the different language via the import/export utility. For more information, see page 94.</td></tr>
<tr><td>Associated tags</td><td>You can select as many as four additional tags from the controller project to associate with the alarm. These tags are sent with an alarm message to the alarm server. Associated tags can be BOOL, INT, SINT, DINT, REAL, or string data types. For example, a digital alarm for a pressure relief valve might also include information such as pump speed and system pressure, and tank temperature
Optionally, embed the associated tags into the message text string.</td></tr>
<tr><td>Severity</td><td>Use the configurable severity range from 1...1000 to rank the importance of an alarm. A severity of 1 is for low priority alarms; a severity of 1000 is for an emergency condition.
By default, in the FactoryTalk alarm service, severities:
1...250 are low alarms.
251...500 are medium alarms.
501...750 are high alarms.
751...1000 are urgent alarms.
You can configure how the FactoryTalk ranges are presented to the operator. The operator can also filter on alarm levels. For example, a maintenance engineer can filter to see only those alarms at severity 128.</td></tr>
</table>

Option	Description
Alarm class	Use the alarm class to group related alarms. Specify the alarm class exactly the same for each alarm you want in the same class. The alarm class is case sensitive. For example, specify class Tank Farm A to group all the tank alarms for a specific area. Or specify class Control Loop to group all alarms for PID loops. You can then display and filter alarms at the HMI based on the class. For example, an operator can display all tank alarms or all PID loop alarms. The alarm class does not replace subscription to specific alarms. The FactoryTalk View SE Alarm object graphics have configuration options to determine which controller alarms an operator sees.
View command	Execute a command on the operator station when requested by an operator for a specific alarm. This lets an operator execute any standard FactoryTalk View command, such as call specific faceplates and displays, execute macros, access help files, and launch external applications. When the alarm condition occurs and is displayed to the operator, a button on the summary and banner displays lets the operator run an associated view command.
Defaults	The Parameters tab of the alarm instruction properties lets you define values for instruction parameters. You can return the parameters to factory defaults and you can define your own set of instruction defaults. The instruction defaults you assign are defaults for only that instance of the instruction.

Multiple Language Versions of Alarm Messages

You can maintain alarm messages in multiple languages. Either enter the different languages in the associated language versions of RSLogix 5000 programming software or in an import/export (.CSV or .TXT) file.

You can access alarm message text from an import/export (.CSV or .TXT) file and add additional lines for translated versions of the original message string. Messages in different languages use ISO language codes in the TYPE column. Text for the operator is in the DESCRIPTION column. The SPECIFIER identifies the type of alarm.

TYPE	NAME	DESCRIPTION	SPECIFIER
ALMMSG:en-us	ALMA Logix Tag Name	HH alarm text for operator in English	HH
ALMMSG:en-us	ALMA Logix Tag Name	H alarm text for operator in English	H
ALMMSG:en-us	ALMA Logix Tag Name	L alarm text for operator in English	L
ALMMSG:en-us	ALMA Logix Tag Name	LL alarm text for operator in English	LL
ALMMSG:en-us	ALMA Logix Tag Name	ROC positive alarm text for operator	POS
ALMMSG:en-us	ALMA Logix Tag Name	ROC negative alarm text for operator	NEG
ALMMSG:de-ch	ALMA Logix Tag Name	HH Mitteilung für den Operator auf Deutsch	HH
ALMMSG:de-ch	ALMA Logix Tag Name	H Mitteilung für den Operator auf Deutsch	H

Use the import/export utility to create and translate message strings into multiple languages. The .TXT import/export format supports all languages, including Chinese, Japanese, and Korean. The .CSV import/export format does not support Chinese, Japanese, or Korean.

Importing and exporting messages always performs a merge. Deleting a message in a .CSV file does not delete the message from the .ACD file. To delete a message, import the .CSV file with the type, name, and specifier fields filled in but the description blank.

When viewing alarm messages at the HMI:

- there is no default language string. If message text does not exist for a specific language, FactoryTalk View software searches for the first language that has a message string and displays that text.
- date and time format do not switch with the language. They follow the format of the operating system.
- nonconfigurable dialogs, such as ACK with description dialog, do not switch languages. They use the language of the operating system.

Alarm Process

At power up of the alarm system, the alarm establishes its initial connection to the controller, as follows:

1. The RSLinx Enterprise server initiates a subscription to the alarm.
2. The controller reserves 100 KB of buffer space for the subscriber.
3. The controller sends this information about all of its alarms to the subscriber:
   - Path and tag information
   - Alarm configuration
   - Message strings in all configured languages
4. Once the subscriber receives the discovery information, it requests a subscription to the alarm.

Each alarm typically transfers 500 bytes of data to the subscriber during this phase. A typical discovery phase for a system of 1000 alarms (500 analog and 500 digital) takes approximately 35 seconds This varies depending on controller loading, network loading, and message string size and languages.

During normal operation of the alarm system. in this phase, the controller sends alarm data to the subscriber, as follows:

1. When an alarm event occurs, the controller timestamps the alarm data and sends it to the subscriber.

   A typical status packet is 500 bytes for an analog alarm and 250 bytes for a digital alarm.

2. The subscriber sends the alarm data to the appropriate client applications and historical databases.

3. The operator acknowledges the alarm and the operator's acknowledge request is logged into the historical database. (This timestamp is from the operator workstation.)

4. The RSLinx Enterprise server sends the acknowledge request to the controller.

5. The controller receives the acknowledge request, marks the alarm as acknowledged, and timestamps the completed action back to the subscriber.

6. The controller sends the acknowledge confirmation with timestamp back to the subscriber. (This time stamp is from the controller.)

7. The subscriber sends the acknowledge to the appropriate clients and historical databases.

Because timestamps occur at multiple places during normal alarm operation, it is important to coordinate the clocks of the controllers and workstations in the system. For more information on, see Different Methods of Synchronizing Clocks with ControlLogix Controllers, Knowledgebase document 40467

Buffering Alarms

The controller reserves 100 KB per subscriber to buffer alarm data in the event that the subscriber loses its connection to the controller. Typically, this buffer holds about 1000 events.

Guideline	Description
If the subscriber loses its connection to the controller, re-establish the connection as soon as possible.	The alarm buffer in the controller continues to buffer new alarms until either the buffer is full (100 KB) or the buffer times out. You configure the buffer timeout from 0 min...2 hr (default is 20 min) when you configure the alarm server in RSLinx Enterprise software. If the subscriber fails to reconnect by the end of this buffer time, the controller clears the buffer and reclaims the 100 KB of buffer space for normal controller operations.
You can check the status of a subscriber connection to the controller by looking at the instruction faceplate in RSLogix 5000 software.	The Status tab on the alarm properties in RSLogix 5000 programming software identifies whether the controller is buffering alarm data. This value is updated at the next occurrence of the alarm event.

Programmatically Access Alarm Information

Each alarm instruction has an alarm structure that stores alarm configuration and execution information. The alarm structure includes both control program elements and operator elements. The alarm instructions do not use mode settings to determine whether program access or operator access is active, so these elements are always active.

There are three ways to perform actions on an alarm instruction.

Access	Alarm Structure Elements	Considerations
Control program	• ProgAck • ProgReset • ProgSuppress • ProgDisable • ProgEnable	Use controller logic to programmatically access elements of the alarming system. For example, the control program can determine whether to disable a series of alarms that are related to a single root cause. Then control program could disable an alarm instruction, MyDigitalAlarm of data type DigitalAlarm, accesses a tag MyDigitalAlarm.ProgDisable.
Custom HMI	• OperAck • OperReset • OperSuppress • OperDisable • OperEnable	Access a custom faceplate to access elements of the alarming system. For example, if the operator needs to remove a tool, rather than manually disable or suppress alarms individually from the alarming screens, the operator can press a disable key that accesses a tag MyDigitalAlarm.OperDisable.
Standard HMI object	Not accessible	Normal operator interaction is through the alarm summary and alarm banner objects in the FactoryTalk View application. This interaction is similar to the custom HMI option described above, but there is no programmatic visibility or interaction.

To perform global alarm operations, access the alarm elements via the relay ladder instructions. For example, assign a BOOL tag DisableToolA to all the ProgDisable fields on the alarm relat ladder faceplates in ToolA. Then use the DisableToolA tag to disable the operation of all the alarms that use this tag.

IMPORTANT

If you assign a tag to the ProgAck, ProgReset, ProgDisable, or ProgEnable functions on the alarm faceplate, do not use the alarm structure elements in the alarm data type to perform the same functions. For example, if you assign DisableToolA to disable an alarm MyDigitalAlarm, you should no longer programmatically access MyDigitalAlarm.ProgDisable. Doing so can cause a condition where the faceplate requests one operation and the alarm tag requests another.

If you want to use the alarm structure elements to programmatically change the alarming system, assign those elements to a faceplate. For example, on the alarm faceplate for ProgDisable, assign the tag from that alarm's structure tag MyAlarmTag.ProgDisable. This lets you programmatically access MyAlarmsTag.ProgDisable in other code locations without conflict.

At the HMI and in the event log, any controller-driven events, either through the alarm structure or the alarm dialog, are logged in the historical database as Discrete Events. This includes any HMI interface that also accesses this same information via the operator elements (.OPER*xxx*). So while timestamps and events are tracked in the log, the log does not include identification of what caused the event. For example, an operator and workstation in this scenario is not tracked because they did not take action via a FactoryTalk alarm graphic object.

Suppress or Disable Alarms

Suppress alarms to remove alarms you know exist from the HMI but still keep the alarms alive. This lets you clear the alarm summary while you are resolving a known alarm without continuing to view alarm information. A suppressed alarm does not appear on the operator summary or banner screens, but a suppressed alarm is still sent to subscribers, logged in the historical database, able to transition alarm status, time stamped, and responsive to other programmatic or operator interactions.

Disable an alarm to treat the alarm as if it does not exist in the control program. A disabled alarm does not transition alarm status or get logged in the historical database. A disabled alarm is still tracked, and can be re-enabled, in the Alarm Status Explorer in FactoryTalk View SE software.

Optimize an Application for Motion Control

Introduction

The Logix5000 controller contains a high-speed motion task which executes motion commands (relay ladder and structured text) and generates position and velocity profile information. The controller sends this profile information to one or more motion modules. RSLogix 5000 programming software provides complete axis configuration and motion programming support.

For more information on motion, see:

- the Motion Analyzer.
- Logix5000 Controllers Motion Instructions Reference Manual, publication 1756-RM007.
- ControlLogix Motion Module Setup and Configuration Manual, 1756-UM006.

Coarse Update Rate

The coarse update rate determines the periodic rate at which the motion task executes to compute the servo commanded position, velocity, and accelerations to be sent to the motion modules when executing motion instructions.

To calculate the coarse update rate:

- 2 * (task execution time + number of actions for every axis).
- divide the result by 1000 and round to the nearest ms.

If the coarse rate is too small, the controller might not have time to execute non-motion logic. As a general rule, one millisecond per axis is required by the motion task in order to allow the controller reasonable execution time.

The motion planner takes almost its entire minimum coarse iteration time. The coarse iteration time is minimally set 1 ms per axis. So if you have a periodic task running every 5 ms and 2 axes of motion, the motion planner runs twice consuming close to 4 of the 5 ms. In this case, it's possible to never finish executing the periodic task.

Axis Limits

Controller	Supported Motion Modules and Axes	Applications
ControlLogix	1756-M03SE (3 axes)	RA SERCOS drives
	1756-L60M03SE (3 axes) 1756-L60 controller with embedded SERCOS interface	RA SERCOS drives
	1756-M08SE (8 axes)	RA SERCOS drives
	1756-M16SE (16 axes)	RA SERCOS drives
	1756-M02AE (2 axes)	RA and third party: • Analog command signal • Quadrature encoder feedback
	1756-HYD02	RA and third party: • Analog command signal • Linear transducer feedback
	1756-M02AS	RA and third party: • Analog command signal • SSI feedback
SoftLogix	1784-PM16SE (16 axes) • Maximum of four 1784-PM16SE cards per computer • Associate only one 1784-PM16SE card with one controller	RA SERCOS drives
	1784-PM02AE (2 axes) • Maximum of four 1784-PM02AE cards per computer • Maximum of four 1784-PM02AE cards can be associated with one controller • Cannot associate a 1784-PM02AE motion card with the same controller as a 1784-PM16SE card	RA and third party: • Analog command signal • Quadrature encoder feedback

Performance Limits

The motion planner interrupts all other tasks, regardless of priority.

- The number of axes and coarse update period for the motion group effect how long and how often the motion planner executes.
- If the motion planner is executing when a task is triggered, the task waits until the motion planner is done.
- If the coarse update rate occurs while a task is executing, the task pauses to let the motion planner execute.

Motion Event Task Triggers

An event task executes automatically based on a preconfigured event occurring. There are different motion-based events.

To trigger an event task when	Use this trigger	With these considerations
Registration input for an axis turns on (or off)	Axis Registration 1 or 2	• In order for the registration input to trigger the event task, first execute a Motion Arm Registration (MAR) instruction. This lets the axis detect the registration input and in turn trigger the event task. • Once the registration input triggers the event task, execute the MAR instruction again to re-arm the axis for the next registration input. • If the scan time of your normal logic is **not** fast enough to re-arm the axis for the next registration input, consider placing the MAR instruction within the event task.
Axis reaches the position that is defined as the watch point	Axis Watch	• In order for the registration input to trigger the event task, first execute a Motion Arm Watch (MAW) instruction. This lets the axis detect the watch position and in turn trigger the event task. • Once the watch position triggers the event task, execute the MAW instruction again to re-arm the axis for the next watch position. • If the scan time of your normal logic is **not** fast enough to re-arm the axis for the next watch position, consider placing the MAW instruction within the event task
Motion planner completes its execution	Motion Group Execution	• The coarse update period for the motion group triggers the execution of both the motion planner and the event task. • Because the motion planner interrupts all other tasks, it executes first. If you assign the event task as the highest priority task, it executes after the motion planner.

For information on configuring an event task, see Chapter 2 Divide Logic into Tasks, Programs, Routines, and Add-On Instructions.

Notes:

Optimize an Application for Use with HMI

Introduction

Rockwell Automation offers these HMI (human-machine interface) platforms.

Platform	Description
PanelView Plus terminal	Dedicated, machine-level HMI running RSView Machine Edition software
RSView Enterprise software	Platform, product family consisting of: • RSView ME (Machine Edition) software for an open, machine-level HMI; also runs on PanelView Plus terminals • RSView SE (Supervisory Edition) Station software for a single-workstation, supervisory-level HMI • RSView SE distributed software for a multi-server, multi-client, supervisory-level HMI
RSView32 software	Single-workstation or single-server, multiple-client, supervisory-level HMI

Software products that provide plant-floor device connectivity for HMI applications includes:

- RSLinx Classic software, also known as RSLinx 2.*x*.
- RSLinx Enterprise software.

Decide How to Implement HMI

Method	Benefits	Considerations
Single HMI	• All HMI/EOI support this method • Limited number of controller connections • No server to setup and manage • Local control and monitoring	• Single point of failure for visualization • Only one person can monitor a single display at a time
Multiple, Independent HMI	• All HMI/EOI support this method • The same HMI screens can be viewed at multiple stations • Multiple people can monitor different parts of system simultaneously • Each HMI gets its own data • No central server to setup and manage • Local control and monitoring	• More controller connections are required • Additional burden on controller to service all communication (program scan impact) • No sharing of data except through the controller • Adding additional HMIs has larger increase on system
Client/Server HMI	• The same HMI screens can be viewed at multiple stations • Server provides data to multiple clients • Fewer controller connections required • Impact on system is smaller than with multiple HMIs • Administer application at the server, not individually at the clients or multiple, independent HMIs	• Server is a single point of failure for all HMIs, unless you implement redundancy • Little communication overhead savings if each client wants different data • Networking knowledge required

Most third-party HMIs are limited to direct communication similar to the multiple HMI method above.

Compare RSView Enterprise and RSView32 Software

HMI Product	Benefits	Considerations
RSView SE	• Supports Windows 2000, Windows XP, and Windows Server 2003 operating systems • Common RSView Studio development environment for RSView SE and RSView ME software (including PanelView Plus and VersaView CE terminals) • FactoryTalk enabled	• Does not support Windows NT operating system • Some RSView32 capabilities still under development in RSView SE
RSView32	• Support Windows NT, Windows 2000, Windows XP, and Windows Server 2003 operating systems • FactoryTalk enabled (version 7.0 and later)	• RSView32 development environment only supports RSView32 software • PanelBuilder software used for PanelView terminals • RSView32 software supports only single-server architectures

Guidelines for RSView SE Software

To set up your RSView Supervisory Edition HMI system successfully, a maximum of:

- five RSView Studio clients can have simultaneous access to an RSView SE application.
- 50 RSView SE clients can have simultaneous access to an RSView SE application.

In nonredundant applications, a maximum of:

- 10 RSView SE servers can be in an RSView SE application.
- two RSView SE servers can be hosted on a single computer.

In redundant applications, a maximum of:

- one RSView SE server can be hosted on a single computer.

Contact Rockwell Software for architectural assistance with redundant server applications or applications requiring more than 2 RSView SE Servers and 20 RSView SE Clients.

How RSLinx Software Communicates with Logix5000 Controllers

RSLinx software acts as a data server to optimize communication to HMI applications. RSLinx software groups data items into a single network packet to reduce the number of messages that get sent over the network and that need to be processed by a controller.

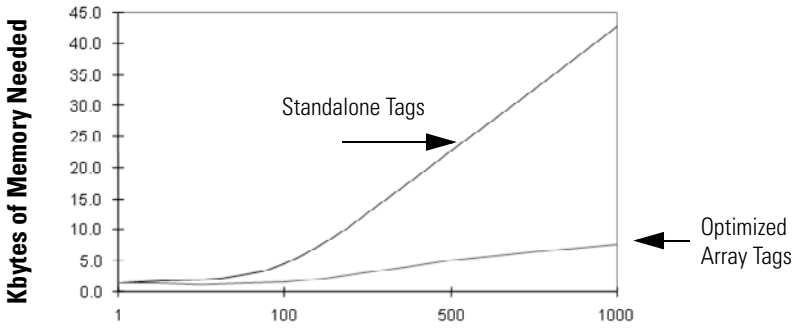
IMPORTANT Unless otherwise indicated, references to RSLinx software include both RSLinx Classic software and RSLinx Enterprise software.

1. When RSLinx software first connects to a Logix5000 controller, it queries the tag database and uploads definitions for all controller-scoped tags. If there are multi-layer, user-defined structures that are controller-scoped, RSLinx software just queries the upper layer.

2. When the HMI client requests data, RSLinx software queries the definitions for program-scoped tags and the lower layers of multi-layer user-defined structures.

3. RSLinx software receives requests for data items from local or remote HMI/EOI clients and combines multiple requests in optimized packets. Each data item is a simple Logix tag, array or user-defined structure. Each optimized packet can be as large as 480 bytes of data and can contain one or more data items.

4. The Logix5000 controller allocates unused system RAM to create an optimization buffer to contain the requested data items.
   - A single optimization buffer can contain as much data as will fit into a single 480-byte packet (optimization is limited to 480 bytes)
   - Currently, RSLinx Enterprise software only provides optimization for array tags
   - If you use RSLogix 5000 software to monitor controller RAM, you can see used memory increase
   - The controller creates an optimization buffer for each RSLinx optimization packet in the scan.



Compare RSLinx Classic and RSLinx Enterprise Software

Comparison	RSLinx Classic (RSLinx 2.*x*) Software	RSLinx Enterprise Software
Supported platforms	• Windows 98 • Windows ME • Windows NT • Windows 2000 • Windows XP • Windows Server 2003	• Windows CE • Windows 2000 • Windows XP • Windows Server 2003
Architecture	Single-threaded	Multi-threaded
Data server	OPC data server Preferred data server for PLC/SLC platforms and applications requiring complex network routings Maximum 10 clients per data server	Factory Talk Live data server Preferred data server for Logix5000 platforms Maximum 20 clients per data server
PLC/SLC systems	Maximum 20 controllers per data server via an Ethernet network	Maximum 20 controllers per data server via an Ethernet network
Logix5000 systems	Maximum: • 10 controllers per data server via an Ethernet network • 10,000 active (on-scan) tags per data server • three RSLinx data servers per controller	Maximum: • 20 controllers per data server via an Ethernet network • 20,000 active (on-scan) tags per data server • three RSLinx Enterprise data servers per controller
User interface and event logs	Yes	• Available user interfaces are FactoryTalk Studio software and FactoryTalk Administration Console software • Event logs are still under development
Benefits	• Supports topic switching with redundant ControlLogix system • Support used-defined tag optimization • RSLinx Gateway software consolidates multiple HMI requests to reduce network traffic • Works with an integrated OPC server	• Uses four read and one write uni-directional connections (fewer than RSLinx software) • Automatically handles Logix tag changes • FactoryTalk Live Data software consolidates multiple HMI requests to reduce network traffic
Considerations	• Requires HMI to be restarted if Logix5000 controller is reloaded with changes to tags on scan • Uses 4fourbi-directional connections	• Does not support topic switching with redundant ControlLogix system • Optimization limited to array tags • Does not yet support OPC • ActiveX faceplates require a separate OPC server

Guidelines for RSLinx Software

Guideline	Description
Use RSLinx software as the data server for multiple HMIs.	For multiple HMI stations: • leverage remote OPC (RSLinx Classic software) or FactoryTalk (RSLinx Enterprise software) software for data collection. • only the RSLinx data server should have an active topic. • do not configure or use topics on the HMI stations. • RSLinx software does not need to be on the HMI stations.
Do not use too many RSLinx stations.	The performance of tag collection decreases as the more RSLinx stations collect data from the same controller. Use an RSLinx Gateway station and have the other data collection stations use remote OPC for data collection
Account for delay time when adding/removing scanned tags.	When switching from one HMI screen to another, it takes time to put items in the controller on scan and take items off scan. Part of this time delay is due to the controller allocating system RAM for the optimization buffer. To eliminate this delay, when switching between HMI screens, put the items in the HMI screens on scan and leave them on scan. For example, you can create a data log to keep the items on scan. Then when switching between HMI screens, data collection continues without interruption. RSLinx Enterprise and RSView SE software account for this time delay. When HMI screens change, these applications deactivate tags rather than remove them from scan

Guidelines to Configure Controller Tags

Guideline	Description
Use INT data types with third party products.	Most third party operator interface products do not support DINT (32-bit) data types. However, there are additional performance and memory-use considerations when using INT data types. See Guidelines for Data Types on page 42. RSView software supports native Logix5000 data types (including BOOL, SINT, INT, DINT, and REAL), structures, and arrays.
Group related data in arrays.	Most third party operator interface products do not support user-defined structures. Arrays also ensure that data is in contiguous memory, which optimizes data transfer between the controller and RSLinx software or other operator interfaces. Arrays of tags transfer more quickly and take up less memory than groups of individual tags.
Map tags to PLC addresses.	To optimize data transfer between the controller and RSLinx software or other operator interfaces, use PLC mapped tags. The RSLinx topic must have the Optimize Poke Packets enabled. The RSView application must write the values through a DownloadAll command or the WritePendingValues VBA method.
Use RSLinx OPC services.	Use RSLinx OPC services to bundle multiple tag requests into a single message to reduce communication overhead. OPC provides better optimization than DDE.

Reference Controller Data from RSView Software

This table shows how to reference data in RSView tag address.

Logix5000 Array Data Type	Description	PLC File Identifier	RSView Tag Data Type
BOOL	Value of 0, 1, or -1	B	Digital
SINT	8-bit integer	A	Byte
INT	16-bit integer	N	Integer
DINT	32-bit integer	L	Long Integer
LINT	64-bit integer value to store date and time values	No PLC identifier	Not supported
REAL	Floating point	F	Floating Point

When addressing a Logix5000 string tag, use the address syntax [OPC_Topic]StringTag.Data[0],SC82 to address a SINT array. The string data is stored in the SINT array .Data of the string tag, and you address the first element of this array (.Data[0]). The maximum number of characters in a STRING tag is 82. If you need more characters than that, create your own user-defined structure to hold the characters.

If you write data into a Logix5000 string tag from an HMI or external source, you must set the L.EN field to indicate the number of characters that are in the string. RSLogix 5000 software and the controller use the .LEN value to determine how many characters are present.

Notes:

Develop Equipment Phases for Batch Control

Introduction

The PhaseManager option of RSLogix 5000 software (introduced in version 15) gives you a state model for your equipment. It includes the following components:

- Phase to run the state model
- Equipment phase instructions for programming the phase
- PHASE data type

Guidelines for Equipment Phases

Guideline	Description
Use a separate phase for each activity of the equipment.	Each phase is a specific activity that the equipment performs. • Use one phase for standalone machines. • Make sure each phase does an independent activity. • Keep the total number of phases and programs in a project within the limit of programs for the controller. • List the equipment that goes with each phase.
Complete one state model for each phase.	Each phase runs its own set of states. A state model divides the operating cycle of the equipment into a series of states. • Decide which state to use for the initial state after power-up. • Start with the initial state and work through the model. • Use only the states you need; skip those states that do not apply. • Use subroutines for producing and standby states. The state model of an equipment phase is similar to the S88 state model. U.S. standard ISA S88.01-1995 and its IEC equivalent IEC 61512-1-1998 is commonly referred to as S88. It is a set of models, terms, and good practices for the design and operation of manufacturing systems.
Separate phase code from equipment code.	One advantage of a phase is that it lets you separate the procedures (recipes) for how to make the product from the control of the equipment that makes the product. This makes it easier to execute different procedures for different products using the same equipment.

Guideline	Description
Separate normal execution from exceptions.	A state model makes it much easier to separate the normal execution of your equipment from any exceptions (faults, failures, off-normal conditions). • Use a prestate routine to watch for faults. A prestate routine is not a phase state routine. Create a routine like you would for any program and assign it as the prestate routine for the equipment phase program. • Use a state bit to limit code to a specific state. • RSLogix 5000 software automatically makes a tag for each phase. The phase tag has bits that identify the state of the phase. For example, My_Phase.Running.
Use Equipment Phases on nonredundant controllers only.	ControlLogix controller redundancy requires a great deal of additional validation. As a result, the PhaseManager option has been tested only in a nonredundant controller environment. If you need redundant controllers, use the Phase Logic Interface (PLI) application code or isolate Equipment Phases to nonredundant controllers.

Equipment Phase Instructions

The equipment phase instructions are available in relay ladder and structured text programming languages. You can use them in relay ladder routines, structured text routines, and SFC actions.

If you want to	Use this instruction
Signal a phase that the state routine is complete so go to the next state	Phase State Complete (PSC)
Change the state or substate of a phase	Equipment Phase Command (PCMD)
Signal a failure for a phase	Equipment Phase Failure (PFL)
Clear the failure code of a phase	Equipment Phase Clear Failure (PCLF)
Initiate communication with RSBizWare Batch software	Equipment Phase External Request (PXRQ)
Clear the NewInputParameters bit of a phase	Equipment Phase New Parameters (PRNP)
Set up breakpoints within the logic of a phase	Equipment Phase Paused (PPD)
Take ownership of a phase to either: • prevent another program or RSBizWare Batch software from commanding a phase. • make sure another program or RSBizWare Batch software does **not** already own a phase.	Attach to Equipment Phase (PATT)
Relinquish ownership of a phase	Detach from Equipment Phase (PDET)
Override a command	Equipment Phase Override (POVR)

For more information, see the PhaseManager User Manual, LOGIX-UM001.

Optimize an Application for Process Control

Introduction

The Logix5000 controller integrates a function block diagram editor and several process control instructions. The controller can generally execute more loops than typical applications require.

Comparison PID and PIDE Instructions

The function block PIDE instruction offers additional enhancements over the relay ladder PID instruction.

Enhanced PID (PIDE)	Standard PID
Velocity form algorithm works on change in error value This algorithm is the same type as used in most DCS systems. The algorithm also makes it easier to implement adaptive gains.	Position form algorithm which works on error values
Full set of modes • Program/operator control • Cascade/ratio • Auto • Manual • Override • Hand	Limited set of modes • Auto • Software manual (similar to PIDE manual mode) • Manual (similar to PIDE hand mode)
Available selection of timing modes • Periodic • Oversample • Real time sampling	No timing modes
Handling for PV/CV faults The PIDE block has built-in PVFault and CVFault members.	No handling for PV/CV faults
Full bumpless transfer into and out of Cascade mode	No bumpless transfer into or out of Cascade mode

Guidelines to Program PID Loops

Guideline	Description
Place PID loops in a periodic task.	Configure the periodic tasks to execute at the desired rate.
Estimate the number of loops based on task execution time.	Estimate the number of PID loops that can be executed as: (execution time of periodic tasks in ms) / 2 This leaves sufficient time for the controller to manage other logic in lower-priority tasks.

For typical applications, estimate 1 ms of program scan per loop. This includes the PID instruction and the other code associated with the loop.

Advanced Process Instructions

Instruction	Description
Alarm (ALM)	Provides alarming for any analog signal.
Enhanced PID (PIDE)	Provides enhanced capabilities over the standard PID instruction. The instruction uses the velocity form of the PID algorithm. The gain terms are applied to the change in the value of error or PV, not the value of error or PV.
Ramp/Soak (RMPS)	Provides for a number of segments of alternating ramp and soak periods.
Scale (SCL)	Converts an unscaled input value to a floating point value in engineering units.
Position Proportional (POSP)	Opens or closes a device by pulsing open or close contacts at a user-defined cycle time with a pulse width proportional to the difference between the desired and actual positions.
Split Range Time Proportional (SRTP)	Takes the 0...100% output of a PID loop and drives heating and cooling digital output contacts with a periodic pulse.
Lead Lag (LDLG)	Provides a phase lead-lag compensation for an input signal.
Function Generate (FGEN)	Converts an input based on a piece-wise linear function.
Totalizer (TOT)	Provides a time-scaled accumulation of an analog input value.
Deadtime (DEDT)	Performs a delay of a single input. You select the amount of deadtime delay.
Discrete 2-State Device (D2SD)	Controls a discrete device which has only two possible states such as on/off and open/closed.
Discrete 3-State Device (D3SD)	Controls a discrete device having three possible states such as fast/slow/off and forward/stop/reverse.

Faceplates

RSLogix 5000 software includes faceplates for some function block instructions. These faceplates are Active-X controls that read the entire data structure for the associated instruction. Use these faceplates with RSView software or any other application that acts as an Active-X container.

IMPORTANT RSLogix 5000 programming software is not a valid Active-X container.

The faceplates communicate with the controller via the RSLinx OPC server available with RSLinx Classic software. The RSLinx OPC server is not available in the RSLinx Lite software that comes with RSLogix 5000 software. You have to purchase a package such as RSLinx Classic OEM, Professional, or Gateway software.

These instructions have faceplates:

- Alarm (ALM)
- Enhanced Select (ESEL)
- Totalizer (TOT)
- Ramp/Soak (RMPS)
- Discrete 2-State Device (D2SD)
- Discrete 3-State Device (D3SD)
- Enhanced PID (PIDE)

FactoryTalk View software, version CPR 7 and later, also includes a set of these instruction faceplates for use with FactoryTalk View SE and ME applications. These faceplates require RSLinx Enterprise software and use native objects in the FactoryTalk Studio development environment. These faceplates are installed along with FactoryTalk View Studio software.

Compare Active-X Faceplates and Graphic Library Elements

Active-X Faceplates	Graphic Library Elements
Requires RSLinx Classic software	Requires RSLinx Enterprise software
For use with Active-X containers, such as RSView 32 or Microsoft Office products	For use with RSView SE or RSView ME software
Only supported on personal computer platforms	Supports all supervisory-edition and machine-edition platforms
Faceplates incorporate checkboxes and radio buttons	Graphics incorporate buttons and text boxes
Does not allow customization	Allows customization

Manage Firmware

Introduction

The Logix controllers, I/O modules, and other devices use firmware that you can update on your own. You choose the firmware revision level and decide when to update the firmware.

Guidelines to Manage Controller Firmware

Guideline	Description
Maintain software versions and firmware revisions at the same major revision levels.	At release, a specific version of software supports the features and functions in a specific revision of firmware. To use a specific revision of firmware, you must have the corresponding software version. This combination of software and firmware is considered to be compatible. A revision number consists of a major and minor revision number in this format xx.yy. **Where** / **Is the** xx: Major revision; Updated every release there is a functional change yy: Minor revision; Updated any time there is a change that does not affect function or interface
Document firmware revisions.	Include software version and firmware revision information in electrical drawings and other project documentation.
Read the associated release notes.	Always read the release notes that accompany new software versions and firmware revisions before you install them. These release notes help you to understand what has improved and changed, and also help you determine whether you need to modify your application because of the changes. In most cases, your application will run normally following an update.
Configure modules so that the controller automatically updates firmware.	Controller firmware, revision 16, includes a firmware supervisor feature that lets controllers automatically flash update devices. To use the firmware supervisor: • local and remote modules can be flashed while in Program or Run modes, as long as their Electronic keying configurations are set to Exact Match and they are supported by the ControlFlash utility. • firmware kits must reside in the controller's CompactFlash card.

Compare Firmware Options

Controllers ship with basic firmware that supports only updating the controller firmware to the required revision. You must update the firmware to a revision that is compatible with your version of RSLogix 5000 software.

ControlFlash Utility	AutoFlash Function	Controller-based Firmware Supervisor
Standalone tool. Manually launch from desktop icon or program list.	Integrated with RSLogix 5000 programming software. The software automatically checks the controller, motion module, and SERCOS drive firmware during a project download. If the firmware is out of date or incompatible, the software prompts you to update the firmware.	Integrated on the controller CompactFlash card and run by the controller without user intervention. Controllers automatically flash update modules on keying mismatch situations.
Supports controllers, communication modules, I/O modules, motion modules, and newer SERCOS drives, as well as many other devices. As of RSlogix 5000 software, version 16, the utility supports these SERCOS drives: • 1394 drives, firmware revision 1.85 and later • Kinetix 6000 drives, firmware revision 1.85 and later • Ultra3000 drives, firmware revision 1.50 and later • 8720MC drives, firmware revision 3.85 and later	Supports the same devices as the ControlFlash utility.	Supports local and remote devices that: • are in the I/O tree and configured as Exact Match. • support firmware upgrades via the ControlFlash utility. • the hardware revision supports the firmware stored for that Exact Match device.
Supports valid CIP path to the device to update, such as serial, DeviceNet, ControlNet, and EtherNet/IP connections.	Supports valid CIP path to the device to update, such as serial, DeviceNet, ControlNet, and EtherNet/IP connections.	Supports all communication paths to devices that reside in the controller I/O tree and that also support the ControlFlash utility. The firmware must already be on a CompactFlash card in the controller.

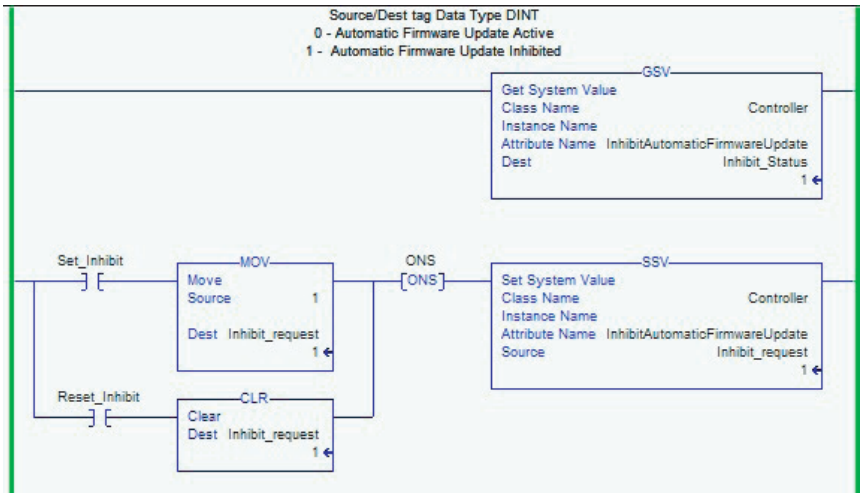
For more information, see the ControlFlash Firmware Upgrade Kit User Manual, publication 1756-6.5.6

Guidelines for the RSLogix 5000 Firmware Supervisor

As of controller firmware, revision 16, the RSLogix 5000 firmware supervisor feature can automatically load firmware when you replace a device in the system.

- OEMs who build multiple machines a month can have the controller flash all the modules and devices in the system without user intervention.
- Machines with strict regulation may require specific firmware revisions for the devices to maintain certification. The firmware supervisor helps make sure devices are at the correct firmware revision.
- Maintenance personnel replacing failed hardware can install the replacement device and the controller automatically flashes the device with the correct firmware revision.

Guideline	Description
The firmware supervisor will flash any Rockwell Automation device that: • can be placed in the I/O Configuration tree. • has electronic keying configured as Exact Match. • can normally be flashed by using ControlFlash software.	The firmware supervisor works on local I/O modules and distributed modules via EtherNet/IP, SERCOS, and ControlNet networks. On DeviceNet networks, the firmware supervisor supports local devices only, such as scanners and linking devices that reside in the I/O tree of the controller project. Because you cannot directly place a remote DeviceNet device in the I/O tree, the firmware supervisor does not manage remote DeviceNet devices. The firmware supervisor supports: • Logix5000 controllers that support CompactFlash memory cards, except for GuardLogix Safety controllers and ControlLogix redundant controllers. The firmware supervisor does not manage the firmware of other controllers in the I/O Configuration tree. These controllers manage firmware on their own CompactFlash cards. • SERCOS drives that support flash updates over a SERCOS network. This includes: – 1394 drives, firmware revision 1.85 and later. – Kinetix 6000 drives, firmware revision 1.85 and later. – Ultra3000 drives, firmware revision 1.50 and later. – 8720MC drives, firmware revision 3.85 and later. • non-modular, distributed I/O products that sit directly on the network without an adapter. Distributed I/O products that require an adapter, such as POINT I/O or FLEX I/O modules, are not supported. Instead, the firmware supervisor manages the firmware for the adapters. PanelView Plus terminals do not support the ControlFlash software, so they are not supported by the firmware supervisor.
For the firmware supervisor to manage firmware for a device, that device must have its electronic keying configured for Exact Match.	Other modules can exist in the I/O Configuration that are not configured as Exact Match, but the firmware supervisor will not maintain the firmware for those modules. To disable the firmware supervisor for a specific device: • change the electronic keying for that device to something besides Exact Match. • disable firmware supervisor from either an SSV instruction or the Nonvolatile Memory tab of the controller properties.
The CompactFlash card must be formatted FAT16.	If you have a CompactFlash card that is formatted FAT32, you must reformat the card FAT16.

Guideline	Description
Each controller must store the firmware files for modules managed by the firmware supervisor on a CompactFlash card.	Enable the firmware supervisor, from the Nonvolatile Memory tab of the controller properties. Click Load/Store. From the Automatic Firmware Updates pull-down menu, choose Store to copy it to the CompactFlash card. The computer running RSLogix 5000 software must have: • ControlFlash utility installed. • the required firmware kits in the ControlFlash default directory for the modules the firmware supervisor is to maintain. RSlogix 5000 software will move firmware kits from your computer to the controller CompactFlash card for the firmware supervisor to use. Controller firmware and application logic is managed outside of firmware supervisor on the Nonvolatile Memory tab. Firmware supervisor adds to the ability to store controller firmware and logic on the CompactFlash card. If you disable the firmware supervisor, you just disable the firmware supervisor updates and not the controller firmware updates that still occur when the controller image is reloaded.
Enable or disable the automatic firmware updates by using GSV and SSV instructions.	
You can monitor the status of automatic firmware updates.	Monitor the status of automatic firmware updates on the Nonvolatile Memory tab on the controller properties. To monitor the status of automatic firmware updates for a specific module, use GSV instructions. This example shows that the firmware supervisor encountered the wrong hardware revision for 1756-OB16D module. 

Access Firmware

RSLogix 5000 software ships with firmware update kits. Firmware revisions are also available on the Rockwell Automation website.

1. Go to support.rockwellautomation.com
2. In the left pane, under Downloads, click Firmware updates.

Notes:

A

Term	Definition
Add-On Instruction	An Add-On Instruction is a user-defined instruction that encapsulate executable logic and data.
array	An array groups data of the same data type under a common name. An array tag occupies a contiguous block of memory in the controller, each element in sequence.
atomic data type	BOOL, SINT, INT, DINT, LINT, and REAL data types.

B

Term	Definition
buffer	A temporary memory area used for queuing incoming and outgoing messages. The buffer area of a device determines how many messages can be queued for processing.

C

Term	Definition
cache	To leave a connection open for a MSG instruction that executes repeatedly.
coarse update rate	Determines the periodic rate at which the motion task executes to compute the servo commanded position, velocity, and accelerations to be sent to the motion modules when executing motion instructions.
compound data type	Array, structure, and string data types.
connection	A communication link between two devices, such as between a controller and an I/O module, PanelView terminal, or another controller. • Connections are allocations of resources that provide more reliable communication between devices than unconnected messages. • You indirectly determine the number of connections the controller uses by configuring the controller to communicate with other devices in the system.
consumed tag	A tag that receives the data that is broadcast by a produced tag over an EtherNet/IP network, ControlNet network, or ControlLogix backplane. A consumed tag must be: • controller scope. • same data type (including any array dimensions) as the remote tag (produced tag). See produced tag.
continuous task	The continuous task runs continuously in the background. Any CPU time not allocated to other operations (such as motion, communication, and periodic tasks) is used to execute the programs within the continuous task.
controller scope	Data accessible anywhere in the controller. The controller contains a collection of tags that can be referenced by the routines and alias tags in any program, as well as other aliases in the controller scope. See program scope.

D

Term	Definition
direct connection	A direct connection is a real-time, data transfer link between the controller and an I/O module. The controller maintains and monitors the connection with the I/O module. Any break in the connection, such as a module fault or the removal of a module while under power, sets fault bits in the data area associated with the module. See rack-optimized connection.

E

Term	Definition
element	An addressable unit of data that is a sub-unit of a larger unit of data. A single unit of an array or structure.
equipment phase	An equipment phase is a type of program. It has routines and a set of isolated tags. It also has: • state model. • state machine. • PHSAE data type.
event task	An event task executes automatically based on a trigger event occurring or if a trigger event does not occur in a specific time interval.
explicit	A connection that is non-time critical and is request/reply in nature. Executing a MSG instruction or executing a program upload are examples of explicit connections. Explicit refers to basic information (such as source address, data type, and destination address) that is included in every message. See implicit.

F

Term	Definition
firmware revision	For products that have firmware components, the product ID label identifies the firmware revision. This revision denotes the operating system for the device. The firmware revision is usually two numbers separated by a period. For example, in firmware revision 10.02, the first number (10) defines the major revision and the second number (02) defines the minor revision. See software version.

H

Term	Definition
HART protocol	HART (Highway Addressable Remote Transmitter) is an open protocol designed to connect analog devices. For HART connectivity, select from products available from Rockwell Automation and our Encompass partners.

I

Term	Definition
implicit	A connection that is time critical in nature. This includes I/O and produced/consumed tags. Implicit refers to information (such as source address, data type, and destination address) that is implied in the message but not contained in the message. See explicit.
index	A reference used to specify an element within an array.

L

Term	Definition
local connection	A connection to a module in a local chassis, extended-local chassis, or any of the I/O banks configured for the controller. Communication occurs across the backplane or virtual backplane and does not require an additional communication module or adapter.

M

Term	Definition
member	An element of a structure that has its own data type and name. • Members can be structures as well, creating nested structure data types. • Each member within a structure can be a different data type.
message	A message asynchronously reads or writes a block of data to another device.

N

Term	Definition
network update time (NUT)	The repetitive time interval in which data can be sent on a ControlNet network. The network update time ranges from 2...100 ms.

P

Term	Definition
parameter	A parameter is a value or tag passed to an instruction or returned from an instruction. An Add-On Instruction supports these parameters: • Input (copied in) • Output (copied out) • InOut (passed by reference)
periodic task	A periodic task executes automatically based on a preconfigured interval. This task is similar to selectable timed interrupts in PLC-5 and SLC 500 processors.
PhaseManager option	The PhaseManager option of RSLogix 5000 software (introduced in version 15) gives you a state model for your equipment. Use the PhaseManager option to create equipment phase programs.
postscan	A function of the controller where the logic within a program is examined before disabling the program in order to reset instructions and data.
prescan	Prescan is an intermediate scan during the transition to Run mode. • The controller performs prescan when you change from Program mode to Run mode. • The prescan examines all programs and instructions and initializes data based on the results. • Some instructions execute differently during prescan than they do during the normal scan.
produced tag	A tag that a controller is making available for use by other controllers. Produced tags are always at controller scope. See consumed tag.
product-defined data type	A structure data type that is automatically defined by the software and controller. By configuring an I/O module, you add the product-defined data type for that module.
program	A set of related routines and tags. Each program contains program tags, a main executable routine, other routines, and an optional fault routine.
program scope	Data accessible only within the current program. Each program contains a collection of tags that can only be referenced by the routines and alias tags in that program. See controller scope.

R

Term	Definition
rack-optimized connection	For digital I/O modules, you can select rack-optimized communication. A rack-optimized connection consolidates connection usage between the controller and all the digital I/O modules in the chassis (or DIN rail). Rather than having individual, direct connections for each I/O module, there is one connection for the entire chassis (or DIN rail). See direct connection.
remote connection	A connection to a module in a remote chassis or DIN rail. Communication requires a communication module and/or adapter.
requested packet interval (RPI)	When communicating over a the network, this is the maximum amount of time between subsequent production of input data. • Typically, this interval is configured in microseconds. • The actual production of data is constrained to the largest multiple of the network update time that is smaller than the selected RPI.
routine	A set of logic instructions in a single programming language, such as a ladder diagram. Routines provide the executable code for the project in a controller (similar to a program file in a PLC or SLC controller).

S

Term	Definition
scheduled connection	A scheduled connection is unique to ControlNet communication. A scheduled connection lets you send and receive data repeatedly at a predetermined rate, which is the requested packet interval (RPI). For example, a connection to an I/O module is a scheduled connection because you repeatedly receive data from the module at a specified rate. Other scheduled connections include connections to: • communication devices. • produced/consumed tags. On a ControlNet network, you must use RSNetWorx for ControlNet software to enable all scheduled connections and establish a network update time (NUT).
software version	The product ID label of a software products identifies the software version. This version denotes the functional version of the software. The software version is usually two numbers separated by a period. For example, in software version 10.02, the first number (10) defines the major revision and the second number (02) defines the minor revision. See firmware revision.
state machine	A state machine: • calls the maine routine (state routine) for an acting state. • manages the transitions between states with minimal coding. • makes sure that the equipment goes from state to state along an allowable path.
state model	A state model divides the operating cycle of your equipment into a series of states. Each state is an instant in the operation of the equipment. It's the actions or conditions of the equipment at a given time.
structure	Some data types are a structure. • A structure stores a group of data, each of which can be a different data type. • Within a structure, each individual data type is called a member. • Like tags, members have a name and data type. • You create your own user-defined structure, using any combination of individual tags and most other structures. • To copy data to a structure, use the COP instruction.
system overhead timeslice	Specifies the percentage of controller time (excluding the time for periodic tasks) that is devoted to communication and background functions (system overhead).

T

Term	Definition
tag	A named area of controller memory where data is stored. Tags are the basic mechanism for allocating memory, referencing data from logic, and monitoring data.
task	A scheduling mechanism for executing a program. By default, each new project file contains a preconfigured continuous task. You configure additional periodic and event tasks, as needed.

U

Term	Definition
unconnected message	An unconnected message is a message that does not require connection resources. An unconnected message is sent as a single request/response.
user-defined data type (UDT)	A UDT is a data structure you define. A user-defined data type groups different types of data into a single named entity. You define the members of the user-defined data type. Like tags, the members have a name and data type.

V

Term	Definition
virtual communication relationship (VCR)	A VCR is a channel that provides for the transfer of data between FOUNDATION Fieldbus devices. The number of VCRs required to send data or receive data depends on the device and type of data. The type of VCR determines whether the transfer is scheduled or unscheduled. • A client/server VCR is for queued, unscheduled, user-initiated, and one-to-one communication. • A report distribution VCR is for queued, unscheduled, user-initiated, and one-to-many communication. • A publisher/subscriber VCR is for buffered, one-to-many communication.

Notes:

Publications

You can view or download publications at http://www.literature.rockwellautomation.com. To order paper copies of technical documentation, contact your local Rockwell Automation distributor or sales representative.

RSLogix 5000 programming software includes PDF files of these publications, in addition to online help and a tutorial.

Resource	Description
• EtherNet/IP Modules in Logix5000 Control Systems User Manual, ENET-UM001 • ControlNet Modules in Logix5000 Control Systems User Manual, CNET-UM001 • DeviceNet Modules in Logix5000 Control Systems User Manual, DNET-UM004	Networks
• Logix5000 Controllers Quick Start, 1756-QS001 • Logix5000 Common Procedures Programming Manual, 1756-PM001 • SFC and ST Programming Languages Programming Manual,1756-PM003 (this manual is an excerpt from the Logix5000 Controllers Common Procedures Programming Manual) • Logix5000 Controllers General Instructions Reference Manual, 1756-RM003 • Logix5000 Controllers Process Control and Drives Instructions Reference Manual, 1756-RM006 • Phase Manager User Manual, LOGIX-UM001 • Logix5000 Controllers Motion Instructions Reference Manual, 1756-RM007 • Logix5000 Controllers System Reference Manual, 1756-QS107 • Logix5000 Controllers Import/Export Reference Manual, 1756-RM084G • Converting PLC-5 or SLC 500 Logic to Logix5000 Logic Reference Manual, 1756-RM085	Logix5000 Controllers
• ControlLogix Controllers Installation Instructions, 1756-IN101 • ControlLogix System User Manual, 1756-UM531 • ControlLogix Motion Module Setup and Configuration Manual, 1756-UM006	ControlLogix Controllers
• 1768 CompactLogix Controllers Installation Instructions, 1768-IN004 • 1768 CompactLogix System User Manual, 1768-UM001	1768 CompactLogix Controllers
• 1769-L31 CompactLogix Controllers Installation Instructions, 1769-IN069 • 1769-L32E, -L35E CompactLogix Controllers Installation Instructions, 1769-IN020 • 1769-L32C, -L35CR CompactLogix Controllers Installation Instructions, 1769-IN070 • CompactLogix System User Manual, 1769-UM011	1769 CompactLogix Controllers
• FlexLogix Controllers Installation Instructions, 1794-IN002 • FlexLogix System User Manual, 1794-UM001	FlexLogix Controllers
• SoftLogix Controllers Installation Instructions, 1789-IN001 • SoftLogix System User Manual, 1789-UM002	SoftLogix Controllers

Websites

Resource	Description
http://www.ab.com/logix/	Logix Product Information
http://www.ab.com/networks/	NetLinx Product Information
http://support.rockwellautomation.com In the left pane under Downloads, select Software Updates.	Software Updates (product serial number required)
http://support.rockwellautomation.com In the left pane under Downloads, select Firmware Updates.	Firmware Updates (product serial number required)
http://www.ab.com/networks/eds/	Rockwell Automation EDS Files
http://literature.rockwellautomation.com	Rockwell Automation Manuals
http://samplecode.rockwellautomation.com	RSLogix 5000 Sample Code

Notes:

How Are We Doing?

Your comments on our technical publications will help us serve you better in the future. Thank you for taking the time to provide us feedback.

You can complete this form and mail it back to us, visit us online at www.ab.com/manuals, or email us at RADocumentComments@ra.rockwell.com

Pub. Title/Type Logix5000 Controllers Design Considerations Reference Manual

Cat. No. Logix controllers Pub. No. 1756-RM094C-EN-P Pub. Date June 2007 Part No. 953030-20

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

Overall Usefulness	1	2	3	How can we make this publication more useful for you?
Completeness (all necessary information is provided)	1	2	3	Can we add more information to help you? procedure/step illustration feature example guideline other explanation definition
Technical Accuracy (all provided information is correct)	1	2	3	Can we be more accurate? text illustration
Clarity (all provided information is easy to understand)	1	2	3	How can we make things clearer?
Other Comments				You can add additional comments on the back of this form.

Your Name _____ Location/Phone _____

Your Title/Function _____

Would you like us to contact you regarding your comments?

___No, there is no need to contact me

___Yes, please call me

___Yes, please email me at _____

___Yes, please contact me via _____

Return this form to: Allen-Bradley Marketing Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Phone: 440-646-3176 Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

PLEASE FOLD HERE

PLEASE REMOVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE

Allen-Bradley
RELIANCE ELECTRIC
DODGE
ROCKWELL SOFTWARE
Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://support.rockwellautomation.com, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect Support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit http://support.rockwellautomation.com.

Installation Assistance

If you experience a problem with a hardware module within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your module up and running.

United States	1.440.646.3223 Monday – Friday, 8am – 5pm EST
Outside United States	Please contact your local Rockwell Automation representative for any technical support issues.

New Product Satisfaction Return

Rockwell tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning, it may need to be returned.

United States	Contact your distributor. You must provide a Customer Support case number (see phone number above to obtain one) to your distributor in order to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for return procedure.

www.rockwellautomation.com

Corporate Headquarters
Rockwell Automation, 777 East Wisconsin Avenue, Suite 1400, Milwaukee, WI, 53202-5302 USA, Tel: (1) 414.212.5200, Fax: (1) 414.212.5201

Headquarters for Allen-Bradley Products, Rockwell Software Products and Global Manufacturing Solutions
Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe: Rockwell Automation SA/NV, Vorstlaan/Boulevard du Souverain 36-BP 3A/B, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Headquarters for Dodge and Reliance Electric Products
Americas: Rockwell Automation, 6040 Ponders Court, Greenville, SC 29615-4617 USA, Tel: (1) 864.297.4800, Fax: (1) 864.281.2433
Europe: Rockwell Automation, Brühlstraße 22, D-74834 Elztal-Dallau, Germany, Tel: (49) 6261 9410, Fax: (49) 6261 17741
Asia Pacific: Rockwell Automation, 55 Newton Road, #11-01/02 Revenue House, Singapore 307987, Tel: (65) 351 6723, Fax: (65) 355 1733

Publication 1756-RM094C-EN-P - June 2007 PN 953030-20
Supersedes Publication 1756-RM094B-EN-P- August 2005