Allen-Bradley

Logix5000 Controllers

1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix

System Reference

Rockwell Automation

Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of these products must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards. In no event will Allen-Bradley be responsible or liable for indirect or consequential damage resulting from the use or application of these products.

Any illustrations, charts, sample programs, and layout examples shown in this publication are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Summary of Changes

This version of the Logix5000 Controllers System Reference Manual corresponds to version 15 of the controllers. Revision bars (shown in the left margin of this page) indicate changed information. Changes made to this manual include:

- Addition of 1769-L32C and 1769-L35CR CompactLogix controllers for ControlNet
- Addition of DriveLogix5730 controller for PowerFlex 700S
- Addition of PSC, PCMD, POVR, PFL, PCLF, PXRQ, PRNP, PPD, PATT and PDET phase manager instructions
- The 1794-L33, 1769-L20, and 1769-L30 controllers have been removed

Notes:

Chapter 1

Logix Controllers

Chapter 2

Sequential Function Charts

Chapter 3

Structured Text

Chapter 4

Function Block Diagram

Chapter 5

Relay Ladder

Chapter 6

Accessing System Values

Chapter 7

Communicate with Other Controllers

Chapter 8

Forcing

Chapter 9

System Faults

Chapter 10

Data Structures

Chapter 11

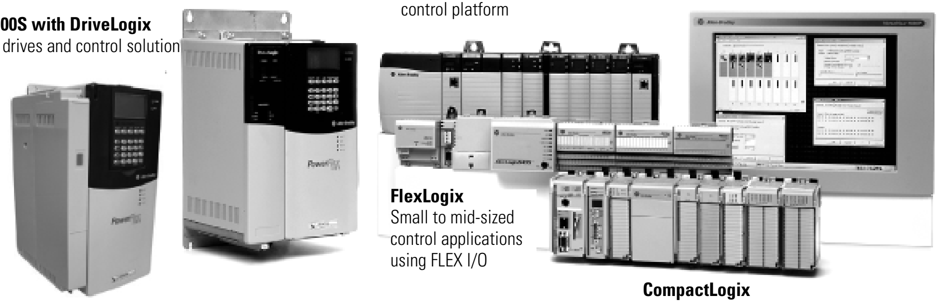
Instruction Set

Logix Controllers *Chapter 1*

Logix Family of Controllers

Rockwell Automation Logix Platforms provide a single integrated control architecture for discrete, drives, motion, and process control.

The integrated Logix architecture provides a common control engine, programming software environment, and communication support across multiple hardware platforms. All Logix controllers operate with a multitasking, multiprocessing operating system and support the same set of instructions in multiple programming languages. One RSLogix 5000 programming software package programs all Logix controllers. And all Logix controllers incorporate the NetLinx architecture to communicate via EtherNet/IP, ControlNet, and DeviceNet networks.



ControlLogix
High-performance, multi-processing control platform

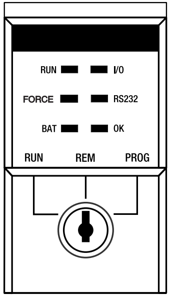
PowerFlex 700S with DriveLogix
An integrated drives and control solution

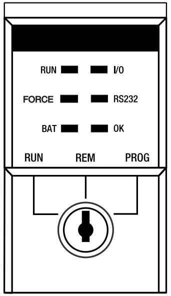
SoftLogix5800
High-performance, PC-based control

FlexLogix
Small to mid-sized control applications using FLEX I/O

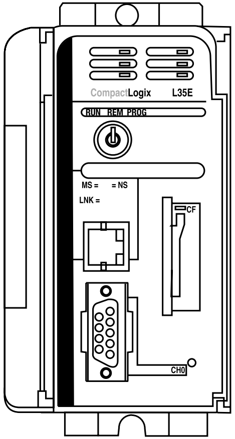
CompactLogix
Compact I/O and control for smaller applications

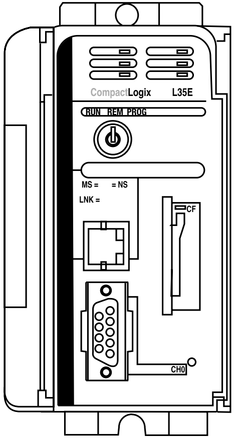
ControlLogix Controllers (1756-L6x, L55Mxx)

Front Panel:	Indicator:	Color:	Description:
 RUN I/O FORCE RS232 BAT OK RUN REM PROG	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
		solid green	The controller is communicating with all the devices in its I/O configuration.
		flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
		flashing red	The chassis is bad. Replace the chassis.
	FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		solid amber	I/O forces are active (enabled). I/O force values may or may not exist.
		flashing amber	One or more input or output addresses have been forced to an On or Off state, but the forces have not been enabled.
	RS232	off	There is no activity.
		solid green	Data is being received or transmitted

Front Panel:	Indicator:	Color:	Description:
 RUN I/O FORCE RS232 BAT OK RUN REM PROG	BAT	off	The battery supports memory.
		solid red	Either the battery is: • not installed. • 95% discharged and should be replaced.
	OK	off	No power is applied.
		flashing red	If the controller is:Then: a new controllerthe controller requires a firmware update not a new controllerA major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	The controller is OK.
		flashing green	The controller is storing or loading a project to or from nonvolatile memory.

CompactLogix Controllers (1769-Lxx)

Front Panel:	Indicator:	Color:	Description:
 CompactLogix L35E RUN REM PROG MS = = NS LNK = CF CH0	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		solid amber	I/O forces are active (enabled). I/O force values may or may not exist.
		flashing amber	One or more input or output addresses have been forced to an On or Off state, but the forces have not been enabled.
	BAT	off	The battery supports memory.
		solid red	Either the battery is: • not installed. • 95% discharged and should be replaced.
	I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
		solid green	The controller is communicating with all the devices in its I/O configuration.
		flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
		flashing red	The controller is not communicating to any devices. The controller is faulted.

Front Panel:	Indicator:	Color:	Description:
 CompactLogix L35E RUN REM PROG MS = = NS LNK = CF CH0	OK	off	No power is applied.
		flashing red	If the controller is:Then: a new controllerthe controller requires a firmware update not a new controllerA major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	The controller is OK.
		flashing green	The controller is storing or loading a project to or from nonvolatile memory.
	DCH0 (RS-232)	off	User-configured communications are active.
		solid green	Default communications are active.
	Channel 1 (RS-232) (1769-L31, -L30 only)	off	There is no activity.
		solid green	Data is being received or transmitted.

CompactLogix Controllers (1769-L31, -L32E, -L35E, -L32C, -L35CR) - CompactFlash

Indicator:	Color:	Description:
CompactFlash CF	off	No activity.
	flashing green	The controller is reading from or writing to the CompactFlash card.
	flashing red	CompactFlash card does not have a valid file system.

CompactLogix Controllers (1769-L32E, -L35E) - EtherNet/IP

Indicator:	Color:	Description:
EtherNet/IP MS	off	There is no activity.
	flashing green	The EtherNet/IP port does not have an IP address and is operating in BOOTP mode.
	solid green	EtherNet/IP communications are active.
	solid red	One of the following occurred: • The controller is holding the EtherNet/IP port in reset or the controller is faulted. • The EtherNet/IP port is performing it's power-up self-test. • An unrecoverable fault has occurred. Cycle power to the controller.
	flashing red	Firmware is being updated.

Indicator:	Color:	Description:
EtherNet/IP NS	off	There is no activity. The EtherNet/IP port does not have an IP address and is operating in BOOTP mode.
	flashing green	The EtherNet/IP port has an IP address but there are no CIP connections established.
	solid green	The EtherNet/IP port has an IP address and CIP connections are established.
	solid red	The assigned IP address is already in use.
	flashing red/green	The EtherNet/IP port is performing its power-up self-test.
EtherNet/IP LNK	off	The EtherNet/IP port is not properly connected to the EtherNet/IP network. Make sure that all Ethernet cables are connected and that the Ethernet switch has power.
	flashing green	One of the following occurred: • The EtherNet/IP port is performing it's power-up self-test. • The EtherNet/IP port is communicating on the network.
	solid green	The EtherNet/IP port is properly connected to the EtherNet/IP network.

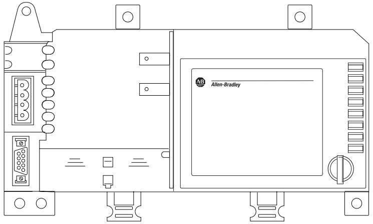
CompactLogix Controllers (1769-L32C, -L35CR) - ControlNet

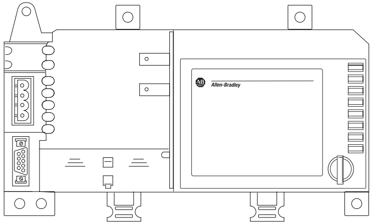
Indicator:	Color:	Description:
ControlNet MS	off	the controller has no power.
		the controller is faulted.
	steady red	a major fault has occurred on the controller.
	flashing red	a minor fault has occurred because a firmware update is in progress.
		a node address switch change occurred. The controller’s node address switches may have been changed since power-up.
		the controller uses invalid firmware.
		the controller’s node address duplicates that of another device.
	steady green	connections are established.
	flashing green	no connections are established.
	flashing red/green	the controller is performing self-diagnostics.

Indicator:	If both channel indicators are:	Description:
ControlNet[1] A ▭ B ▭	off	a channel is disabled.
	steady green	normal operation is occurring.
	flashing green/off	temporary network errors have occurred.
		the node is not configured to go online.
	flashing red/off	media fault has occurred.
		no other nodes present on the network.
	flashing red/green	the network is configured incorrectly.
	If either channel indicator is:	
	off	you should check the MS indicators.
	steady red	the controller is faulted.
	alternating red/green	the controller is performing a self-test.
	alternating red/off	the node is configured incorrectly.

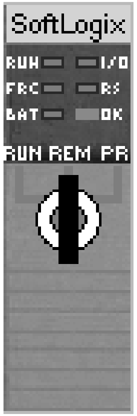
[1] Channel B is only labelled on the 1769-L35CR controller. The 1769-L32C controller only has channel A but uses the second indicator in some LED patterns as described in this table.

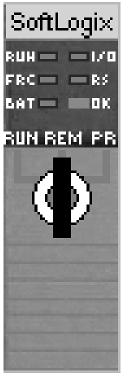
FlexLogix Controllers (1794-L34)

Front Panel:	Indicator:	Color:	Description:
	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	OK	off	No power is applied.
		flashing red	If the controller is:Then: a new controllerthe controller requires a firmware update not a new controllerA major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	The controller is OK.
		flashing green	The controller is storing or loading a project to or from nonvolatile memory.
	BATTERY	off	The battery supports memory.
		red	Either the battery is: • not installed. • 95% discharged and should be replaced.

Front Panel:	**Indicator:**	**Color:**	**Description:**
	I/O	off	Either: • The controller project is not downloaded (the condition after power up). • No I/O **or** communications are configured.
		solid green	The controller is communicating to **all** devices.
		flashing green	One or more devices are not responding.
	LOCAL and LOCAL2	off	The rail is inhibited.
		solid green	The controller is communicating to **all** devices on that rail.
		flashing green	One or more devices on that rail not responding.
		flashing red	No modules exist on that rail.
	RS232	off	There is no activity.
		solid green	Data is being received or transmitted.
	FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		solid amber	I/O forces are active (enabled). I/O force values may or may not exist.
		flashing amber	One or more input or output addresses have been forced to an On or Off state, but the forces have not been enabled.

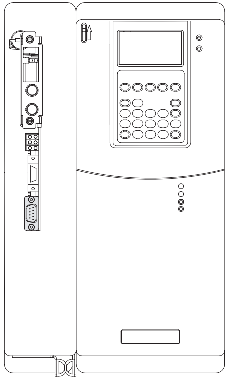
SoftLogix5800 Controllers (1789-L10, -L30, -L60)

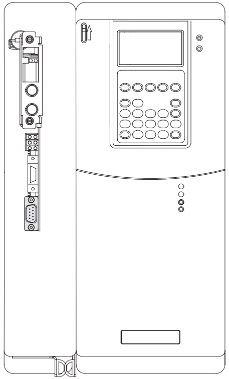
Front Panel:	Indicator:	Color:	Description:
 SoftLogix RUN I/O FRC RS BAT OK RUN REM PR	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
		solid green	The controller is communicating with all the devices in its I/O configuration.
		flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
		flashing red	A virtual chassis error was detected. Contact your Rockwell Automation representative or local distributor.
	FRC	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		flashing green	At least one tag contains an I/O force value. I/O force values are inactive (disabled).
		solid green	I/O forces are active (enabled). I/O force values may or may not exist.
	RS232[1]	off	No COM port was selected.
		solid green	The selected COM port was successfully assigned to channel 0 of the controller.
		solid red	There is a COM port conflict or you selected an invalid COM port number.

Front Panel:	Indicator:	Color:	Description:
	BAT[1]	off	Normal operation.
		flashing amber	The controller is in power-up mode.
		solid red	Persistent storage for the controller has failed.
	OK	flashing red	If the controller is:Then: a new controllerthe controller requires a firmware update not a new controllerA major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	The controller is OK.

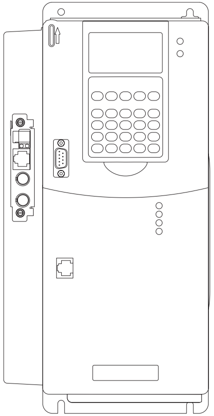
[1] Note that these LEDs function slightly different than the same LEDs on a ControlLogix controller.

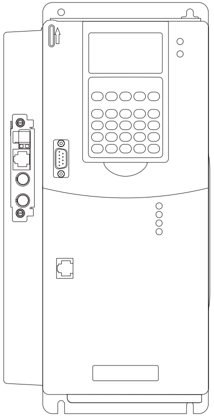
PowerFlex 700S with DriveLogix5720

Front Panel:	Indicator:	Color:	Description:
	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		flashing amber	At least one tag contains an I/O force value. I/O force values are inactive (disabled).
		solid amber	I/O forces are active (enabled). I/O force values may or may not exist.
	BAT	off	The battery supports memory.
		solid red	Either the battery is: • not installed. • 95% discharged and should be replaced.
	I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
		solid green	The controller is communicating with all the devices in its I/O configuration.
		flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
		flashing red	No required I/O connections can be made, controller is in Run mode.

Front Panel:	**Indicator:**	**Color:**	**Description:**
	RS232	off	No COM port was selected.
		solid green	The selected COM port was successfully assigned to channel 0 of the controller.
		solid red	There is a COM port conflict or you selected an invalid COM port number.
	OK	flashing red	If the controller is: Then: a new controller: the controller requires a firmware update not a new controller: A major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	The controller is OK.
		flashing green	The controller is storing or loading a project to or from nonvolatile memory.

PowerFlex 700S with DriveLogix5730

Front Panel:	**Indicator:**	**Color:**	**Description:**
	RUN	off	The controller is in Program or Test mode.
		solid green	The controller is in Run mode.
	FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
		solid amber	I/O forces are active (enabled). I/O force values may or may not exist.
		flashing amber	One or more input or output addresses have been forced to an On or Off state, but the forces have not been enabled.
	BAT	off	The battery supports memory.
		solid red	Either the battery is: • not installed. • 95% discharged and should be replaced.
	I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
		solid green	The controller is communicating with all the devices in its I/O configuration.
		flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
		flashing red	The controller is not communicating to any devices. The controller is faulted.

Front Panel:	Indicator:	Color:	Description:
	COM	off	No RS-232 activity.
		flashing green	RS-232 activity.
	OK	off	No power is applied.
		flashing red	If the controller is: Then: a new controller the controller requires a firmware update not a new controller A major fault occurred. To clear the fault, either: - Turn the keyswitch from PROG to RUN to PROG - Go online with RSLogix 5000 software
		solid red	The controller detected a non-recoverable fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains solid red, contact your Rockwell Automation representative or local distributor.
		solid green	Controller is OK.
		flashing green	The controller is storing or loading a project to or from nonvolatile memory.

Controller Comparison

Common Characteristics	1756 ControlLogix	1769 CompactLogix	1789 SoftLogix	1794 FlexLogix	PowerFlex 700S with DriveLogix
controller tasks • continuous • periodic • event	• 32 tasks (only 1 continuous) • event tasks: supports all event triggers	• 1769-L35E, -L35CR: 8 tasks • 1769-L32E, -L32C: 6 tasks • 1769-L31: 4 tasks • only 1 continuous • event tasks: supports consumed tag trigger and EVENT instruction	• 32 tasks (only 1 continuous) • event tasks: supports all event triggers, plus outbound and Windows events	• 8 tasks (only 1 continuous) • event tasks: supports consumed tag trigger and EVENT instruction	• 8 tasks (only 1 continuous) • event tasks: supports axis and motion event triggers
user memory	1756-L55M12 750 Kbytes 1756-L55M13 1.5 Mbytes 1756-L55M14 3.5 Mbytes 1756-L55M16 7.5 Mbytes 1756-L55M22 750 Kbytes 1756-L55M23 1.5 Mbytes 1756-L55M24 3.5 Mbytes 1756-L61 2 Mbytes 1756-L62 4 Mbytes 1756-L63 8 Mbytes	1769-L31 512 Kbytes 1769-L32E, -L32C 750 Kbytes 1769-L35E, -L35CR 1.5 Mbytes	1789-L10 2 Mbytes 3 slots, no motion 1789-L30 64 Mbytes 5 slots 1789-L60 64 Mbytes 16 slots	1794-L34 512 Kbytes	5720 256 Kbytes 768 Kbytes with expansion memory 5730 1.5Mbytes
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-L6x CompactFlash	CompactFlash	none	yes	5720 yes (expansion memory) 5730 CompactFlash

Common Characteristics	1756 ControlLogix	1769 CompactLogix	1789 SoftLogix	1794 FlexLogix	PowerFlex 700S with DriveLogix
built-in communication ports	1 port RS-232 serial (DF1 or ASCII)	• 1769-L31: 2 RS-232 serial ports (one DF1 only, other DF1 or ASCII) • 1769-L32C, -L35CR: 1 ControlNet port and 1 RS-232 serial port (DF1 or ASCII) • 1769-L32E, -L35E: 1 EtherNet/IP port and 1 RS-232 serial port (DF1 or ASCII)	depends on personal computer	• 1 RS-232 serial port (DF1 or ASCII) • 2 slots for 1788 communication cards	5720 • 1 RS-232 serial port (DF1 or ASCII) • 1 slot for 1788 communication cards 5730 • 1 RS-232 serial port (DF1 or ASCII) • 1 slot for 1788 (option) • 1 embedded ethernet (option)
communication options (these options have specific products and profiles for their platform - other options are available via 3rd party products and generic profiles)	EtherNet/IP ControlNet DeviceNet Data Highway Plus Universal Remote I/O serial Modbus via ladder routine DH-485 SynchLink	EtherNet/IP ControlNet DeviceNet serial Modbus via ladder routine DH-485	EtherNet/IP ControlNet DeviceNet serial	EtherNet/IP ControlNet DeviceNet serial Modbus via ladder routine DH-485	EtherNet/IP ControlNet DeviceNet serial Modbus via ladder routine DH-485
connections	64 over ControlNet (48 recommended) 128 over EtherNet/IP	32 over ControlNet 32 over EtherNet/IP	64 over ControlNet (48 recommended) EtherNet/IP limited by type and number of cards	32 over ControlNet 32 over EtherNet/IP	32 over ControlNet 32 over EtherNet/IP
controller redundancy	full redundancy support	not applicable	not applicable	controller hot backup via DeviceNet	not applicable

Common Characteristics	1756 ControlLogix	1769 CompactLogix	1789 SoftLogix	1794 FlexLogix	PowerFlex 700S with DriveLogix
native I/O	1756 ControlLogix I/O	1769 Compact I/O	supported via 3rd party PCI bus I/O cards	1794 FLEX I/O 1797 FLEX Ex I/O	5720 • 1794 FLEX I/O • 1797 FLEX Ex I/O 5730 • 1769 Compact I/O
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
integrated motion	SERCOS interface analog interface with options: • quadrature encoder input • LDT input • SSI input	not applicable	SERCOS interface analog interface with options: • quadrature encoder input • LDT input • SSI input	not applicable	1 full servo 1 feedback axis
mounting and/or installation options	1756 chassis	panel mount DIN rail	none	panel mount DIN rail	embedded
programming languages	• relay ladder • structured text • function block • sequential function chart	• relay ladder • structured text • function block • sequential function chart	• relay ladder • structured text • function block • sequential function chart • external routines (Windows DLLs developed using C/C++)	• relay ladder • structured text • function block • sequential function chart	• relay ladder • structured text • function block • sequential function chart

Select the Operating Mode of the Controller

Use this table to determine the operating mode of the controller:

If you want to:	Select one of these modes:				
	Run	Remote			Program
		Run	Test	Program	
turn outputs to the state commanded by the logic of the project	X	X			
turn outputs to their configured state for Program mode			X	X	X
execute (scan) tasks	X	X	X		
change the mode of the controller through software		X	X	X	
download a project		X	X	X	X
schedule a ControlNet network				X	X
while online, edit the project		X	X	X	X
send messages	X	X	X		
send and receive data in response to a message from another controller	X	X	X	X	X
produce and consume tags	X	X	X	X	X

Turn the key on the front panel of the controller to select the mode.

Non-Volatile Memory

These controllers have nonvolatile memory for project storage.

Controller Type:	Catalog Number:	Firmware Revision:
CompactLogix5332E	1769-L32E[1]	13.x or later
CompactLogix5335E	1769-L35E[1]	12.x or later
CompactLogix5331	1769-L31[1]	13.x or later
CompactLogix5332C	1769-L32C[1]	13.x or later
CompactLogix5335CR	1769-L35CR[1]	13.x or later
ControlLogix5555	1756-L55M22	10.x or later
	1756-L55M23	8.x or later
	1756-L55M24	8.x or later
ControlLogix5560M03SE	1756-L60M03SE[1]	13.x or later
ControlLogix5561 and ControlLogix5562	1756-L61, -L62[1]	12.x or later for series A 13.x or later for series B
ControlLogix5563	1756-L63[1]	11.x or later for series A 13.x or later for series B
DriveLogix5720	various	10.x or later
DriveLogix5730	various[1]	13.x or later
FlexLogix5434 Series B	1794-L34/B	11.x or later

[1] Requires a 1784-CF64 Industrial CompactFlash memory card.

On the controller properties, you select to store/load a project to/from non-volatile memory:

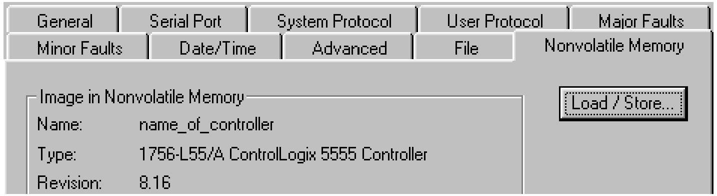


Image in Nonvolatile Memory

Name: name_of_controller

Type: 1756-L55/A ControlLogix 5555 Controller

Revision: 8.16

Project that is currently in the nonvolatile memory of the controller (if any project is there).

Project that is currently in the user memory (RAM) of the controller.

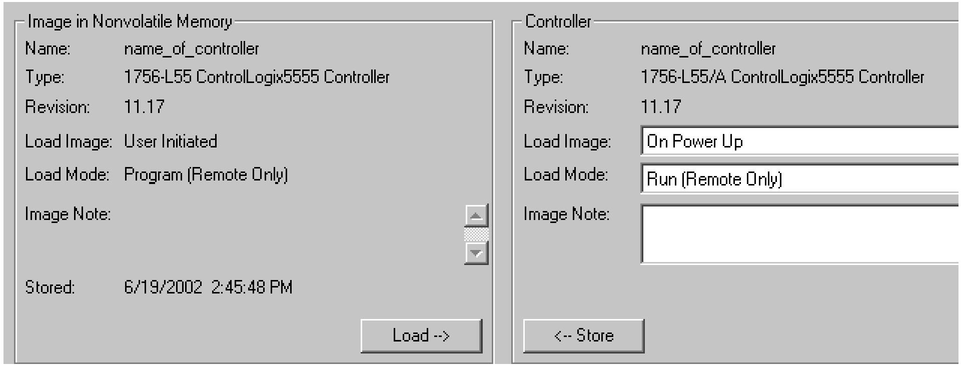


Image in Nonvolatile Memory

Name: name_of_controller

Type: 1756-L55 ControlLogix5555 Controller

Revision: 11.17

Load Image: User Initiated

Load Mode: Program (Remote Only)

Image Note:

Stored: 6/19/2002 2:45:48 PM

Controller

Name: name_of_controller

Type: 1756-L55/A ControlLogix5555 Controller

Revision: 11.17

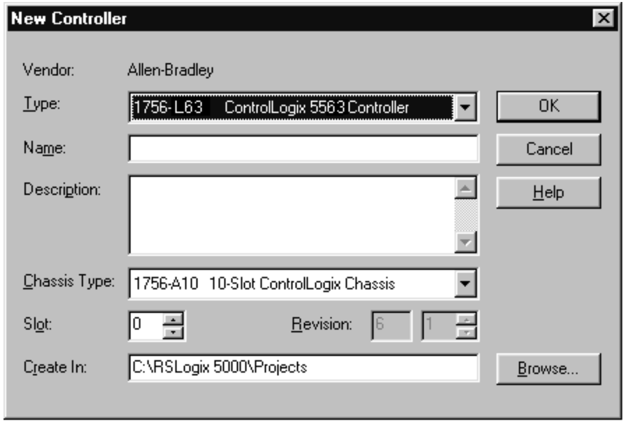
Load Image: On Power Up

Load Mode: Run (Remote Only)

Image Note:

Create a Project

From RSLogix 5000 software, select File →New.



Vendor: Allen-Bradley

Type: 1756-L63 ControlLogix 5563 Controller

Name:

Description:

Chassis Type: 1756-A10 10-Slot ControlLogix Chassis

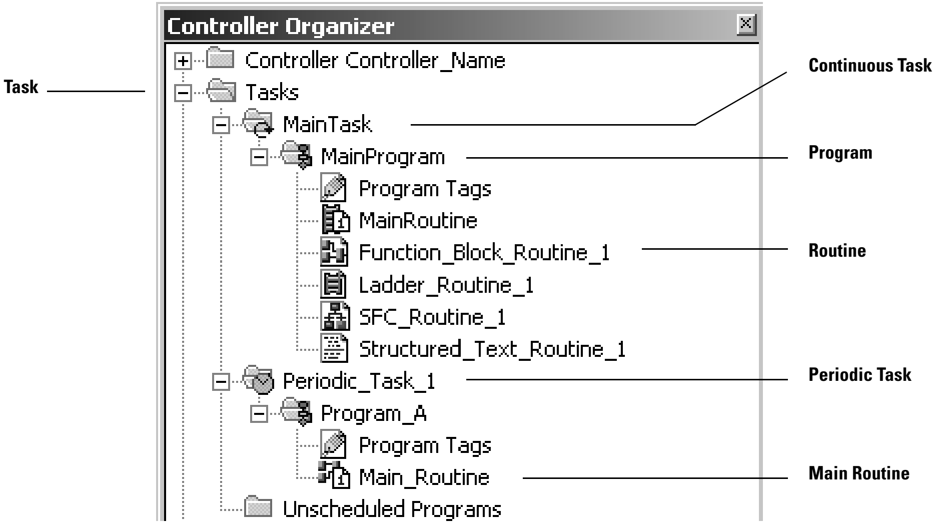
Slot: 0

Revision: 6 1

Create In: C:\RSLogix 5000\Projects

Controller Organizer

The programming software uses the Controller Organizer to define a project.



Controller Organizer

- Controller Controller_Name
- Tasks
  - MainTask
    - MainProgram
      - Program Tags
      - MainRoutine
      - Function_Block_Routine_1
      - Ladder_Routine_1
      - SFC_Routine_1
      - Structured_Text_Routine_1
  - Periodic_Task_1
    - Program_A
      - Program Tags
      - Main_Routine
  - Unscheduled Programs

Controller Tasks

A task provides scheduling and priority information for a set of one or more programs that execute based on specific criteria. Once a task is triggered (activated), all the programs assigned (scheduled) to the task execute in the order in which they are displayed in the controller organizer.

Task:	Definition:
continuous task	The continuous task runs in the background. Any CPU time not allocated to other operations (such as motion, communications, and periodic or event tasks) is used to execute the programs within 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.
periodic task	A periodic task performs a function at a specific rate. • Whenever the time for the periodic task expires, the periodic task interrupts any lower priority tasks, executes one time, and then returns control to where the previous task left off. • You can configure the time period from 1 ms to 2000 s. The default is 10 ms. The performance of a periodic tasks depends on the type of Logix controller and the logic in the task. You assign a priority level (1 is the highest, 15 is the lowest) to each periodic task: • The highest priority task interrupts all lower priority tasks. • A higher priority task can interrupt a lower priority task multiple times. • Tasks at the same priority execute on a time-slice basis at 1 ms intervals.
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, and then returns control to where the previous task left off. Available triggers are Module Input Data State Change, Consumed Tag, Axis Registration 1 or 2, Axis Watch, Motion Group Execution, EVENT Instruction.

The number of tasks supported depends on the controller:

Controller:	Number of Tasks Supported:
ControlLogix	32 tasks, one of which can be continuous There are 15 configurable priority levels for periodic tasks(1-15), with 1 being the highest priority and 15 being the lowest priority.
CompactLogix and PowerFlex 700S with DriveLogix5730	1769-L35E, -L35CR: 8 tasks, one of which can be continuous 1769-L32E, -L32C: 6 tasks, one of which can be continuous 1769-L31, -L30, -L20: 4 tasks, one of which can be continuous There are 15 configurable priority levels for periodic tasks(1-15), with 1 being the highest priority and 15 being the lowest priority. The CompactLogix controller uses a dedicated periodic task at priority 7 to process I/O data. This periodic task executes at the fastest RPI you have scheduled for the system. Its total execution time is as long as it takes to scan the configured I/O modules.
FlexLogix and PowerFlex 700S with DriveLogix5720	8 tasks, one of which can be continuous There are 15 configurable priority levels for periodic tasks(1-15), with 1 being the highest priority and 15 being the lowest priority. The controller uses a dedicated periodic task at priority 7 to process I/O data. This periodic task executes at the fastest RPI you have scheduled for the system. Its total execution time is as long as it takes to scan the configured I/O modules.
SoftLogix5800	32 tasks, one of which can be continuous There are 3 configurable priority levels for periodic tasks (1-3), with 1 being the highest priority and 3 being the lowest priority.

A task can have as many as 32 separate programs, each with its own executable 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 grouped. Programs can only appear once in the Controller Organizer and cannot be shared by multiple tasks.

When a task is triggered, the scheduled programs within the task execute to completion from first to last. Each program contains program tags, a main routine, other routines, and an optional fault routine. When a program executes, its main routine executes first. Use the main routine to call (execute) other routines (subroutines). To call another routine within the program, use a Jump to Subroutine (JSR) instruction.

Event task details

Not all Logix controllers support all event task triggers:

If you have this controller:	Then you can use these event task triggers:					
	Module Input Data State Change	Consumed Tag	Axis Registration 1 or 2	Axis Watch	Motion Group Execution	EVENT instruction
CompactLogix		X				X
FlexLogix		X				X
ControlLogix	X	X	X	X	X	X
DriveLogix5720			X	X	X	X
DriveLogix5730		X	X	X	X	X
SoftLogix5800	X[1]	X[2]	X	X	X	X

[1] Requires a 1756 I/O module or a virtual backplane.

[2] A SoftLogix5800 controller produces and consumes tags only over a ControlNet network.

To use an input module to trigger an event task, the module must support event task triggering. If the module is in a remote location, the associated communication modules must also support event triggering. These modules can trigger an event task.

Category	Module
1756 Discrete	1756-IA8D
	1756-IA16, -IA16I
	1756-IA32
	1756-IB16, -IB16D, -IB16I
	1756-IB16ISOE
	1756-IB32/A, -IB32/B
	1756-IC16
	1756-IG16
	1756-IH16I, -IH16ISOE
	1756-IM16I
	1756-IN16
	1756-IV16/A
	1756-IV32/A
1756 Analog	1756-IF16
	1756-IF4FXOF2F/A
	1756-IF6CIS
	1756-IF6I
	1756-IF8
	1756-IR6I
	1756-IT6I
	1756-IT6I2
1756 Specialty	1756-CFM/A
	1756-HSC
	1756-PLS/B
1756 Communication	1756-CNB/A, -CNB/B, -CNB/D
	1756-CNBR/A, -CNBR/B, -CNBR/D
	1756-DNB
	1756-ENBT/A
	1756-SYNCH/A
1756 Generic	1756-MODULE
SoftDNB	1784-PCIDS/A
1789 Generic	1789-MODULE

Controller Tags

The most common data types are.

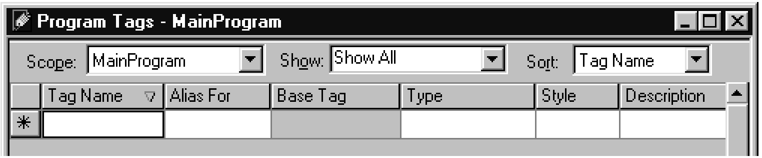
For:	Select:	For:	Select:
analog device in floating-point mode	REAL	digital I/O point	BOOL
analog device in integer mode (for very fast sample rates)	INT	floating-point number	REAL
ASCII characters	string	integer (whole number)	DINT
bit	BOOL	sequencer	CONTROL
counter	COUNTER	timer	TIMER

To organize your data:

For a:	Use a:
group of common attributes that are used by more than one machine	user-defined data type
group of data with the same data type	array
single value	tag of a single element
I/O device	

Create a tag

From the *Logic* menu, select *Edit Tags*.



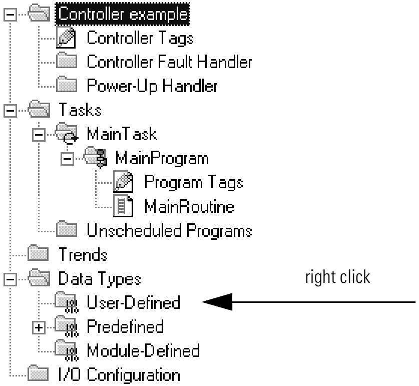
Scope: MainProgram
Show: Show All
Sort: Tag Name

	Tag Name	Alias For	Base Tag	Type	Style	Description
✳						

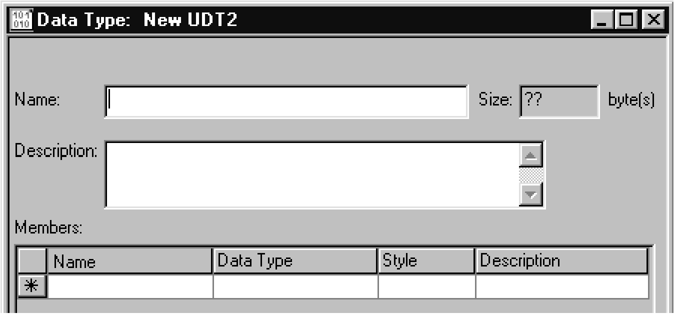
You can configure tags to communicate directly with other controllers:

To:	Use a:
send data over the backplane and ControlNet network at a specified interval	produced tag
receive data from another controller over the backplane or ControlNet network at a specified interval	consumed tag

Create a user-defined data type



right click



Name:

Size: ?? byte(s)

Description:

Members:

	Name	Data Type	Style	Description
✳				

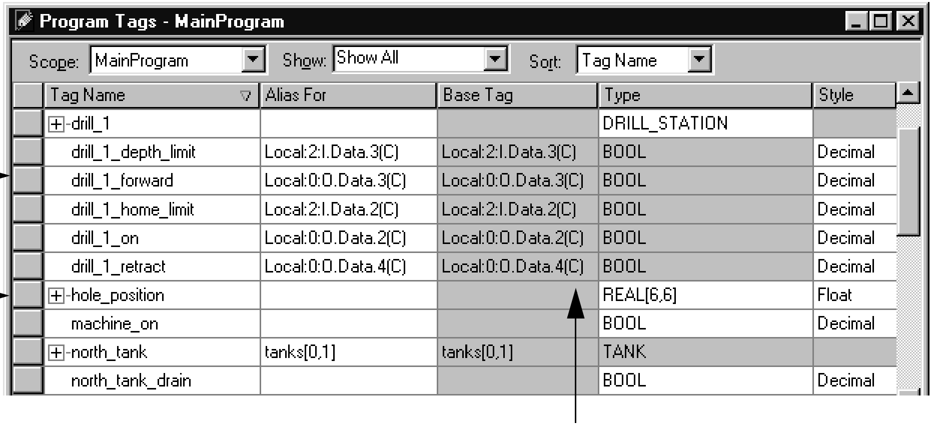
Aliases

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

- Both tags share the same value (s).
- When the value (s) of one of the tags changes, the other tag reflects the change as well.

drill_1_depth_limit is an alias for *Local:2:I.Data.3* (a digital input point). When the input turns on, the alias tag also turns on.

drill_1_on is an alias for *Local:0:O.Data.2* (a digital output point). When the alias tag turns on, the output tag also turns on.



Scope: MainProgram Show: Show All Sort: Tag Name

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

The *(C)* indicates that the tag is at the controller scope.

Choose a Programming Language

In general, if the function or group of functions represent:	Then use this language:
continuous or parallel execution of multiple operations (not sequenced)	ladder logic
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	
continuous process and drive control	function block diagram
loop control	
calculations in circuit flow	
high-level management of multiple operations	sequential function chart (SFC)
repetitive sequences of operations	
batch process	
motion control using structured text	
state machine operations	
continued	

In general, if the function or group of functions represent:	Then use this language:
complex mathematical operations	structured text
specialized array or table loop processing	
ASCII string handling or protocol processing	

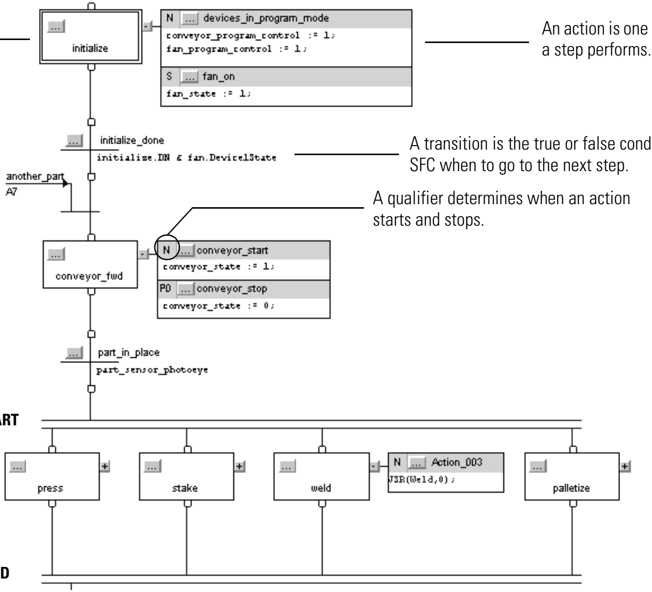
Notes:

Sequential Function Charts

Chapter **2**

A **sequential function chart** (SFC) is similar to a flowchart. It uses steps and transitions to perform specific operations or actions.

A step represents a major function of your process. It contains the actions that occur at a particular time, phase, or station.



```
conveyor_program_control := 1;
fan_program_control := 1;
```

```
fan_state := 1;
```

```
initialize.DN & fan.DeviceState
```

```
conveyor_state := 1;
```

```
conveyor_state := 0;
```

```
part_sensor_photoeye
```

```
JSR(Weld,0);
```

An action is one of the functions that a step performs.

A transition is the true or false condition that tells the SFC when to go to the next step.

A qualifier determines when an action starts and stops.

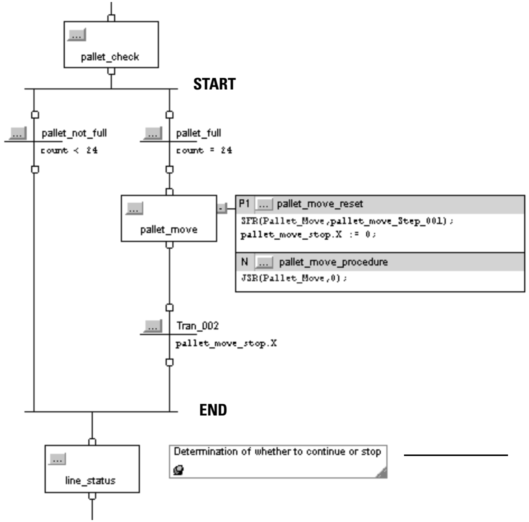
START

A simultaneous branch executes more than 1 step at the same time.

END

continued

Example SFC continued



```
count < 24
```

```
count = 24
```

```
SFR(Pallet_Move,pallet_move_Step_001);
pallet_move_stop.X := 0;
```

```
JSR(Pallet_Move,0);
```

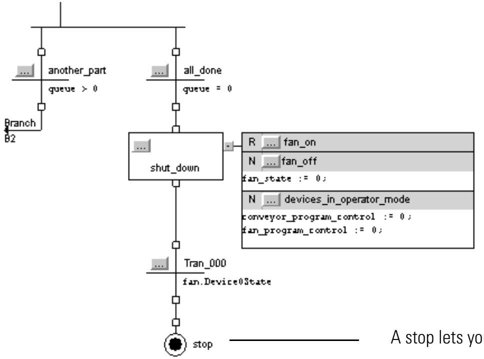
```
pallet_move_stop.X
```

A selection branch chooses between different execution paths.

A text box lets you add descriptive text or notes to your SFC.

continued

Example SFC continued



```
queue > 0
```

```
queue = 0
```

```
fan_state := 0;
```

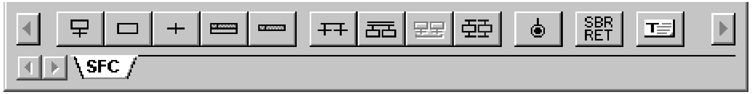
```
conveyor_program_control := 0;
fan_program_control := 0;
```

```
fan.Device0State
```

A wire connects one element to another element anywhere on the chart.

A stop lets you stop and wait for a command to restart.

Editing an SFC



Button	SFC Element Created	Description
⊤	step and transition pair	Add a step and transition pair. See the descriptions for step and transition below.
▭	step	Add a step. A step represents a major function of a process. It contains the events that occur at a particular time, phase, or station.
+	transition	Add a transition. A transition is the true or false condition or conditions that determine when to go to the next step.
▤	action	Add an action or a boolean action to the selected step. Click the step and then press this button. An action represents a functional division of a step. Several actions make up a step. Each action performs a specific function, such as controlling a motor, opening a valve, or placing a group of devices in a specific mode. Each action includes a qualifier. When a step is active (executing) the qualifier determines when the action starts and stops.
▬	boolean action	
┬┬	selection branch diverge	Starts a selection branch. Use the new path button to add paths to the branch structure.

Button	SFC Element Created	Description
	simultaneous branch diverge	Starts a simultaneous branch. Use the new path button to add paths to the branch structure.
	converge SFC elements	Ends the current branch. Select the last step of each path in the branch and then press this button. A simultaneous branch ends with a double horizontal line and no transition. A selection branch ends with a transition for each path and a single horizontal line.
	extend branch	Add a path to a branch. Click the first step of the path that is to the left of where you want to add the new path and then press the button.
	stop	End a path in a branch without connecting to other SFC elements.
SBR RET	subroutine/return	Add a subroutine call.
T	text box	Create a text box. Once the text box appears, you can click and drag the text box to the location you want. Double-click the text box to add text.

SFC_STEP Structure

Member	Data type	Details
T	DINT	When a step becomes active, the Timer (T) value resets and then starts to count up in milliseconds. The timer continues to count up until the step goes inactive, regardless of the Preset (PRE) value.
PRE	DINT	Enter the time in the Preset (PRE) member. When the Timer (T) reaches the Preset value, the Done (DN) bit turns on and stays on until the step becomes active again. As an option, enter a numeric expression that calculates the time at runtime.
DN	BOOL	When the Timer (T) reaches the Preset (PRE) value, the Done (DN) bit turns on and stays on until the step becomes active again.
LimitLow	DINT	Enter the time in the LimitLow member (milliseconds). • If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on. • The AlarmLow bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.
AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.
AlarmLow	BOOL	If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on. • The bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.
LimitHigh	DINT	Enter the time in the LimitHigh member (milliseconds). • If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on. • The AlarmHigh bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.
AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.

Member	Data type	Details
AlarmHigh	BOOL	If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on. • The bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.
X	BOOL	The X bit is on the entire time the step is active (executing).
FS	BOOL	The FS bit is on during the first scan of the step.
SA	BOOL	The SA bit is on when the step is active except during the first and last scan of the step.
LS	BOOL	The LS bit is on during the last scan of the step. Use this bit only if you do the following: On the *Controller Properties* dialog box, *SFC Execution* tab, set the *Last Scan of Active Step* to *Don't Scan* or *Programmatic reset*.
Reset	BOOL	An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies. • The Reset bit indicates to which step or stop the SFC will go to begin executing again. • Once the SFC executes, the Reset bit clears.
TMax	DINT	Use this for diagnostic purposes. The controller clears this value only when you choose the *Restart Position* of *Restart at initial step* and the controller changes modes or experiences a power cycle.
OV	BOOL	Use this for diagnostic purposes.
Count	DINT	This is *not* a count of scans of the step. • The count increments each time the step becomes active. • It increments again only after the step goes inactive and then active again. • The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.

Member	Data type	Details	
Status	DINT	**For this member:**	**Use this bit:**
		Reset	22
		AlarmHigh	23
		AlarmLow	24
		AlarmEn	25
		OV	26
		DN	27
		LS	28
		SA	29
		FS	30
		X	31

SFC_ACTION Structure

Member	**Data type**	**Details**	
Q	BOOL	The status of the Q bit depends on whether the action is a boolean action or non-boolean action:	
		If the action is:	**Then the Q bit is:**
		boolean	on (1) the entire time the action is active, including the last scan of the action
		non-boolean	on (1) while the action is active but off (0) at the last scan of the action
		To use a bit to determine when an action is active, use the Q bit.	
A	BOOL	The A bit is on the entire time the action is active.	
T	DINT	When an action becomes active, the Timer (T) value resets and then starts to count up in milliseconds. The timer continues to count up until the action goes inactive, regardless of the Preset (PRE) value.	
PRE	DINT	Enter the time limit or delay in the Preset (PRE) member. The action starts or stops when the Timer (T) reaches the Preset value.	
Count	DINT	This is *not* a count of scans of the action. • The count increments each time the action becomes active. • It increments again only after the action goes inactive and then active again. • The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.	
Status	DINT	**For this member:**	**Use this bit:**
		Q	30
		A	31

Action Qualifiers

If you want the action to:	And:	Assign this qualifier:	Which means:
start when the step is activated	stop when the step is deactivated	N	Non-Stored (default)
	execute only once	P1	Pulse (Rising Edge)
	stop before the step is deactivated or when the step is deactivated	L	Time Limited
	stay active until a *Reset* action turns off this action	S	Stored
	stay active until a *Reset* action turns off this action or a specific time expires, even if the step is deactivated	SL	Stored and Time Limited
start a specific time after the step is activated *and* the step is still active	stop when the step is deactivated	D	Time Delayed
	stay active until a *Reset* action turns off this action	DS	Delayed and Stored
start a specific time after the step is activated, even if the step is deactivated before this time	stay active until a *Reset* action turns off this action	SD	Stored and Time Delayed
execute once when the step is activated	execute once when the step is deactivated	P	Pulse
start when the step is deactivated	execute only once	P0	Pulse (Falling Edge)
turn off (reset) a stored action: • S Stored • SL Stored and Time Limited • DS Delayed and Stored • SD Stored and Time Delayed	→	R	Reset

SFC_STOP Structure

<table>
<tr><th>Member:</th><th>Data type:</th><th>Details:</th></tr>
<tr><td>X</td><td>BOOL</td><td>• When the SFC reaches the stop, the X bit turns on.
• The X bit clears if you configure the SFCs to restart at the initial step and the controller changes from program to run mode.
• In a nested SFC, the X bit also clears if you configure the SFCs for automatic reset and the SFC leaves the step that calls the nested SFC.</td></tr>
<tr><td>Reset</td><td>BOOL</td><td>An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies.
• The Reset bit indicates to which step or stop the SFC will go to begin executing again.
• Once the SFC executes, the Reset bit clears.</td></tr>
<tr><td>Count</td><td>DINT</td><td>This is <i>not</i> a count of scans of the stop.
• The count increments each time the stop becomes active.
• It increments again only after the stop goes inactive and then active again.
• The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.</td></tr>
<tr><td>Status</td><td>DINT</td><td><table>
<tr><th>For this member:</th><th>Use this bit:</th></tr>
<tr><td>Reset</td><td>22</td></tr>
<tr><td>X</td><td>31</td></tr>
</table></td></tr>
</table>

How Do You Want to Use the Action?

There are two types of actions:

If you want to:	Then use a:
execute structured text directly in the SFC	non-boolean action
call a subroutine	
use the automatic reset option to reset data upon leaving a step	
only set a bit and program other logic to monitor the bit to determine when to execute.	boolean action

Use a non-boolean action

A non-boolean action contains the logic for the action. It uses structured text to execute assignments and instructions or call a subroutine. With non-boolean actions, you also have the option to postscan (automatically reset) the assignments and instructions before leaving a step:

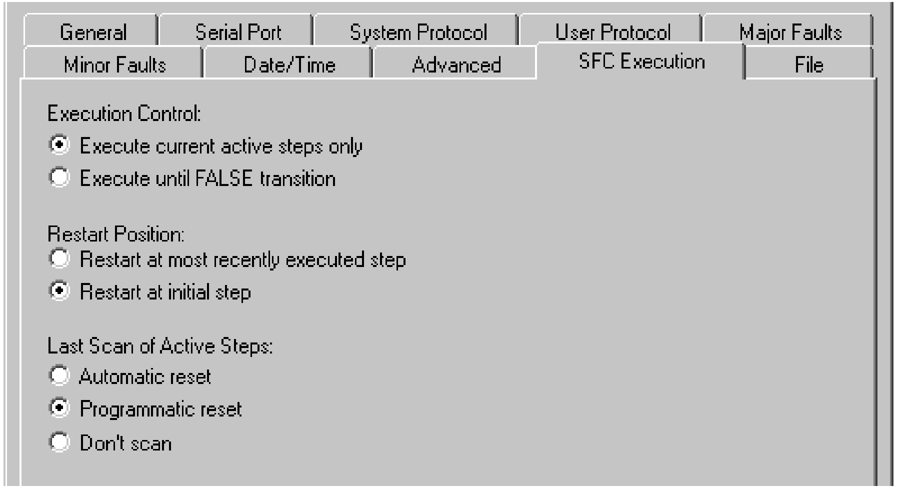
- During postscan the controller executes the assignments and instructions as if all conditions are false.
- The controller postscans both embedded structured text and any subroutine that the action calls.

Use a boolean action

A boolean action contains no logic for the action. It simply sets a bit in its tag (SFC_ACTION structure). To do the action, other logic must monitor the bit and execute when the bit is on. With boolean actions, you have to manually reset the assignments and instructions that are associated with the action. Since there is no link between the action and the logic that performs the action, the automatic reset option does not effect boolean actions. You can reuse a boolean action multiple times within the same SFC.

Configure the Execution of an SFC

From Controller Properties:



Execution Control:

● Execute current active steps only

○ Execute until FALSE transition

Restart Position:

○ Restart at most recently executed step

● Restart at initial step

Last Scan of Active Steps:

○ Automatic reset

● Programmatic reset

○ Don't scan

Notes:

Structured Text

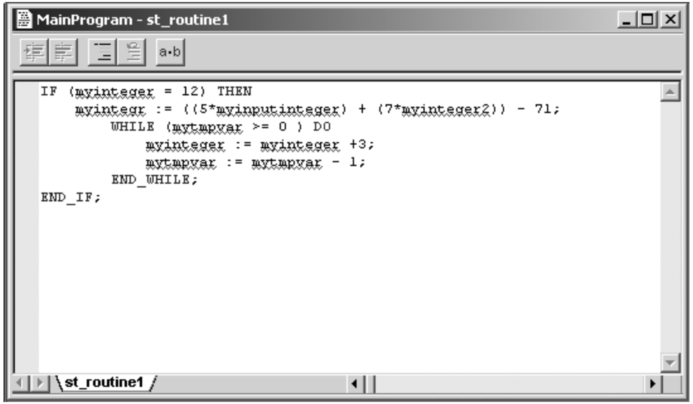
Chapter **3**

Structured Text Syntax

Structured text is a textual programming language that uses statements to define what to execute.

- Structured text is not case sensitive.
- Use tabs and carriage returns (separate lines) to make your structured text easier to read. They have no effect on the execution of the structured text.

This is an example of a structured text routine.



```
IF (myinteger = 12) THEN
    myintegr := ((5*myinputinteger) + (7*myinteger2)) - 71;
        WHILE (mytmpvar >= 0 ) DO
            myinteger := myinteger +3;
            mytmpvar := mytmpvar - 1;
        END_WHILE;
END_IF;
```

Structured text can contain these components:

Term:	**Definition:**		**Examples:**
assignment (see page 3-4)	Use an assignment statement to assign values to tags. The := operator is the assignment operator. Terminate the assignment with a semi colon “;”.		*tag* := *expression*;
expression (see page 3-6)	An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression) or to a true or false state (BOOL expression). An expression contains:		
	tags	A named area of the memory where data is stored (BOOL, SINT,INT,DINT, REAL, string).	*value1*
	immediates	A constant value.	*4*
	operators	A symbol or mnemonic that specifies an operation within an expression.	*tag1* + *tag2* *tag1* >= *value1*
	functions	When executed, a function yields one value. Use parentheses to contain the operand of a function. Even though their syntax is similar, functions differ from instructions in that functions can only be used in expressions. Instructions cannot be used in expressions.	*function*(*tag1*)

instruction
(see page 3-13)
	An instruction is a standalone statement.
	An instruction uses parenthesis to contain its operands.
	Depending on the instruction, there can be zero, one, or multiple operands.
	When executed, an instruction yields one or more values that are part of a data structure.
	Terminate the instruction with a semi colon “;”.
	Instructions cannot be used in expressions. Functions can only be used in expressions.
			instruction();
			instruction(operand);
			instruction(operand1, operand2,operand3);

Term:	**Definition:**	**Examples:**
construct (see page 3-15)	A conditional statement used to trigger structured text code (i.e, other statements). Terminate the construct with a semi colon ";".	IF...THEN CASE FOR...DO WHILE...DO REPEAT...UNTIL EXIT
comment (see page 3-25)	Text that explains or clarifies what a section of structured text does. Use comments to make it easier to interpret the structured text. Comments do not affect the execution of the structured text. Comments can appear anywhere in structured text.	//*comment* (**start of comment . . . end of comment**) /**start of comment . . . end of comment**/

Entering spaces in structured text syntax is optional. Spaces have no effect on the execution of the structured text. For example, both of these statements execute the same:

```
Tag_B:=Tag_A

Tag_B := Tag_A
```

Assignments

Use an assignment to change the value stored within a tag. An assignment has this syntax:

```
tag := expression ;
```

where:

Component:	Description:	
tag	represents the tag that is getting the new value the tag must be a BOOL, SINT, INT, DINT, or REAL	
:=	is the assignment symbol	
expression	represents the new value to assign to the tag	
	If *tag* is this data type:	**Use this type of expression:**
	BOOL	BOOL expression
	SINT DINT INT REAL	numeric expression
;	ends the assignment	

The *tag* retains the assigned value until another assignment changes the value.

Specify a non-retentive assignment

A non-retentive assignment is reset to zero each time the controller:

- enters the RUN mode
- leaves the step of an SFC if you configure the SFC for *Automatic reset.*

A non-retentive assignment has this syntax:

tag [:=] *expression* ;

where:

Component:	Description:	
tag	represents the tag that is getting the new value the tag must be a BOOL, SINT, INT, DINT, or REAL	
[:=]	is the non-retentive assignment symbol	
expression	represents the new value to assign to the tag	
	If *tag* is this data type:	**Use this type of expression:**
	BOOL	BOOL expression
	SINT DINT INT REAL	numeric expression
;	ends the assignment	

Expressions

An expression is a tag name, equation, or comparison. To write an expression, use any of the following:

- tag name that stores the value (variable)
- number that you enter directly into the expression (immediate value)
- functions, such as: ABS, TRUNC
- operators, such as: +, -, <, >, And, Or

BOOL expression: An expression that produces either the BOOL value of 1 (true) or 0 (false).

- A bool expression uses bool tags, relational operators, and logical operators to compare values or check if conditions are true or false. For example, `tag1>65`.
- A simple bool expression can be a single BOOL tag.
- Typically, you use bool expressions to condition the execution of other logic.

Numeric expression: An expression that calculates an integer or floating-point value.

- A numeric expression uses arithmetic operators, arithmetic functions, and bitwise operators. For example, `tag1+5`.
- Often, you nest a numeric expression within a bool expression. For example, `(tag1+5)>65`.

Arithmetic operators

Arithmetic operators calculate new values.

To:	Use this operator:	Optimal data type:
add	+	DINT, REAL
subtract/negate	-	DINT, REAL
multiply	*	DINT, REAL
exponent (x to the power of y)	**	DINT, REAL
divide	/	DINT, REAL
modulo-divide	MOD	DINT, REAL

Arithmetic functions

Arithmetic functions perform math operations. Specify a constant, a non-boolean tag, or an expression for the function.

For:	**Use this function:**	**Optimal data type:**
absolute value	ABS (*numeric_expression*)	DINT, REAL
arc cosine	ACOS (*numeric_expression*)	REAL
arc sine	ASIN (*numeric_expression*)	REAL
arc tangent	ATAN (*numeric_expression*)	REAL
cosine	COS (*numeric_expression*)	REAL
radians to degrees	DEG (*numeric_expression)*	DINT, REAL
natural log	LN (*numeric_expression*)	REAL
log base 10	LOG (*numeric_expression*)	REAL
degrees to radians	RAD (*numeric_expression*)	DINT, REAL
sine	SIN (*numeric_expression*)	REAL
square root	SQRT (*numeric_expression*)	DINT, REAL
tangent	TAN (*numeric_expression*)	REAL
truncate	TRUNC (*numeric_expression*)	DINT, REAL

Relational operators

Relational operators compare two values or strings to provide a true or false result. The result of a relational operation is a BOOL value:

If the comparison is:	The result is:
true	1
false	0

For this comparison:	Use this operator:	Optimal Data Type:
equal	=	DINT, REAL, string
less than	<	DINT, REAL, string
less than or equal	<=	DINT, REAL, string
greater than	>	DINT, REAL, string
greater than or equal	>=	DINT, REAL, string
not equal	<>	DINT, REAL, string

Logical operators

Logical operators let you check if multiple conditions are true or false. The result of a logical operation is a BOOL value:

If the comparison is:	The result is:
true	1
false	0

For:	Use this operator:	Data Type:
logical AND	&, AND	BOOL
logical OR	OR	BOOL
logical exclusive OR	XOR	BOOL
logical complement	NOT	BOOL

Bitwise operators

Bitwise operators manipulate the bits within a value based on two values.

For:	Use this operator:	Optimal Data Type:
bitwise AND	&, AND	DINT
bitwise OR	OR	DINT
bitwise exclusive OR	XOR	DINT
bitwise complement	NOT	DINT

Determine the order of execution

The operations you write into an expression are performed in a prescribed order, not necessarily from left to right.

- Operations of equal order are performed from left to right.
- If an expression contains multiple operators or functions, group the conditions in parenthesis “( )” to ensure the correct order.

Order:	**Operation:**
1.	( )
2.	function (…)
3.	**
4.	–(negate)
5.	NOT
6.	*, /, MOD
7.	+, - (subtract)
8.	<, <=, >, >=
9.	=, <>
10.	&, AND
11.	XOR
12.	OR

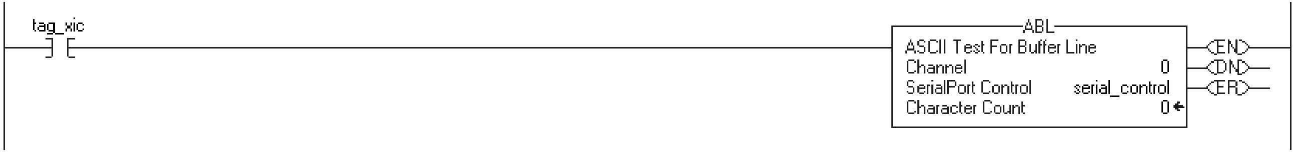
Instructions

Structured text statements can also be instructions. See the Locator Table at the beginning of this manual for a list of the instructions available in structured text. A structured text instruction executes each time it is scanned. A structured text instruction within a construct executes every time the conditions of the construct are true. If the conditions of the construct are false, the statements within the construct are not scanned. There is no rung-condition or state transition that triggers execution.

This differs from function block instructions that use EnableIn to trigger execution. Structured text instructions execute as if EnableIn is always set.

This also differs from relay ladder instructions that use rung-condition-in to trigger execution. Some relay ladder instructions only execute when rung-condition-in toggles from false to true. These are transitional relay ladder instructions. In structured text, instructions will execute each time they are scanned unless you pre-condition the execution of the structured text instruction.

For example, the ABL instruction is a transitional instruction in relay ladder. In this example, the ABL instruction only executes on a scan when *tag_xic* transitions from cleared to set. The ABL instruction does not execute when *tag_xic* stays set or when *tag_xic* is cleared.



In structured text, if you write this example as:

```
IF tag_xic THEN ABL(0,serial_control);

END_IF;
```

the ABL instruction will execute every scan that *tag_xic* is set, not just when *tag_xic* transitions from cleared to set.

If you want the ABL instruction to execute only when *tag_xic* transitions from cleared to set, you have to condition the structured text instruction. Use a one shot to trigger execution.

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);


IF (osri_1.OutputBit) THEN
        ABL(0,serial_control);
END_IF;
```

Constructs

Constructs can be programmed singly or nested within other constructs.

If you want to:	Use this construct:	See page:
do something if or when specific conditions occur	IF...THEN	3-16
select what to do based on a numerical value	CASE...OF	3-17
do something a specific number of times before doing anything else	FOR...DO	3-19
keep doing something as long as certain conditions are true	WHILE...DO	3-21
keep doing something until a condition is true	REPEAT...UNTIL	3-23

IF...THEN

Use IF…THEN to do something if or when specific conditions occur. The syntax is:

```
                   IF bool_expression1 THEN
                             <statement >;       <──────  statements to execute when
                                                          bool_expression1 is true
                             .
                             .
                             .
optional         ┌ ELSIF bool_expression2 THEN
               ──┤           <statement>;        <──────  statements to execute when
                 └                                        bool_expression2 is true
                             .
                             .
                             .
optional         ┌ ELSE
               ──┤           <statement>;        <──────  statements to execute when both
                 └                                        expressions are false
                             .
                             .
                             .
                   END_IF;
```

CASE…OF

Use CASE to select what to do based on a numerical value. The syntax is:

```
CASE numeric_expression OF
     selector1 :     <statement>;
                     .
                     .
                     .
     selector2 :     <statement>;
                     .
                     .
                     .
     selector3 :     <statement>;
                     .
                     .
                     .
ELSE
                     <statement>;
                     .
                     .
                     .
END_CASE;
```

specify as many alternative selector values (paths) as you need

statements to execute when *numeric_expression* = *selector1*

statements to execute when *numeric_expression* = *selector2*

statements to execute when *numeric_expression* = *selector3*

optional

statements to execute when *numeric_expression* ≠ any selector

The syntax for entering the selector values is:

When selector is:	**Enter:**
one value	*value*: *statement*
multiple, distinct values	*value1*, *value2*, *valueN* : <*statement*> Use a comma (,) to separate each value.
a range of values	*value1*..*valueN* : <*statement*> Use two periods (..) to identify the range.
distinct values plus a range of values	*valuea, valueb, value1*..*valueN* : <*statement*>

FOR…DO

Use the FOR…DO loop to do something a specific number of times before doing anything else. The syntax is:

```
FOR count := initial_value
    TO final_value
    BY increment
    DO
    <statement>;
    IF bool_expression THEN
        EXIT;
    END_IF;
END_FOR;
```

optional { BY *increment*

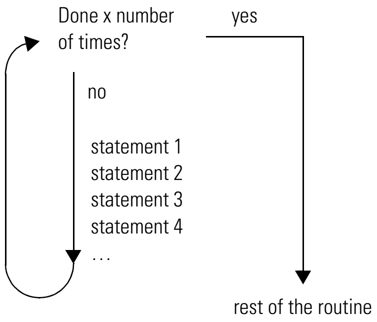
If you don’t specify an increment, the loop increments by 1.

optional { IF *bool_expression* THEN / EXIT; / END_IF;

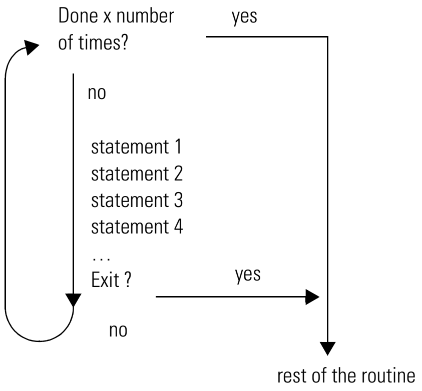
If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.

A major fault will occur if:	Fault type:	Fault code:
the construct loops too long	6	1

The following diagrams show how a FOR...DO loop executes and how an EXIT statement leaves the loop early.



The FOR...DO loop executes a specific number of times.



To stop the loop before the count reaches the last value, use an EXIT statement.

WHILE…DO

Use the WHILE…DO loop to keep doing something as long as certain conditions are true. The syntax is:

```
WHILE bool_expression1 DO
    <statement>;
    IF bool_expression2 THEN
        EXIT;
    END_IF;
END_WHILE;
```

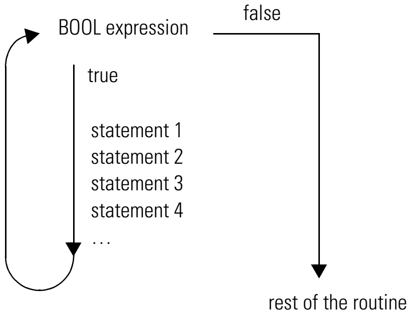
statements to execute while *bool_expression1* is true

optional

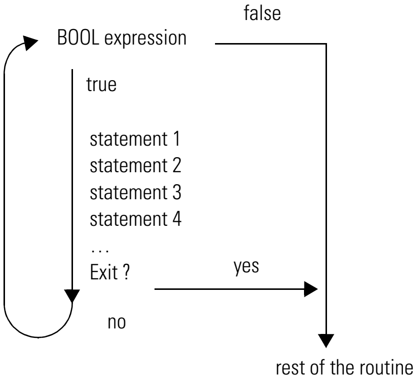
If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.

A major fault will occur if:	Fault type:	Fault code:
the construct loops too long	6	1

The following diagrams show how a WHILE...DO loop executes and how an EXIT statement leaves the loop early.



While the *`bool_expression`* is true, the controller executes only the statements within the WHILE…DO loop.



To stop the loop before the conditions are true, use an EXIT statement.

REPEAT…UNTIL

Use the REPEAT…UNTIL loop to keep doing something until conditions are true. The syntax is:

```
REPEAT
    <statement>;
    IF bool_expression2 THEN
        EXIT;
    END_IF;
UNTIL bool_expression1
END_REPEAT;
```

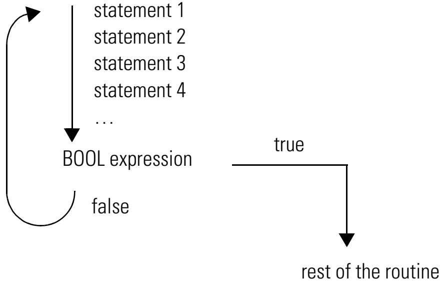
optional

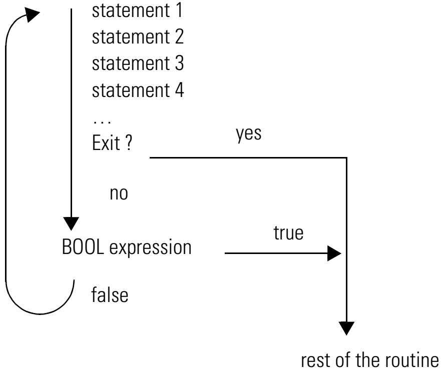
statements to execute while *bool_expression1* is false

If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.

A major fault will occur if:	**Fault type:**	**Fault code:**
the construct loops too long	6	1

The following diagrams show how a REPEAT...UNTIL loop executes and how an EXIT statement leaves the loop early.





While the *`bool_expression`* is false, the controller executes only the statements within the REPEAT…UNTIL loop.

To stop the loop before the conditions are false, use an EXIT statement.

Comments

To add comments to your structured text:

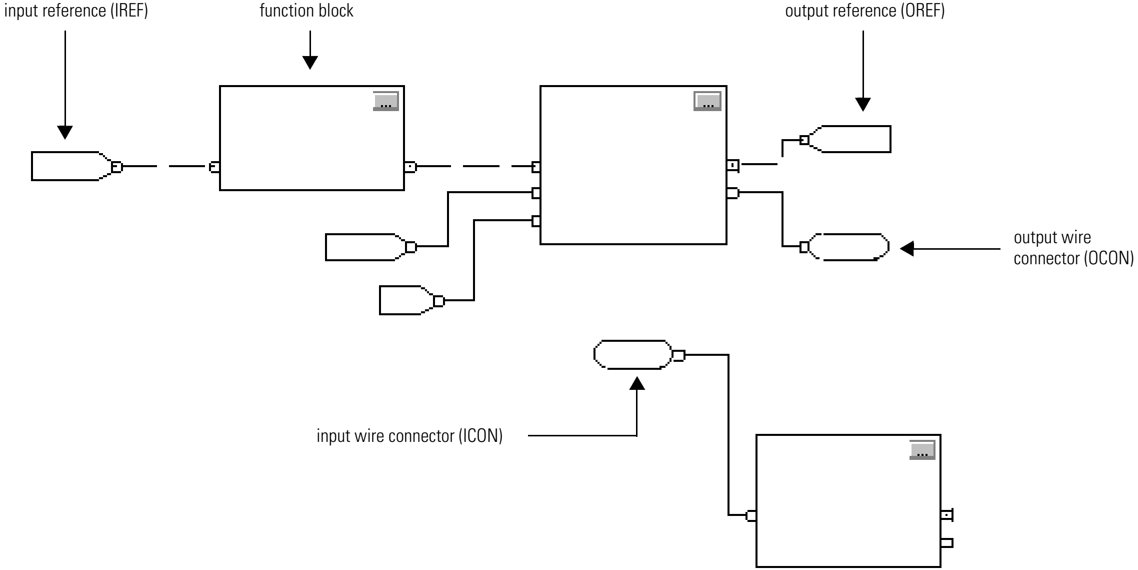
To add a comment:	Use one of these formats:
on a single line	//_comment_ (*_comment_*) /*_comment_*/
at the end of a line of structured text	
within a line of structured text	(*_comment_*) /*_comment_*/
that spans more than one line	(*_start of comment . . . end of comment_*) /*_start of comment . . . end of comment_*/

Notes:

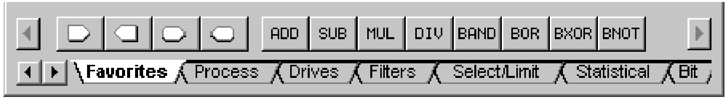
Function Block Diagram

Chapter **4**

Function block diagrams are visual programs that can contain the following elements. Each function block is an instruction that defines a control action.:

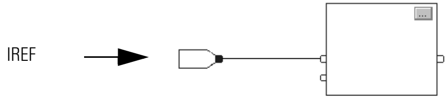
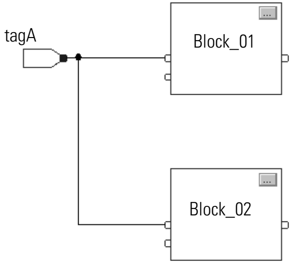


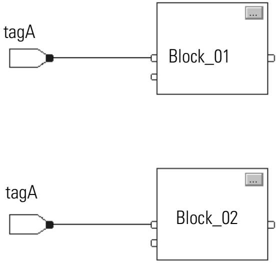
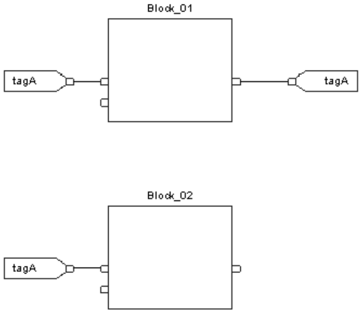
Editing a Function Block Diagram.



This toolbar button:	Creates this ladder element:	Description:
▷	IREF	Add an input reference to supply a value from an input device or tag.
◁	OREF	Add an output reference to send a value to an output device or tag.
⬭	ICON	Add input and output wire connectors. Use wire connectors to transfer data between function blocks when they are: • far apart on the same sheet • on different sheets within the same routine Use wire connectors to disperse data to several points in the routine by assigning one OCON to multiple ICONs.
⬭	OCON	
ADD SUB MUL DIV	instruction	Select a specific function block to perform an operation on an input value or values and produce an output value or values Use the tabs on the bottom of the toolbar to display other available function blocks.

Data Latching

Condition:	Example:
If you use an IREF to specify input data for a function block instruction, the data in that IREF is latched for the scan of the function block routine. The IREF latches data from program-scoped and controller-scoped tags. The controller updates all IREF data at the beginning of each scan.	
In this example, the value of tagA is stored at the beginning of the routine's execution. The stored value is used when Block_01 executes. The same stored value is also used when Blcock_02 executes. If the value of tagA changes during execution of the routine, the stored value of tagA in the IREF does not change until the next execution of the routine.	

Condition:	Example:
This example is the same as the one above. The value of tagA is stored only once at the beginning of the routine's execution. The routine uses this stored value throughout the routine.	
You can use the same tag in multiple IREFs and an OREF in the same routine. Because the values of tags in IREFs are latched every scan through the routine, all IREFs will use the same value, even if an OREF obtains a different tag value during execution of the routine. In this example, if tagA has a value of 25.4 when the routine starts executing this scan, and Block_01 changes the value of tagA to 50.9, the second IREF wired into Block_02 will still use a value of 25.4 when Block_02 executes this scan. The new tagA value of 50.9 will not be used by any IREFs in this routine until the start of the next scan.	

Order of Execution

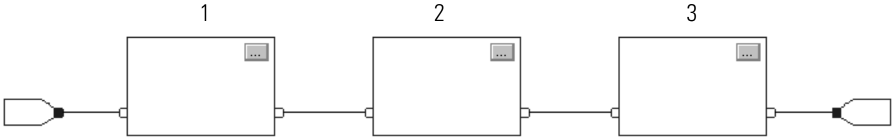
The RSLogix 5000 programming software automatically determines the order of execution for the function blocks in a routine when you:

- verify a function block routine
- verify a project that contains a function block routine
- download a project that contains a function block routine

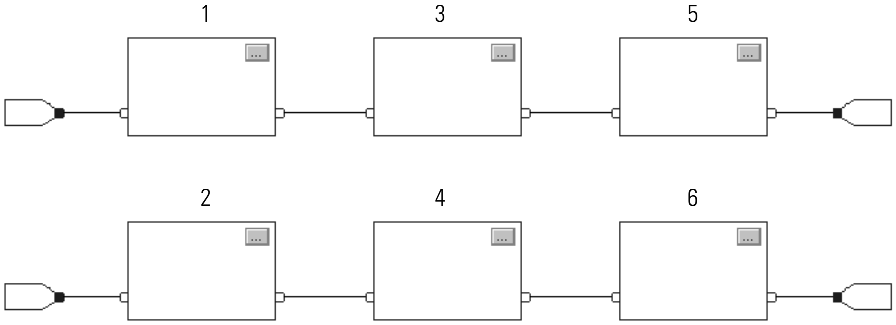
You define execution order by wiring function blocks together and indicating the data flow of any feedback wires, if necessary.

If function blocks are not wired together, it does not matter which block executes first. There is no data flow between the blocks.

If you wire the blocks sequentially, the execution order moves from input to output. The inputs of a block require data to be available before the controller can execute that block. For example, block 2 has to execute before block 3 because the outputs of block 2 feed the inputs of block 3.

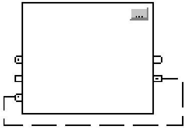


Execution order is only relative to the blocks that are wired together. The following example is fine because the two groups of blocks are not wired together. The blocks within a specific group execute in the appropriate order in relation to the blocks in that group.



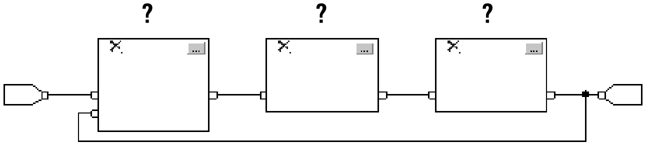
Resolve a Loop

To create a feedback loop around a block, wire an output pin of the block to an input pin of the same block. The following example is OK. The loop contains only a single block, so execution order does not matter.

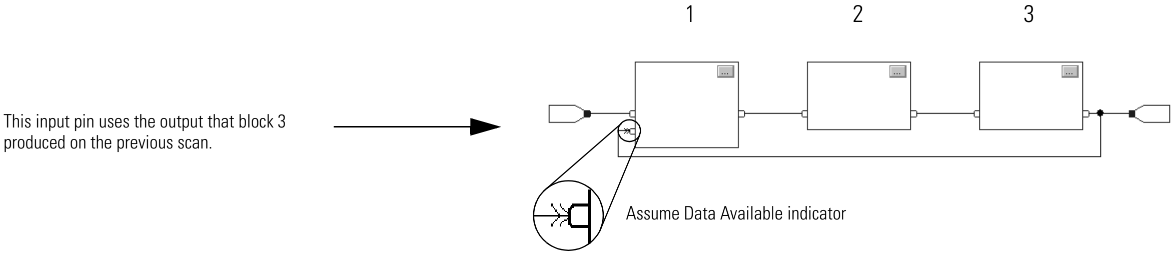


This input pin uses an output that the block produced on the previous scan.

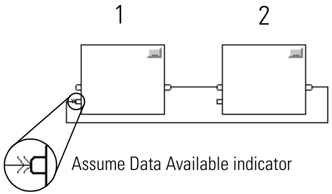
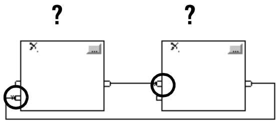
If a group of blocks are in a loop, the controller cannot determine which block to execute first. In other words, it cannot resolve the loop.



To identify which block to execute first, mark the input wire that creates the loop (the feedback wire) with the *Assume Data Available* indicator. In the following example, block 1 uses the output from block 3 that was produced in the previous execution of the routine.

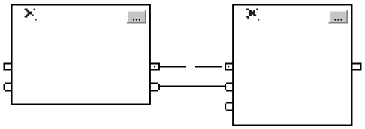
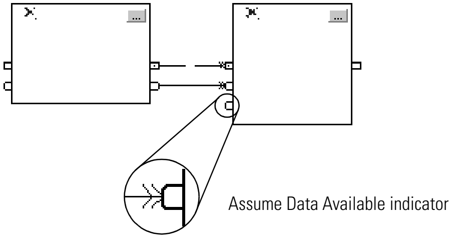
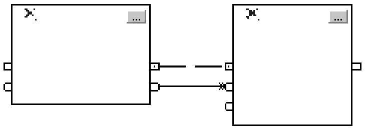

1
2
3
This input pin uses the output that block 3 produced on the previous scan.
Assume Data Available indicator

The *Assume Data Available* indicator defines the data flow within the loop. The arrow indicates that the data serves as input to the first block in the loop. *Do not* mark all the wires of a loop with the *Assume Data Available* indicator.

This is OK	This is NOT OK
	The controller cannot resolve the loop because all the wires use the *Assume Data Available* indicator. 

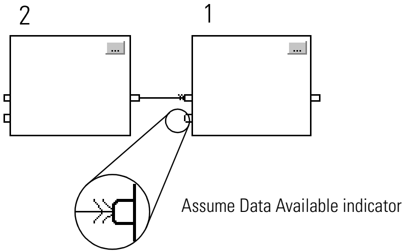
Resolve Data Flow Between Two Blocks

If you use two or more wires to connect two blocks, use the same data flow indicators for all of the wires between the two blocks.

This is OK	This is NOT OK
 Neither wire uses the *Assume Data Available* indicator.  Both wires use the *Assume Data Available* indicator.	One wire uses the *Assume Data Available* indicator while the other wire does not. 

Create a One Scan Delay

To produce a one scan delay between blocks, use the *Assume Data Available* indicator. In the following example, block 1 executes first. It uses the output from block 2 that was produced in the previous scan of the routine.



Summary

In summary, a function block routine executes in this order:

1. The controller latches all data values in IREFs.
2. The controller executes the other function blocks in the order determined by how they are wired.
3. The controller writes outputs in OREFs.

Define Program/Operator Control

Several instructions support the concept of Program/Operator control. These instructions include:

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

Program/Operator control lets you control these instructions simultaneously from both your user program and from an operator interface device. When in Program control, the instruction is controlled by the Program inputs to the instruction; when in Operator control, the instruction is controlled by the Operator inputs to the instruction.

Program or Operator control is determined by using these inputs:

Input:	**Description:**
.ProgProgReq	A program request to go to Program control.
.ProgOperReq	A program request to go to Operator control.
.OperProgReq	An operator request to go to Program control.
.OperOperReq	An operator request to go to Operator control.

To determine whether an instruction is in Program or Control control, examine the ProgOper output. If ProgOper is set, the instruction is in Program control; if ProgOper is cleared, the instruction is in Operator control.

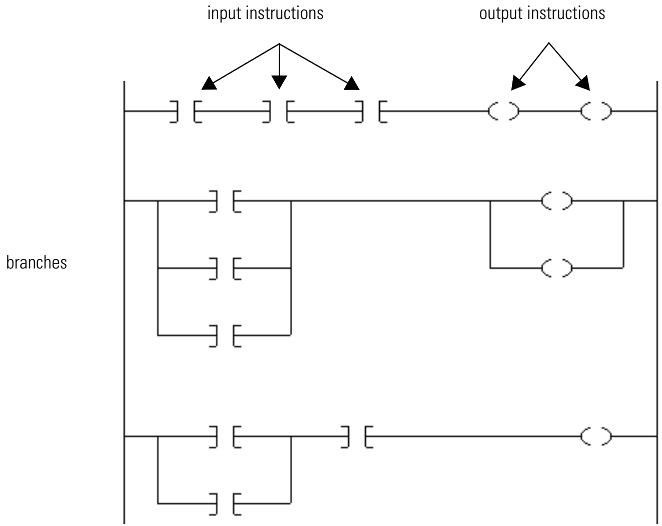
Control:	Description:
program	The Program request inputs take precedence over the Operator request inputs. This provides the capability to use the ProgProgReq and ProgOperReq inputs to “lock” an instruction in a desired control. Constantly setting the ProgProgReq can “lock” the instruction into Program control. This is useful for automatic startup sequences when you want the program to control the action of the instruction without worrying about an operator inadvertently taking control of the instruction. In this example, you have the program set the ProgProgReq input during the startup, and then clear the ProgProgReq input once the startup was complete. Once the ProgProgReq input is cleared, the instruction remains in Program control until it receives a request to change. For example, the operator could set the OperOperReq input from a faceplate to take over control of that instruction. Program request inputs are not normally cleared by the instruction because these are normally wired as inputs into the instruction. If the instruction clears these inputs, the input would just get set again by the wired input. There might be situations where you want to use other logic to set the Program requests in such a manner that you want the Program requests to be cleared by the instruction. In this case, you can set the ProgValueReset input and the instruction will always clear the Program mode request inputs when it executes.
operator	Operator request inputs to an instruction are always cleared by the instruction when it executes. This allows operator interfaces to work with these instructions by merely setting the desired mode request bit. You don’t have to program the operator interface to reset the request bits. Operator control takes precedence over Program control if both input request bits are set. For example, if ProgProgReq and ProgOperReq are both set, the instruction goes to Operator control.

Relay Ladder

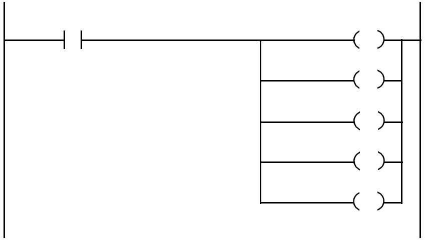
Chapter 5

Relay Ladder Logic

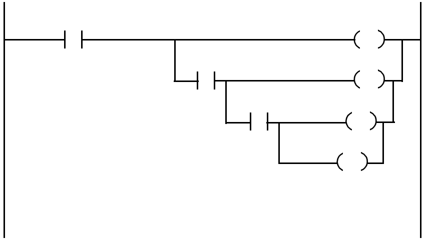
Relay ladder logic places input and output instructions on rungs.



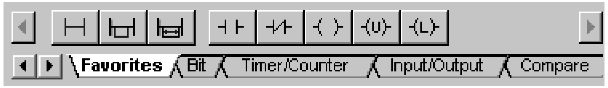
There is no limit to the number of parallel branch levels that you can enter. The following figure shows a parallel branch with five levels. The main rung is the first branch level, followed by four additional branches.

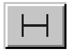
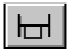
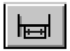
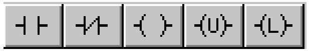


You can nest branches to as many as 6 levels. The following figure shows a nested branch. The bottom output instruction is on a nested branch that is three levels deep.



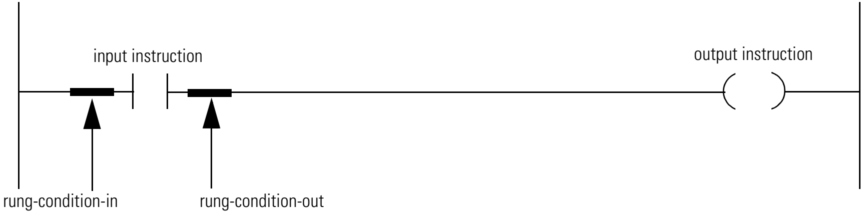
Editing Relay Ladder



This toolbar button:	Creates this ladder element:	Description:
	ladder rung	A rung determines the execution order of input and output instructions.
	branch	A branch is two or more instructions in parallel.
	a branch level	There is no limit to the number of parallel branch levels that you can enter. You can nest branches to as many as 6 levels.
	instruction	**Input instruction**: An instruction that checks, compares, or examines specific conditions in your machine or process. **Output instruction**: An instruction that takes some action, such as turn on a device, turn off a device, copy data, or calculate a value. Use the tabs on the bottom of the toolbar to display other available instructions.

Rung Condition

The controller evaluates ladder instructions based on the rung condition preceding the instruction (rung-condition-in).



Only input instructions affect the rung-condition-in of subsequent instructions on the rung:

- If the rung-condition-in to an input instruction is true, the controller evaluates the instruction and sets the rung-condition-out to match the results of the evaluation.
  - If the instruction evaluates to true, the rung-condition-out is true.
  - If the instruction evaluates to false, the rung-condition-out is false.
- An output instruction does not change the rung-condition-out.
  - If the rung-condition-in to an output instruction is true, the rung-condition-out is set to true.
  - If the rung-condition-in to an output instruction is false, the rung-condition-out is set to false.

Accessing System Values

Chapter 6

System Values Stored by the Controller

The controller automatically stored different status information:

If you want to:	See page:
use specific key words in logic to monitor specific status conditions	6-2
get or set system data (status information)	6-3
available status information - GSV/SSV objects	6-5
get information about controller memory	6-26

Monitor Status Flags

The controller supports status keywords you can use in your logic to monitor specific events:

To determine if:	Use:
the value you are storing cannot fit into the destination because it is either: • greater than the maximum value for the destination • less than the minimum value for the destination **Important:** Each time S:V goes from cleared to set, it generates a minor fault (type 4, code 4)	S:V
the instruction's destination value is 0	S:Z
the instruction's destination value is negative	S:N
an arithmetic operation causes a carry or borrow that tries to use bits that are outside of the data type	S:C
this is the first, normal scan of the routines in the current program	S:FS
at least one minor fault has been generated: • The controller sets this bit when a minor fault occurs due to program execution. • The controller does not set this bit for minor faults that are not related to program execution, such as battery low.	S:MINOR

The status keywords are not case sensitive. Because the status flags can change so quickly, RSLogix 5000 software does *not* display the status of the flags. You *cannot* define a tag alias to a keyword.

Get and Set System Data (Status Information)

The controller stores system data in objects. There is no status file, as in the PLC-5 controller. Use the GSV/SSV instructions get and set controller system data that is stored in objects. To get or set a system value:

1. Select the system object you want.

To get or set:	Select:	To get or set:	Select:
axis of a servo module	AXIS	status, faults, and mode of a module	MODULE
system overhead timeslice	CONTROLLER	group of axes	MOTIONGROUP
physical hardware of a controller	CONTROLLERDEVICE	fault information or scan time for a program	PROGRAM
coordinated system time for the devices in one chassis	CST	instance number of a routine	ROUTINE
DF1 communication driver for the serial port	DF1	configuration of the serial port	SERIALPORT
fault history for a controller	FAULTLOG	properties or elapsed time of a task	TASK
attributes of a message instruction	MESSAGE	wall clock time of a controller	WALLCLOCKTIME

2. In the list of attributes for the object, identify the attribute that you want to access.

3. Create a tag for the value of the attribute:

If the data type of the attribute is:	Then:
one element (e.g., DINT)	Create a tag for the attribute.
more than one element (e.g., DINT[7])	A. Create a user-defined data type that matches the organization of data for the attribute. B. Create a tag for the attribute.

4. In your logic, use a GSV instruction to get the value of an attribute or an SSV instruction to set the value of an attribute.

5. Assign the required operands to the instruction:

For this operand:	Select:
Class name	name of the object
Instance name	name of the specific object (e.g., name of the required I/O module, task, message) Not all objects require this entry. To specify the current task, program, or routine, select *THIS*.
Attribute Name	name of the attribute
Dest (GSV)	tag that will store the retrieved value If the tag is a user-defined data type or an array, select the first member or element.
Source (SSV)	tag that stores the value to be set If the tag is a user-defined data type or an array, select the first member or element.

Available Status Information - GSV/SSV Objects

CONTROLLER attributes

Attribute:	Data Type:	Instruction:	Description:
TimeSlice	INT	GSV SSV	Percentage of available CPU that is assigned to communications. Valid values are 10-90. This value cannot be changed when the keyswitch is in the run position.

CONTROLLERDEVICE attributes

Attribute:	Data Type:	Instruction:	Description:
DeviceName	SINT[33]	GSV	ASCII string that identifies the catalog number of the controller and memory board. The first byte contains a count of the number of ASCII characters returned in the array string.
ProductCode	INT	GSV	Identifies the type of controller: **Value:** **Meaning:** 3 ControlLogix5550 15 SoftLogix5860 41 FlexLogix5433 43 FlexLogix5434 48 PowerFlex 700S with DriveLogix5720 50 CompactLogix5320 51 ControlLogix5555 52 PowerFlex 700S with DriveLogix5730

Attribute:	**Data Type:**	**Instruction:**	**Description:**
ProductRev	INT	GSV	Identifies the current product revision. Display should be hexadecimal. The low byte contains the major revision; the high byte contains the minor revision.
SerialNumber	DINT	GSV	Serial number of the device. The serial number is assigned when the device is built.
Status	INT	GSV	**Device Status Bits** **Bits 7-4: Meaning:** 0000 reserved 0001 flash update in progress 0010 reserved 0011 reserved 0100 flash is bad 0101 faulted 0110 run 0111 program **Fault Status Bits** **Bits 11-8: Meaning:** 0001 recoverable minor fault 0010 unrecoverable minor fault 0100 recoverable major fault 1000 unrecoverable major fault **Controller Status Bits** **Bits 13-12: Meaning:** 01 keyswitch in run 10 keyswitch in program 11 keyswitch in remote **Bits 15-14 Meaning** 01 controller is changing modes 10 debug mode if controller is in run mode
Type	INT	GSV	Identifies the device as a controller. Controller = 14
Vendor	INT	GSV	Identifies the vendor of the device. Allen-Bradley = 0001

CST attributes

Attribute:	Data Type:	Instruction:	Description:
CurrentStatus	INT	GSV	Current status of the coordinated system time. **Bit:** **Meaning:** 0 timer hardware faulted: the device’s internal timer hardware is in a faulted state 1 ramping enabled: the current value of the timer’s lower 16+ bits ramp up to the requested value, rather than snap to the lower value. 2 system time master: the CST object is a master time source in the ControlLogix system 3 synchronized: the CST object’s 64-bit CurrentValue is synchronized by a master CST object via a system time update 4 local network master: the CST object is the local network master time source 5 in relay mode: the CST object is acting in a time relay mode 6 duplicate master detected: a duplicate local network time master has been detected. This bit is always 0 for time-dependent nodes. 7 unused 8-9 00 = time dependent node 01 = time master node 10 = time relay node 11 = unused 10-15 unused
CurrentValue	DINT[2]	GSV	Current value of the timer. DINT[0] contains the lower 32; DINT[1] contains the upper 32 bits. The timer source is adjusted to match the value supplied in update services and from local communication network synchronization. The adjustment is either a ramping to the requested value or an immediate setting to the request value, as reported in the CurrentStatus attribute.

DF1 attributes

Attribute:	Data Type:	Instruction:	Description:
ACKTimeout	DINT	GSV	The amount of time to wait for an acknowledgment to a message transmission (point-to-point and master only). Valid value 0-32,767. Delay in counts of 20 msec periods. Default is 50 (1 second).
DiagnosticCounters	INT[19]	GSV	Array of diagnostic counters for the DF1 communication driver.

word offset	DF1 point-to-point	DF1 slave	master
0	signature (0x0043)	signature (0x0042)	signature (0x0044)
1	modem bits	modem bits	modem bits
2	packets sent	packets sent	packets sent
3	packets received	packets received	packets received
4	undelivered packets	undelivered packets	undelivered packets
5	unused	messages retried	messages retried
6	NAKs received	NAKs received	unused
7	ENQs received	poll packets received	unused
8	bad packets NAKed	bad packets not ACKed	bad packets not ACKed
9	no memory sent NAK	no memory not ACKed	unused
10	duplicate packets received	duplicate packets received	duplicate packets received
11	bad characters received	unused	unused
12	DCD recoveries count	DCD recoveries count	DCD recoveries count
13	lost modem count	lost modem count	lost modem count
14	unused	unused	priority scan time maximum
15	unused	unused	priority scan time last
16	unused	unused	normal scan time maximum
17	unused	unused	normal scant time last
18	ENQs sent	unused	unused

Attribute:	**Data Type:**	**Instruction:**	**Description:**
DuplicateDetection	SINT	GSV	Enables duplicate message detection. **Value:** **Meaning:** 0 duplicate message detection disabled non zero duplicate message detection enabled
EmbeddedResponseEnable	SINT	GSV	Enables embedded response functionality (point-to-point only). **Value:** **Meaning:** 0 initiated only after one is received (default) 1 enabled unconditionally
ENQTransmitLimit	SINT	GSV	The number of inquiries (ENQs) to send after an ACK timeout (point-to-point only). Valid value 0-127. Default setting is 3.
EOTSuppression	SINT	GSV	Enable suppressing EOT transmissions in response to poll packets (slave only). **Value:** **Meaning:** 0 EOT suppression disabled (disabled) non zero EOT suppression enabled
ErrorDetection	SINT	GSV	Specifies the error-detection scheme. **Value:** **Meaning:** 0 BCC (default) 1 CRC
MasterMessageTransmit	SINT	GSV	Current value of the master message transmission (master only). **Value:** **Meaning:** 0 between station polls (default) 1 in poll sequence (in place of master's station number)
NAKReceiveLimit	SINT	GSV	The number of NAKs received in response to a message before stopping transmission (point-to-point communication only). Valid value 0-127. Default is 3.

Attribute:	Data Type:	Instruction:	Description:
NormalPollGroupSize	INT	GSV	Number of stations to poll in the normal poll node array after polling all the stations in the priority poll node array (master only). Valid value 0-255. Default is 0.
PollingMode	SINT	GSV	Current polling mode (master only). Default setting is 1. **Value:** **Meaning:** 0 message-based, but don't allow slaves to initiate messages 1 message-based, but allow slaves to initiate messages (default) 2 standard, single-message transfer per node scan 3 standard, multiple-message transfer per node scan
ReplyMessageWait	DINT	GSV	The time (acting as a master) to wait after receiving an ACK before polling the slave for a response (master only). Valid value 0-65,535. Delay in counts of 20 msec periods. The default is 5 periods (100 msec).
StationAddress	INT	GSV	Current station address of the serial port. Valid value 0-254. Default is 0.
SlavePollTimeout	DINT	GSV	The amount of time in msecs that the slave waits for the master to poll before the slave declares that it is unable to transmit because the master is inactive (slave only). Valid value 0-32,767. Delay in counts of 20 msec periods. The default is 3000 periods (1 minute).
TransmitRetries	SINT	GSV	Number of times to resend a message without getting an acknowledgment (master and slave only). Valid value 0-127. Default is 3.
PendingACKTimeout	DINT	SSV	Pending value for the ACKTimeout attribute.
PendingDuplicateDetection	SINT	SSV	Pending value for the DuplicateDetection attribute.
PendingEmbeddedResponseEnable	SINT	SSV	Pending value for the EmbeddedResponse attribute.
PendingENQTransmitLimit	SINT	SSV	Pending value for the ENQTransmitLimit attribute.
PendingEOTSuppression	SINT	SSV	Pending value for the EOTSuppression attribute.

Attribute:	Data Type:	Instruction:	Description:
PendingErrorDetection	SINT	SSV	Pending value for the ErrorDetection attribute.
PendingNormalPollGroupSize	INT	SSV	Pending value for the NormalPollGroupSize attribute.
PendingMasterMessageTransmit	SINT	SSV	Pending value for the MasterMessageTransmit attribute.
PendingNAKReceiveLimit	SINT	SSV	Pending value for the NAKReceiveLimit attribute.
PendingPollingMode	SINT	SSV	Pending value for the PollingMode attribute.
PendingReplyMessageWait	DINT	SSV	Pending value for the ReplyMessageWait attribute.
PendingStationAddress	INT	SSV	Pending value for the StationAddress attribute.
PendingSlavePollTimeout	DINT	SSV	Pending value for the SlavePollTimeout attribute.
PendingTransmitRetries	SINT	SSV	Pending value for the TransmitRetries attribute.

FAULTLOG attributes

Attribute:	Data Type:	Instruction:	Description:
MajorEvents	INT	GSV SSV	How many major faults have occurred since the last time this counter was reset.
MinorEvents	INT	GSV SSV	How many minor faults have occurred since the last time this counter was reset.

Attribute:	Data Type:	Instruction:	Description:
MajorFaultBits	DINT	GSV SSV	Individual bits indicate the reason for the current major fault. **Bit:** **Meaning:** 1 power loss 3 I/O 4 instruction execution (program) 5 fault handler 6 watchdog 7 stack 8 mode change 11 motion
MinorFaultBits	DINT	GSV SSV	Individual bits indicate the reason for the current minor fault. **Bit:** **Meaning:** 4 instruction execution (program) 6 watchdog 9 serial port 10 battery

MESSAGE attributes

Attribute:	Data Type:	Instruction:	Description:
ConnectionPath	SINT[130]	GSV SSV	Data to setup the connection path. The first two bytes (low byte and high byte) are the length in bytes of the connection path.
ConnectionRate	DINT	GSV SSV	Requested packet rate of the connection.

Attribute:	**Data Type:**	**Instruction:**	**Description:**
MessageType	SINT	GSV SSV	Specifies the type of message. **Value:** **Meaning:** 0 not initialized
Port	SINT	GSV SSV	Indicates which port the message should be sent on. **Value:** **Meaning:** 1 backplane 2 serial port
TimeoutMultiplier	SINT	GSV SSV	Determines when a connection should be considered timed out and closed. **Value:** **Meaning:** 0 connection will timeout in 4 times the update rate default) 1 connection will timeout in 8 times the update rate 2 connection will timeout in 16 times the update rate
UnconnectedTimeout	DINT	GSV SSV	Timeout in microseconds for all unconnected messages. The default is 30,000,000 microseconds (30 s).

MODULE attributes

Attribute:	Data Type:	Instruction:	Description:
EntryStatus	INT	GSV	Specifies the current state of the specified map entry. The lower 12 bits should be masked when performing a comparison operation. Only bits 12-15 are valid. **Value:** **Meaning:** 16#0000 **Standby:** the controller is powering up. 16#1000 **Faulted:** any of the MODULE object's connections to the associated module fail. This value should not be used to determine if the module failed because the MODULE object leaves this state periodically when trying to reconnect to the module. Instead, test for Running state (16#4000). Check for FaultCode not equal to 0 to determine if a module is faulted. When Faulted, the FaultCode and FaultInfo attributes are valid until the fault condition is corrected. 16#2000 **Validating:** the MODULE object is verifying MODULE object integrity prior to establishing connections to the module. 16#3000 **Connecting**: the MODULE object is initiating connections to the module. 16#4000 **Running:** all connections to the module are established and data is transferring. 16#5000 **Shutting down:** the MODULE object is in the process of shutting down all connections to the module. 16#6000 **Inhibited:** the MODULE object is inhibited (the inhibit bit in the Mode attribute is set). 16#7000 **Waiting:** the parent object upon which this MODULE object depends is not running.
FaultCode	INT	GSV	A number which identifies a module fault, if one occurs.
FaultInfo	DINT	GSV	Provides specific information about the MODULE object fault code.
ForceStatus	INT	GSV	Specifies the status of forces. **Bit:** **Meaning:** 0 forces installed (1=yes, 0-no) 1 forces enabled (1=yes, 0=no)
Instance	DINT	GSV	Provides the instance number of this MODULE object.

Attribute:	**Data Type:**	**Instruction:**	**Description:**
LEDStatus	INT	GSV	Specifies the current state of the I/O LED on the front of the controller. **Value:** **Meaning:** 0 **LED off:** No MODULE objects are configured for the controller (there are no modules in the I/O Configuration section of the controller organizer). 1 Flashing red: None of the MODULE objects are Running. 2 Flashing green: At least one MODULE object is not Running. 3 Solid green: All the Module objects are Running. **Note:** You do not enter an object name with this attribute because this attribute applies to the entire collection of modules.
Mode	INT	GSV SSV	Specifies the current mode of the MODULE object. **Bit:** **Meaning:** 0 If set, causes a major fault to be generated if any of the MODULE object connections fault while the controller is in Run mode. 2 If set, causes the MODULE object to enter Inhibited state after shutting down all the connections to the module.

PROGRAM attributes

Attribute:	**Data Type:**	**Instruction:**	**Description:**
DisableFlag	SINT	GSV SSV	Controls this program’s execution. **Value:** **Meaning:** 0 execution enabled 1 execution disabled
Instance	DINT	GSV	Provides the instance number of this PROGRAM object.

Attribute:	**Data Type:**	**Instruction:**	**Description:**
LastScanTime	DINT	GSV SSV	Time it took to execute this program the last time it was executed. Time is in microseconds.
MajorFaultRecord	DINT[11]	GSV SSV	Records major faults for this program We recommend that you create a user-defined structure to simplify access to the MajorFaultRecord attribute:

Name:	**Data Type:**	**Style:**	**Description:**
TimeLow	DINT	Decimal	lower 32 bits of fault timestamp value
TimeHigh	DINT	Decimal	upper 32 bits of fault timestamp value
Type	INT	Decimal	fault type (program, I/O, etc.)
Code	INT	Decimal	unique code for the fault (depends on fault type)
Info	DINT[8]	Hexadecimal	fault specific information (depends on fault type and code)

MaxScanTime	DINT	GSV SSV	Maximum recorded execution time for this program. Time is in microseconds.
MinorFaultRecord	DINT[11]	GSV SSV	Records minor faults for this program We recommend that you create a user-defined structure to simplify access to the MinorFaultRecord attribute:

Name:	**Data Type:**	**Style:**	**Description:**
TimeLow	DINT	Decimal	lower 32 bits of fault timestamp value
TimeHigh	DINT	Decimal	upper 32 bits of fault timestamp value
Type	INT	Decimal	fault type (program, I/O, etc.)
Code	INT	Decimal	unique code for the fault (depends on fault type)
Info	DINT[8]	Hexadecimal	fault specific information (depends on fault type and code)

SFCRestart	INT	GSV SSV	unused - reserved for future use

REDUNDANCY attributes

Attribute:	Data Type:	Instruction:	Description:
ChassisRedundancyState	INT	GSV	Redundancy status of the entire chassis. **Value:** **Meaning:** 16#1 power-up or undetermined 16#2 primary with qualified secondary 16#3 primary with disqualified secondary 16#4 primary with no secondary
CompatibilityResults	INT	GSV	The results of the compatibility checks with the partner controller. **Value:** **Meaning:** 0 undetermined 1 no compatible partner 2 fully compatible partner
KeyswitchAlarm	DINT	GSV	The keyswitch settings of the controller and its partner match or do not match. **Value:** **Meaning:** 0 the keyswitches match or no partner is present 1 keyswitches do not match
ModuleRedundancyState	INT	GSV	Redundancy status of the controller. **Value:** **Meaning:** 16#1 power-up or undetermined 16#2 primary with qualified secondary 16#3 primary with disqualified secondary 16#4 primary with no secondary 16#6 primary with qualifying secondary

Attribute:	Data Type:	Instruction:	Description:
PartnerChassisRedundancyState	INT	GSV	Redundancy state of the partner chassis. **Value:** **Meaning:** 16#8 qualified secondary 16#9 disqualified secondary with primary
PartnerKeyswitch	DINT	GSV	Position of the keyswitch of the partner. **Value:** **Meaning:** 0 unknown 1 RUN 2 PROG 3 REM
PartnerMinorFaults	DINT	GSV	Minor faults of the partner (if the ModuleRedundancyState indicates that a partner is present). **Value:** **Meaning:** 4 problem with an instruction (program) 6 periodic task overlap (watchdog) 9 problem with the serial port 10 low battery

Attribute:	Data Type:	Instruction:	Description:
PartnerMode	DINT	GSV	Mode of the partner. **Value:** **Meaning:** 16#0 power up 16#1 program 16#2 run 16#3 test 16#4 faulted 16#5 run-to-program 16#6 test-to-program 16#7 program-to-run 16#8 test-to-run 16#9 run-to-test 16#A program-to-test 16#B into faulted 16#C faulted-to-program
PartnerModuleRedundancyState	INT	GSV	Redundancy state of the partner. **Value:** **Meaning:** 16#7 qualifying secondary 16#8 qualified secondary 16#9 disqualified secondary with primary
PhysicalChassisID	INT	GSV	In a pair of redundant chassis, identifies a specific chassis without regard to the state of the chassis. **Value:** **Meaning:** 0 unknown 1 Chassis A 2 Chassis B

Attribute:	Data Type:	Instruction:	Description:
QualificationInProgress	INT	GSV	Status of the qualification process. **Value:** **Meaning:** -1 qualification is not in progress 0 unsupported 1 - 99 for modules that can measure their completion percentage, the percent of qualification that is complete; for modules that cannot measure their completion percentage, 50 = qualification is in progress and 100 = qualification is complete.
SRMSlotNumber	INT	GSV	Slot number of the 1757-SRM module in this chassis
LastDataTransferSize	DINT	GSV	This attribute is only valid on a primary controller that is configured for redundancy. **If:** **Then this value is the:** a synchronized partner is present amount of data that was last transferred to the partner, specified in DINTs no partner is present or a disqualified partner is present amount of data that would have been last transferred to a synchronized partner, specified in DINTs
MaxDataTransferSize	DINT	GSV SSV	Maximum value of the LastDataTransferSize attribute. This attribute is only valid on a primary controller that is configured for redundancy. To reset this value, use an SSV instruction with a Source value of 0.

ROUTINE attributes

Attribute:	Data Type:	Instruction:	Description:
Instance	DINT	GSV	Provides the instance number of this ROUTINE object. Valid values are 0-65,535.

SERIALPORT attribute

Attribute:	Data Type:	Instruction:	Description:
BaudRate	DINT	GSV	Specifies the baud rate. Valid values are 110, 300, 600, 1200, 2400, 4800, 9600, and 19200 (default).
DataBits	SINT	GSV	Specifies the number of bits of data per character. **Value:** **Meaning:** 7 7 data bits (ASCII only) 8 8 data bits (default)
Parity	SINT	GSV	Specifies the parity. **Value:** **Meaning:** 0 no parity (no default) 1 odd parity (ASCII only) 2 even parity
RTSOffDelay	INT	GSV	Amount of time to delay turning off the RTS line after the last character has been transmitted. Valid value 0-32,767. Delay in counts of 20 msec periods. The default is 0 msec.
RTSSendDelay	INT	GSV	Amount of time to delay transmitting the first character of a message after turning on the RTS line. Valid value 0-32,767. Delay in counts of 20 msec periods. The default is 0 msec.
StopBits	SINT	GSV	Specifies the number of stop bits. **Value:** **Meaning:** 1 1 stop bit (default) 2 2 stop bits (ASCII only)
PendingBaudRate	DINT	SSV	Pending value for the BaudRate attribute.
PendingDataBits	SINT	SSV	Pending value for the DataBits attribute.
PendingParity	SINT	SSV	Pending value for the Parity attribute.

Attribute:	Data Type:	Instruction:	Description:
PendingRTSOffDelay	INT	SSV	Pending value for the RTSOffDelay attribute.
PendingRTSSendDelay	INT	SSV	Pending value for the RTSSendDelay attribute.
PendingStopBits	SINT	SSV	Pending value for the StopBits attribute.

TASK attributes

Attribute:	Data Type:	Instruction:	Description:
DisableUpdateOutputs	DINT	GSV SSV	Enables or disables the processing of outputs at the end of a task. **Value:** **Meaning:** 0 enable the processing of outputs at the end of the task non zero disable the processing of outputs at the end of the task
InhibitTask	DINT	GSV SSV	Prevents the task from executing. If a task is inhibited, the controller still prescans the task when the controller transitions from program to run or test mode. **Value:** **Meaning:** 0 enable the task 0 (default) non zero inhibit (disable) the task
Instance	DINT	GSV	Provides the instance number of this TASK object. Valid values are 0-31.
LastScanTime	DINT	GSV SSV	Time it took to execute this task the last time it was executed. Time is in microseconds.
MaxInterval	DINT[2]	GSV SSV	The maximum time interval between successive executions of the task. DINT[0] contains the lower 32 bits of the value; DINT[1] contains the upper 32 bits of the value. A value of 0 indicates 1 or less executions of the task.

Attribute:	Data Type:	Instruction:	Description:
MaxScanTime	DINT	GSV SSV	Maximum recorded execution time for this program. Time is in microseconds.
MinInterval	DINT[2]	GSV SSV	The minimum time interval between successive executions of the task. DINT[0] contains the lower 32 bits of the value; DINT[1] contains the upper 32 bits of the value. A value of 0 indicates 1 or less executions of the task.
OverlapCount	DINT	GSV SSV	Number of times that the task was triggered while it was still executing. Valid for an event or a periodic task. To clear the count, set the attribute to 0.
Priority	INT	GSV	Relative priority of this task as compared to the other tasks. Valid values 0-15.
Rate	DINT	GSV	The time interval between executions of the task. Time is in microseconds.
StartTime	DINT[2]	GSV SSV	Value of WALLCLOCKTIME when the last execution of the task was started. DINT[0] contains the lower 32 bits of the value; DINT[1] contains the upper 32 bits of the value.
Status	DINT	GSV SSV	Status information about the task. Once the controller sets one of these bits, you must manually clear the bit. **Bit:** **Meaning:** 0 an EVNT instruction triggered the task (event task only) 1 a timeout triggered the task (event task only) 2 an overlap occurred for this task

Attribute:	Data Type:	Instruction:	Description:
Timeout	DINT	GSV SSV	The timeout value for an event task. Time is in microseconds.
EnableTimeOut	DINT	GSV SSV	Enables or disables the timeout function of an event task. **Value:** **Meaning:** 0 disable the timeout function non zero enable the timeout function
Watchdog	DINT	GSV SSV	Time limit for execution of all programs associated with this task. Time is in microseconds. If you enter 0, these values are assigned: **Time:** **Task Type:** 0.5 sec periodic 5.0 sec continuous

WALLCLOCKTIME attributes

Attribute:	Data Type:	Instruction:	Description:
CSTOffset	DINT[2]	GSV SSV	Positive offset from the CurrentValue of the CST object (coordinated system time, see page 6-7). DINT[0] contains the lower 32 bits of the value; DINT[1] contains the upper 32 bits of the value. Value in μs. The default is 0.
CurrentValue	DINT[2]	GSV SSV	Current value of the wall clock time. DINT[0] contains the lower 32 bits of the value; DINT[1] contains the upper 32 bits of the value. The value is the number of microseconds that have elapsed since 0000 hours 1 January 1972. The CST and WALLCLOCKTIME objects are mathematically related in the controller. For example, if you add the CST CurrentValue and the WALLCLOCKTIME CTSOffset, the result is the WALLCLOCKTIME CurrentValue.
DateTime	DINT[7]	GSV SSV	The date and time in a readable format. DINT[0] year DINT[1] integer representation of month (1-12) DINT[2] integer representation of day (1-31) DINT[3] hour (0-23) DINT[4] minute (0-59) DINT[5] seconds (0-59) DINT[6] microseconds (0-999,999)

Determine Controller Memory Information

Depending on your type of controller, the memory of the controller may be divided into several areas:

If you have this controller:	Then it stores this:	In this memory:
ControlLogix	I/O tags	I/O memory
	produced tags	
	consumed tags	
	communication via Message (MSG) instructions	
	communication with workstations	
	communication with polled (OPC/DDE) tags that use RSLinx software[1]	
	tags other than I/O, produced, or consumed tags	data and logic memory[2]
	logic routines	
	communication with polled (OPC/DDE) tags that use RSLinx software[1]	
CompactLogix FlexLogix PowerFlex 700S with DriveLogix SoftLogix	These controllers do not divide their memory. They store all elements in one common memory area. When you use the following procedure to get the memory values for these controllers, the values show up as I/O memory.	

[1] To communicate with polled tags, the controller uses both I/O and data and logic memory.

[2] 1756-L55M16 controllers have an additional memory section for logic.

To get memory information from the controller, use a MSG instruction:

MSG Configuration Tab

For this item:	Type or select:	Which means:
Message Type	CIP Generic	Execute a Control and Information Protocol command.
Service Type	Custom	Create a CIP Generic message that is not available in the drop-down list.
Service Code	3	Use the GetAttributeList service. This lets you read specific information about the controller.
Class	72	Get information from the user memory object.
Instance	1	This object contains only 1 instance.
Attribute	0	Null value

For this item:	Type or select:	Which means:	
Source Element	*source_array* of type SINT[12]		
	In this element:	**Enter:**	**Which means:**
	source_array[0]	5	Get 5 attributes
	source_array[1]	0	Null value
	source_array[2]	1	Get free memory
	source_array[3]	0	Null value
	source_array[4]	2	Get total memory
	source_array[5]	0	Null value
	source_array[6]	5	Get largest contiguous block of additional free logic memory
	source_array[7]	0	Null value
	source_array[8]	6	Get largest contiguous block of free I/O memory
	source_array[9]	0	Null value
	source_array[10]	7	Get largest contiguous block of free data and logic memory
	source_array[11]	0	Null value
Source Length	12	Write 12 bytes (12 SINTs).	
Destination	*INT_array* of type INT[29]		

MSG Communication Tab

For this item:	Type:
Path	1, *slot_number_of_controller*

The MSG instruction returns the following information to *INT_array* (destination tag of the MSG):

If you want the:	Then copy these array elements:	Description:
amount of free I/O memory (32-bit words)	*INT_array*[3]	lower 16 bits of the 32 bit value
	INT_array[4]	upper 16 bits of the 32 bit value
amount of free data and logic memory (32-bit words)	*INT_array*[5]	lower 16 bits of the 32 bit value
	INT_array[6]	upper 16 bits of the 32 bit value
1756-L55M16 controllers only—amount of additional free logic memory (32-bit words)	*INT_array*[7]	lower 16 bits of the 32 bit value
	INT_array[8]	upper 16 bits of the 32 bit value
total size of I/O memory (32-bit words)	*INT_array*[11]	lower 16 bits of the 32 bit value
	INT_array[12]	upper 16 bits of the 32 bit value
total size of data and logic memory (32-bit words)	*INT_array*[13]	lower 16 bits of the 32 bit value
	INT_array[14]	upper 16 bits of the 32 bit value
1756-L55M16 controllers only—additional logic memory (32-bit words)	*INT_array*[15]	lower 16 bits of the 32 bit value
	INT_array[16]	upper 16 bits of the 32 bit value

If you want the:	**Then copy these array elements:**	**Description:**
1756-L55M16 controllers only—largest contiguous block of additional free logic memory (32-bit words)	*INT_array*[19]	lower 16 bits of the 32 bit value
	INT_array[20]	upper 16 bits of the 32 bit value
largest contiguous block of free I/O memory (32-bit words)	*INT_array*[23]	lower 16 bits of the 32 bit value
	INT_array[24]	upper 16 bits of the 32 bit value
largest contiguous block of free data and logic memory (32-bit words)	*INT_array*[27]	lower 16 bits of the 32 bit value
	INT_array[28]	upper 16 bits of the 32 bit value

The MSG instruction returns each memory value as two separate INTs.

- The first INT represents the lower 16 bits of the value.
- The second INT represents the upper 16 bits of the value.

To convert the separate INTs into one usable value, use a Copy (COP) instruction, where:

In this operand:	**Specify:**	**Which means:**
Source	first INT of the 2 element pair (lower 16 bits)	Start with the lower 16 bits
Destination	DINT tag in which to store the 32-bit value	Copy the value to the DINT tag.
Length	1	Copy 1 times the number of bytes in the Destination data type. In this case, the instruction copies 4 bytes (32 bits), which combines the lower and upper 16 bits into one 32-bit value.

Communicate with Other Controllers

Chapter **7**

Communication Options

Select a method for transferring data between controllers:

If the data:	Then:	See page:
needs regular delivery at a rate that you specify (i.e., deterministic)	produce and consume a tag	7-2
is sent when a specific condition occurs in your application	send a message	7-9
is transmitted between Logix controllers and PLC or SLC processors	map PLC/SLC addresses	7-13
is gathered from multiple controllers (and consumed tags are not an option or not desired)	send a message to multiple controllers	7-13

Produce and Consume a Tag

You can use produced and consumed tags with the following controller and network combinations.

This controller:	Can produce and consume tags over this network:		
	Logix Backplane	ControlNet	EtherNet/IP
SLC 500		X	
PLC-5		X	
ControlLogix	X	X	X
1769-L32E, -L35E CompactLogix			X
1769-L32C, -L35CR CompactLogix		X	
FlexLogix		X	X
PowerFlex 700S with DriveLogix		X	X
SoftLogix		X	X

Produced and consumed tags work as follows:

- A connection transfers the data between controllers:
  - Multiple controllers can consume (receive) the data.
  - The data updates at the requested packet interval (RPI), as configured by the consuming tags.

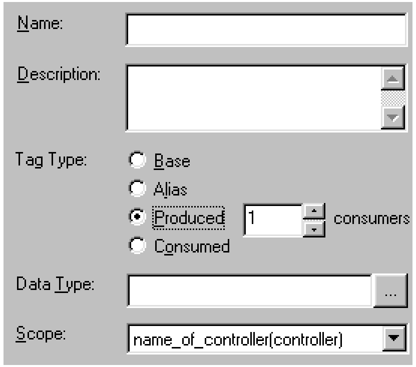
- Each produced or consumed tag uses the following number of connections:

Each:	Uses this many connections at the local controller:	Uses this many connections at the communication device:
produced tag	`number_of_consumers` +1	`number_of_consumers`
consumed tag	1	1

Follow these guidelines:

- Create the tags at the controller scope. You can only share controller-scoped tags.
- Use one of these data types:
  - DINT
  - REAL
  - array of DINTs or REALs
  - user-defined
- Use the same data type for the produced tag and corresponding consumed tag (s).
- To share tags with a PLC-5C controller, use a user-defined data type.
- Limit the size of the tag to less than or equal to 500 bytes. If you must transfer more than 500 bytes, transfer the data in packets.
- If you are producing several tags for the same controller:
  - Group the data into one or more user-defined data types. (This uses less connections than producing each tag separately.)
  - Group the data according to similar update rates. (To conserve network bandwidth, use a greater RPI for less critical data.)

Produce a tag



Name:

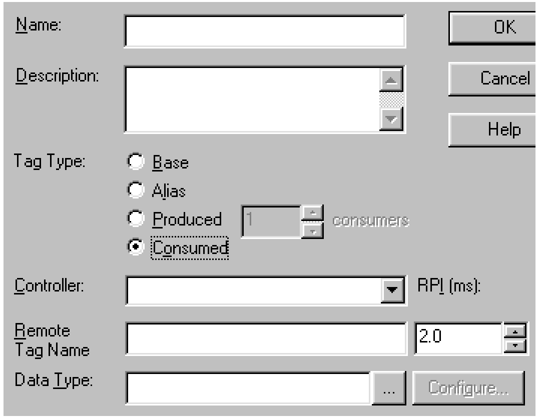
Description:

Tag Type: ○ Base ○ Alias ● Produced 1 consumers ○ Consumed

Data Type: ...

Scope: name_of_controller(controller)

Consume a tag



Name:

Description:

Tag Type: ○ Base ○ Alias ○ Produced 1 consumers ● Consumed

Controller:

RPI (ms): 2.0

Remote Tag Name

Data Type: ...

IMPORTANT If a consumed-tag connection fails, all of the other tags being consumed from that remote controller stop receiving new data.

Produce tags for a PLC-5C controller

1. Create a user-defined data type that contains an array of INTs with an even number of elements, such as INT[2]. (When you produce INTs, you must produce two or more.)

2. Create a produced tag and select the user-defined data type.

3. In the ControlNet configuration for the target PLC-5C controller:
   - Insert a Receive Scheduled Message.
   - In the Message size, enter the number of integers in the produced tag.

4. In RSNetWorx for ControlNet software, schedule the network.

Produce REALs for a PLC-5C controller

1. How many values do you want to produce?

If you are producing:	Then:
Only one REAL value	Create a produced tag and select the REAL data type.
More than one REAL value	A. Create a user-defined data type that contains an array of REALs. B. Create a produced tag and select the user-defined data type from Step A.

2. In the ControlNet configuration for the target PLC-5C controller:
   - Insert a Receive Scheduled Message.
   - In the Message size, enter two times the number of REALs in the produced tag. For example, if the produced tag contains 10 REALs, enter 20 for the Message size.

   When a PLC-5C controller consumes a tag that is produced by a Logix5000 controller, it stores the data in consecutive 16-bit integers. The PLC-5C stores floating-point data, which requires 32-bits regardless of the type of controller, as follows:

   - The first integer contains the upper (left-most) bits of the value.
   - The second integer contains the lower (right-most) bits of the value.
   - This pattern continues for each floating-point value.

3. In the PLC-5C controller, re-construct the floating point data, as depicted in the following example:
4. In RSNetWorx for ControlNet software, schedule the network.

Consume Integers from a PLC-5C Controller

1. In the ControlNet configuration of the PLC-5C controller, insert a Send Scheduled Message.
2. In the controller organizer, add the PLC-5C controller to the I/O configuration.
3. Create a user-defined data type that contains the following members:

Data type:	Description:
DINT	Status
INT[x], where “x” is the output size of the data from the PLC-5C controller. (If you are consuming only one INT, no dimension is required.)	Data produced by a PLC-5C controller

4. Create a consumed tag with the following properties:

For this tag property:	Type or select:
Tag Type	Consumed
Controller	The PLC-5C that is producing the data
Remote Instance	The message number from the ControlNet configuration of the PLC-5C controller
RPI	A power of two times the NUT of the ControlNet network. For example, if the NUT is 5ms, select an RPI of 5, 10, 20, 40, etc.
Data Type	The user-defined data type that you created.

5. In RSNetWorx for ControlNet software, schedule the network.

Adjust for bandwidth limitations

When you share a tag over a ControlNet network, the tag must fit within the bandwidth of the network:

- As the number of connections increases, several connections may need to share a network update time (NUT).
- Since a ControlNet network can only pass 500 bytes in one NUT, the data of each connection must be less then 500 bytes.

Depending on the size of your system, you may not have enough bandwidth. You can make these adjustments:

- Reduce your NUT. At a faster NUT, less connections have to share an update slot.
- Increase the RPI of your connections. At higher RPIs, connections can take turns sending data during an update slot.
- For a ControlNet bridge module in a remote chassis, select the most efficient communication format for that chassis:

Are most of the modules in the chassis non-diagnostic, digital I/O modules?	Then select this communication format for the remote CNB module:
Yes	Rack Optimization
No	None

  The Rack Optimization format uses an additional 8 bytes for each slot in its chassis. Analog modules or modules that are sending or getting diagnostic, fuse, timestamp, or schedule data require direct connections and cannot take advantage of the rack optimized form. Selecting “None” frees up the 8 bytes per slot for other uses, such as produced or consumed tags.
- Separate the tag into two or more smaller tags:
  - Group the data according to similar update rates.
  - Assign a different RPI to each tag.
- Create logic to transfer the data in smaller sections (packets).

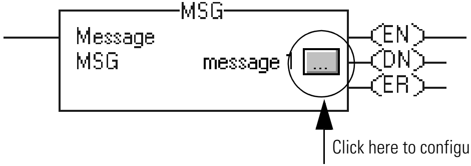
Send a Message

For each message, create a tag to control the message:

- Create the tag at the controller scope.
- Use the MESSAGE data type.
- In the Logix5000 controller, use the **DINT** data type for integers whenever possible. Logix5000 controllers execute more efficiently and use less memory when working with 32-bit integers (DINTs).
- If your message is to or from a PLC-5® or SLC 500™ controller *and* it transfers integers (not REALs), use a buffer of **INT**s:
  - Create a buffer for the data (controller scope) using the *INT*`[x]` data type.
  - Use an FAL instruction to move the data between the buffer and your application.

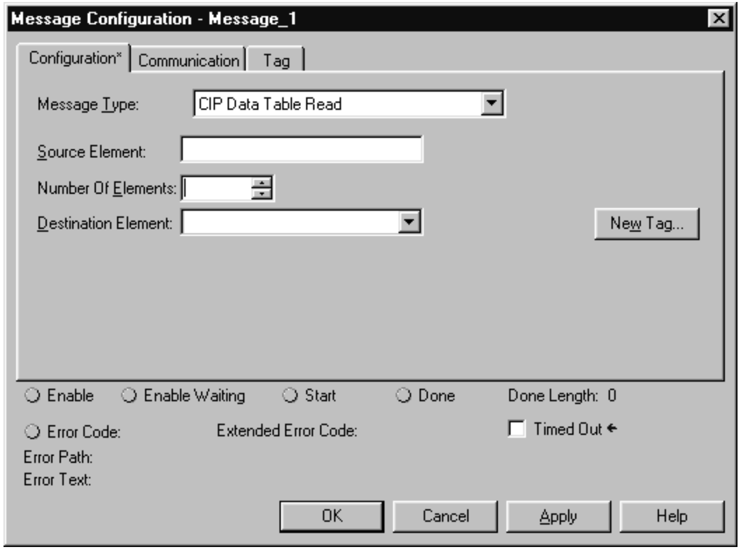
To send the same message to multiple controllers, reconfigure the MSG instruction during runtime, write new values to the members of the MESSAGE data type.

After you enter the MSG instruction and specify the MESSAGE structure, use the Message Configuration dialog box to specify the details of the message.



Click here to configure the MSG instruction

The details you configure depend on the message type you select.



Message Configuration - Message_1

Message Type: CIP Data Table Read

Done Length: 0

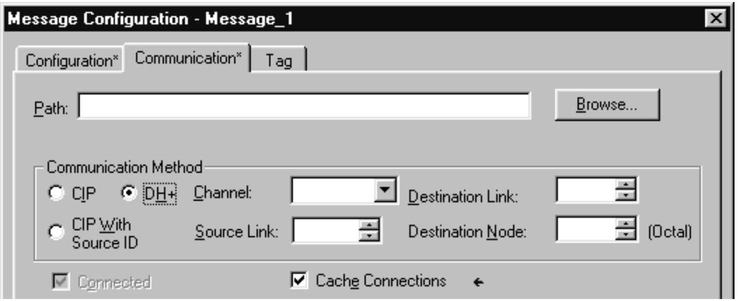
Specify the message type:

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

Then, specify this configuration information:

For this property:	Specify:
Source Element	• If you select a read message type, the Source Element is the address of the data you want to read in the target device. Use the addressing syntax of the target device. • If you select a write message type, the Source Tag is the first element of the tag that you want to send to the target device.
Number of Elements	The number of elements you read/write depends on the type of data you are using. An element refers to one "chunk" of related data. For example, tag *timer1* is one element that consists of one timer control structure.
Destination Element	• If you select a read message type, the Destination Element is the first element of the tag in the Logix5000 controller where you want to store the data you read from the target device. • If you select a write message type, the Destination Element is the address of the location in the target device where you want to write the data.

When you configure a MSG instruction, specify these details on the Communication tab.



Communication Method: ○ CIP ● DH+ ○ CIP With Source ID

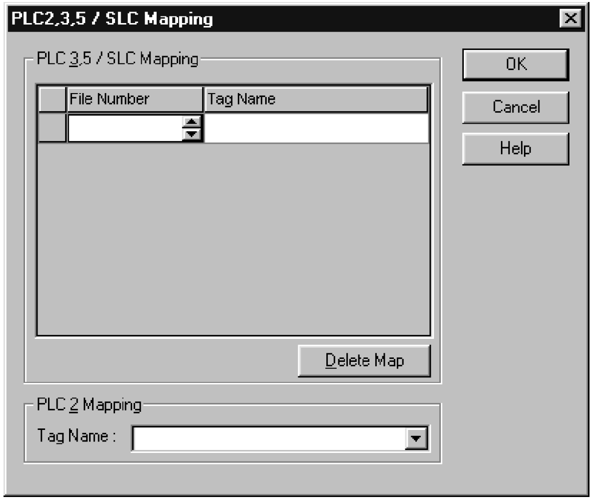
☑ Connected ☑ Cache Connections

Map PLC/SLC Addresses

You only map PLC/SLC addresses if you send a message from a PLC or SLC 500 processor to a Logix controller and the PLC/SLC processor does not support logical ASCII addressing. To use a logical address (e.g., N7:0) to specify a value (tag) in a Logix controller, you must map files to tags:

- 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).

For each file that is referenced in a PLC or SLC command, make a map entry:



- Type the file number of the logical address.
- Type or select the controller-scoped (global) tag that supplies or receives data for the file number. (You can map multiple files to the same tag.)
- For PLC-2 commands, specify the tag that supplies or receives the data.

Send a Message to Multiple Devices

To send a message to multiple devices:

- Define source and destination elements
- Create the MESSAGE_CONFIGURATION data type
- Create the configuration array
- Get the size of the local array
- Load the message properties for a device
- Configure the message
- Step to the next device

Define source and destination elements

An array stores the data that is read from or written to each remote controller. Each element in the array corresponds to a different remote device. Create the *local_array* tag, which stores the data in this controller.

Tag Name	Type
local_array	`data_type [length]` where: `data_type` is the data type of the data that the message sends or receives, such as DINT, REAL, or STRING. `length` is the number of elements in the local array.

Create the MESSAGE_CONFIGURATION data type

Create a user-defined data type to store the configuration variables for the message to each device.

- Some of the required members of the data type use a string data type.
- The default STRING data type stores 82 characters.
- If your paths or remote tag names or addresses use less than 82 characters, you have the option of creating a new string type that stores fewer characters. This lets you conserve memory.
- To create a new string type, choose *File ⇒New Component ⇒String Type…*
- If you create a new string type, use it in place of the STRING data type in this procedure.

To store the configuration variables for the message to each controller, create the following user-defined data type.

Data Type: MESSAGE_CONFIGURATION	
Name	MESSAGE_CONFIGURATION
Description	Configuration properties for a message to another controller
Members	

	Name	**Data Type**	**Style**	**Description**
	⊞ Path	STRING		
	⊞ RemoteElement	STRING		

Create the configuration array

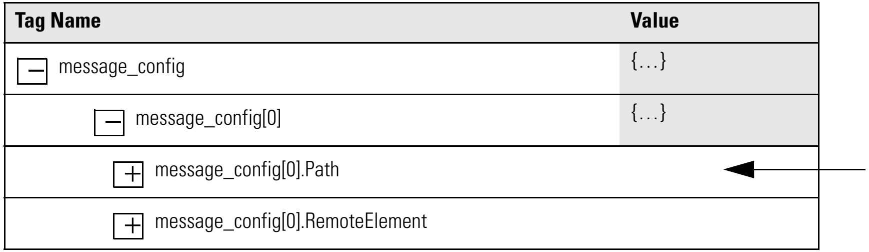
Store the configuration properties for each device in an array. Before each execution of the MSG instruction, your logic loads new properties into the instruction. This sends the message to a different controller.

1. Create this array:

Tag Name	Type	Scope
message_config	MESSAGE_CONFIGURATION[*number*]	any

where *number* is the number of devices to which to send the message.

2. Into the *message_config* array, enter the **path** to the first controller that receives the message.



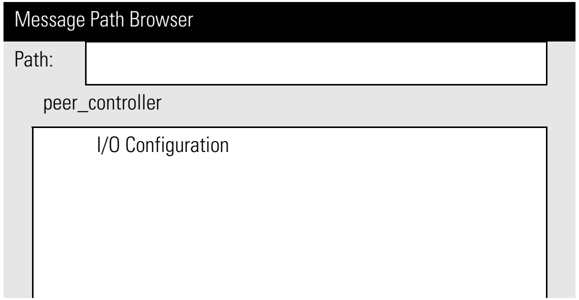
Tag Name	Value
⊟ message_config	{...}
⊟ message_config[0]	{...}
⊞ message_config[0].Path	
⊞ message_config[0].RemoteElement	

Right-click and choose *Go to Message Path Editor.*

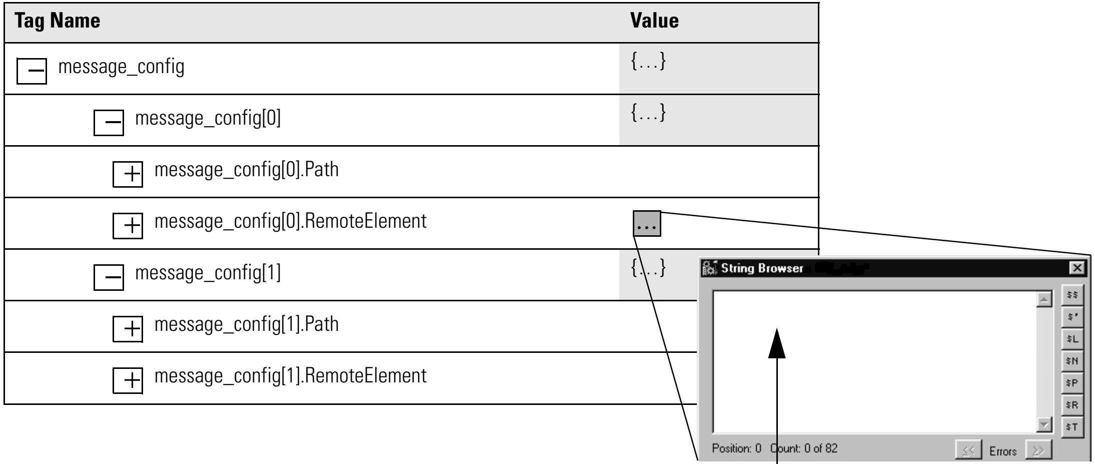
Type the **path** to the remote controller.

or

Browse to the remote controller.



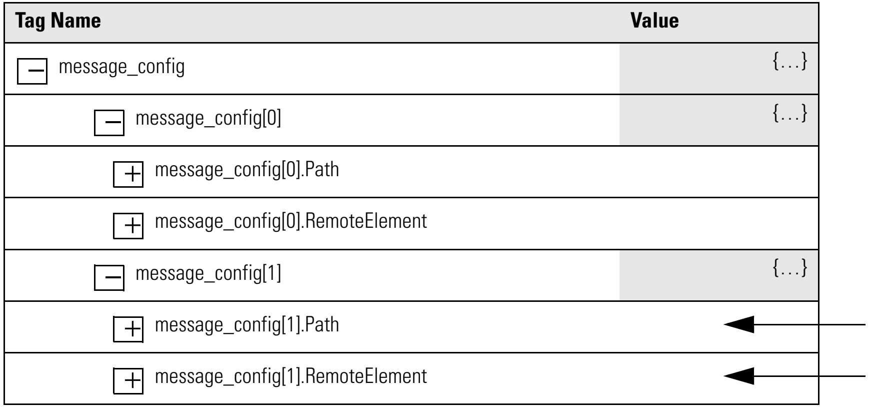
3. Into the *message_config* array, enter the tag name or address of the data in the first controller to receive the message.



Tag Name	Value
⊟ message_config	{...}
⊟ message_config[0]	{...}
⊞ message_config[0].Path	
⊞ message_config[0].RemoteElement	...
⊟ message_config[1]	{...}
⊞ message_config[1].Path	
⊞ message_config[1].RemoteElement	

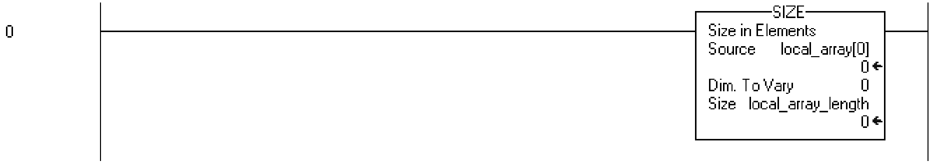
Type the tag name or address of the data in the other controller.

4. Enter the path and remote element for each additional controller:



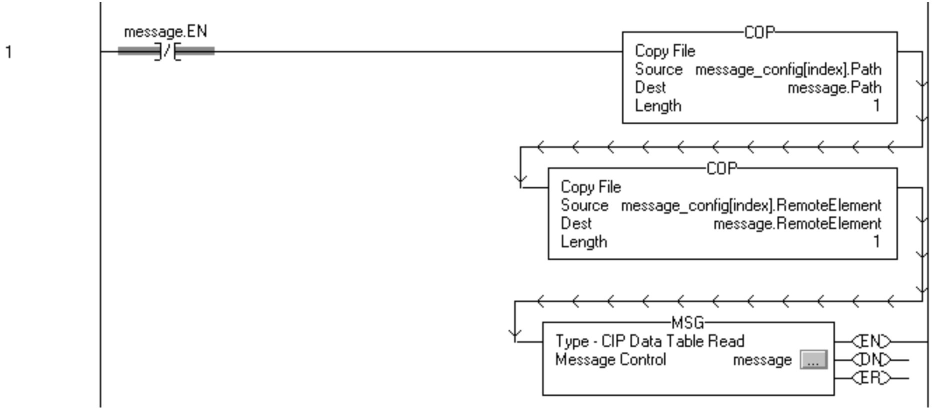
Tag Name	Value
⊟ message_config	{...}
⊟ message_config[0]	{...}
⊞ message_config[0].Path	
⊞ message_config[0].RemoteElement	
⊟ message_config[1]	{...}
⊞ message_config[1].Path	
⊞ message_config[1].RemoteElement	

Get the size of the local array



Size in Elements
Source local_array[0]
0
Dim. To Vary 0
Size local_array_length
0

Load the message properties for a device



Copy File
Source message_config[index].Path
Dest message.Path
Length 1

Copy File
Source message_config[index].RemoteElement
Dest message.RemoteElement
Length 1

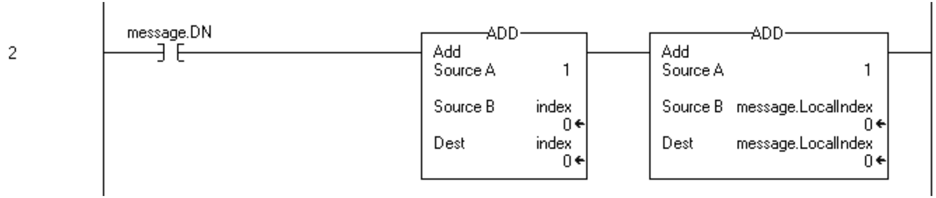
Type - CIP Data Table Read
Message Control message

Configure the message

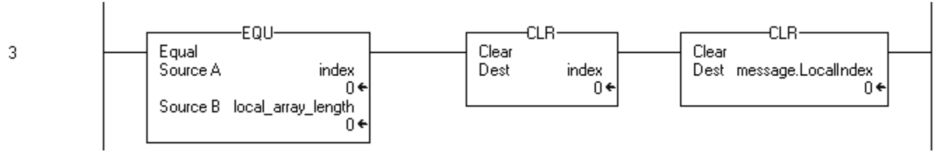
Although your logic controls the remote element and path for the message, the Message Properties dialog box requires an initial configuration. Make sure to clear the Cache Connections option.

On this tab:	**If you want to:**	**For this item:**	**Type or select:**
Configuration	read (receive) data from the other controllers	Message Type	the read-type that corresponds to the other controllers
		Source Element	tag or address that contains the data in the first controller
		Number Of Elements	1
		Destination Tag	local_array[*]
		Index	0
	write (send) data to the other controllers	Message Type	the write-type that corresponds to other controllers
		Source Tag	local_array[*]
		Index	0
		Number Of Elements	1
		Destination Element	tag or address that contains the data in the first controller
Communication	→	Path	path to the first controller
		Cache Connections	Clear the *Cache Connection* check box. Since this procedure continuously changes the path of the message, it is more efficient to clear this check box.

Step to the next controller



Restart the sequence



Notes:

Forcing

What You Can Force

Use a force to override data that your logic either uses or produces. For example, use forces in the following situations:

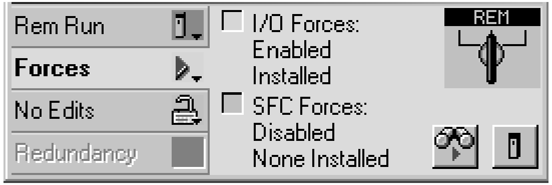
- test and debug your logic
- check wiring to an output device
- temporarily keep your process functioning when an input device has failed

Use forces only as a temporary measure. They are *not* intended to be a permanent part of your application.

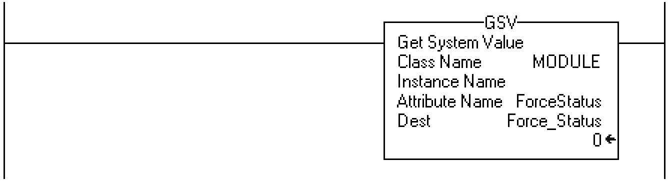
You can force the following elements:

If you want to:	Then:
override an input value, output value, produced tag, or consumed tag	add an I/O force
override the conditions of a transition one time to go from an active step to the next step	step through a transition or a force of a path
override one time the force of a simultaneous path and execute the steps of the path	
override the conditions of a transition in a sequential function chart	add an SFC force
execute some but not all the paths of a simultaneous branch of a sequential function chart	

Before you use a force, determine the status of forces for the controller:

<table>
<tr><th>Use this method:</th><th>To determine the status of:</th><th>Description:</th></tr>
<tr><td>online toolbar</td><td>I/O forces
SFC forces</td><td>Forces tab  □ I/O Forces: Enabled Installed
□ SFC Forces: Disabled None Installed</td></tr>
<tr><td>FORCE LED</td><td>I/O forces</td><td><table>
<tr><th>If the FORCE LED is:</th><th>Then:</th></tr>
<tr><td>off</td><td>• No tags contain force values.
• I/O forces are inactive (disabled).</td></tr>
<tr><td>flashing</td><td>• At least one tag contains a force value.
• I/O forces are inactive (disabled).</td></tr>
<tr><td>solid</td><td>• I/O forces are active (enabled).
• Force values may or may not exist.</td></tr>
</table></td></tr>
<tr><td>continued</td><td></td><td></td></tr>
</table>

Use this method:	To determine the status of:	Description:
GSV instruction	I/O forces	



```
GSV
Get System Value
Class Name          MODULE
Instance Name
Attribute Name  ForceStatus
Dest               Force_Status
                              0
```

Force_Status is a DINT tag.

To determine if:	Examine this bit:	For this value:
forces are installed	0	1
no forces are installed	0	0
forces are enabled	1	1
forces are disabled	1	0

Force I/O

Use an I/O force to accomplish the following:

- override an input value from another controller (i.e., a consumed tag)
- override an input value from an input device
- override your logic and specify an output value for another controller (i.e., a produced tag)
- override your logic and specify the state of an output device

IMPORTANT Forcing increases logic execution time. The more values you force, the longer it takes to execute the logic.

IMPORTANT I/O forces are held by the controller and not by the programming workstation. Forces remain even if the programming workstation is disconnected.

When you force an I/O value:

- You can force all I/O data, except for configuration data.
- If the tag is an array or structure, such as an I/O tag, force a BOOL, SINT, INT, DINT, or REAL element or member.
- If the data value is a SINT, INT, or DINT, you can force the entire value or you can force individual bits within the value.
- You can also force an alias to an I/O structure member, produced tag, or consumed tag. An alias tag shares the same data value as its base tag, so forcing an alias tag also forces the associated base tag.

Forcing an input or consumed tag:

- overrides the value regardless of the value of the physical device or produced tag
- does not affect the value received by other controllers monitoring that input or produced tag

Forcing an output or produced tag overrides the logic for the physical device or other controller (s). Other controllers monitoring that output module in a listen-only capacity will also see the forced value.

To force I/O:

1. What is the state of the I/O Forces indicator?

If:	Then note the following:
off	No I/O forces currently exist.
flashing	No I/O forces are active. But at least one force already exists in your project. When you enable I/O forces, *all* existing I/O forces will also take effect.
solid	I/O forces are enabled (active). When you install (add) a force, it immediately takes effect.

2. Open the routine that contains the tag that you want to force.

3. Right-click the tag and choose *Monitor…* If necessary, expand the tag to show the value that you want to force.

4. Install the force value:

To force a:	Do this:
BOOL value	Right-click the tag and choose *Force ON* or *Force OFF*.
non-BOOL value	In the *Force Mask* column for the tag, type the value to which you want to force the tag. Then press the *Enter* key.

5. Are I/O forces enabled? (See step 1.)

If:	Then:
no	From the *Logic* menu, choose *I/O Forcing* ⇒*Enable All I/O Forces*. Then choose *Yes* to confirm.
yes	Stop.

Step Through a Transition

To override a false transition one time and go from an active step to the next step, use the *Step Through* option. With the *Step Through* option:

- You *do not* have to add, enable, disable, or remove forces.
- The next time the SFC reaches the transition, it executes according to the conditions of the transition.

To step through the transition of an active step or a force of a simultaneous path:

1. Open the SFC routine.

2. Right-click the transition or the path that is forced and choose *Step Through*.

Force an SFC

To override the logic of an SFC, you have the following options:

If you want to:	Then:
override the conditions of a transition each time the SFC reaches the transition	Force a Transition
prevent the execution of one or more paths of a simultaneous branch	Force a Simultaneous Path

Force a Transition

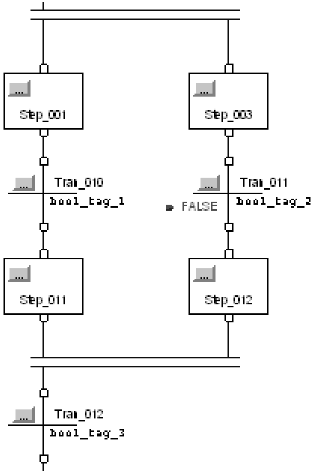
To override the conditions of a transition through repeated executions of an SFC, force the transition. The force remains until you remove it or disable forces

If you want to:	Then:
prevent the SFC from going to the next step	force the transition false
cause the SFC go to the next step regardless of transition conditions	force the transition true

If you force a transition within a simultaneous branch to be false, the SFC stays in the simultaneous branch as long as the force is active (installed and enabled).

- To leave a simultaneous branch, the last step of each path must execute at least one time and the transition below the branch must be true.
- Forcing a transition false prevents the SFC from reaching the last step of a path.

- When you remove or disable the force, the SFC can execute the rest of the steps in the path.

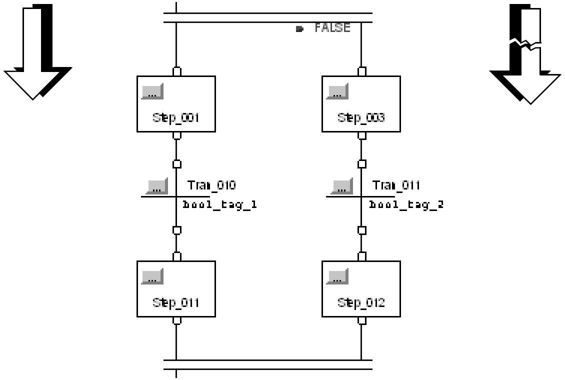


For example, to exit this branch, the SFC must be able to:

- execute *Step_011* at least once
- get past *Tran_011* and execute *Step_012* at least once
- determine that *Tran_012* is true

Force a Simultaneous Path

To prevent the execution of a path of a simultaneous branch, force the path false. When the SFC reaches the branch, it executes only the un-forced paths.



This path executes.

This path *does not* execute.

If you force a path of a simultaneous branch to be false, the SFC stays in the simultaneous branch as long as the force is active (installed and enabled).

- To leave a simultaneous branch, the last step of each path must execute at least one time and the transition below the branch must be true.
- Forcing a path false prevents the SFC from entering a path and executing its steps.
- When you remove or disable the force, the SFC can execute the steps in the path.

To force an SFC:

1. What is the state of the SFC Forces indicator?

If:	Then note the following:
off	No SFC forces currently exist.
flashing	No SFC forces are active. But at least one force already exists in your project. When you enable SFC forces, *all* existing SFC forces will also take effect.
solid	SFC forces are enabled (active). When you install (add) a force, it immediately takes effect.

2. Open the SFC routine.

3. Right-click the transition or start of a simultaneous path that you want to force, and choose either *Force TRUE* (only for a transition) or *Force FALSE*.

4. Are SFC forces enabled?

If:	Then:
no	From the *Logic* menu, choose *SFC Forcing ⇒Enable All SFC Forces*. Then choose *Yes* to confirm.
yes	Stop.

Notes:

System Faults

Controller Faults

The controller stored different fault information:

Fault type:	Description:	See page:
major fault	A fault condition that is severe enough for the controller to shut down, unless the condition is cleared. When a major fault occurs, the controller: 1. Sets a major fault bit 2. Runs user-supplied fault logic, if it exists 3. If the user-supplied fault logic cannot clear the fault, the controller goes to faulted mode 4. Sets outputs according to their output state during program mode 5. OK LED flashes red	9-2
minor fault	A fault condition that is *not* severe enough for the controller to shut down.	9-10
user-defined faults	If you want to suspend (shut down) the controller based on conditions in your application, create a user-defined major fault. With a user-defined major fault: • You define a value for the fault code. • The controller handles the fault the same as other major faults: – The controller changes to the faulted mode (major fault) and stops executing the logic. – Outputs are set to their configured state or value for faulted mode.	9-15

Major Faults

If a fault condition occurs that is severe enough for the controller to shut down, the controller generates a major fault and stops the execution of logic.

1. Create the following user-defined data type. It stores information about the fault.

Data Type: FAULTRECORD

Name	FAULTRECORD
Description	Stores the MajorFaultRecord attribute or MinorFaultRecord attribute of the PROGRAM object.
Members	

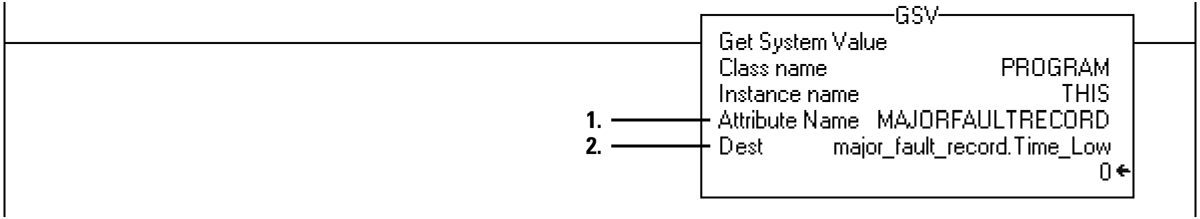
	Name	**Data Type**	**Style**	**Description**
	Time_Low	DINT	Decimal	lower 32 bits of the fault timestamp value
	Time_High	DINT	Decimal	upper 32 bits of the fault timestamp value
	Type	INT	Decimal	fault type (program, I/O, etc.)
	Code	INT	Decimal	unique code for the fault
	Info	DINT[8]	Hex	fault specific information

2. Create a fault routine to clear specific faults and let the controller resume execution. Where you place the routine depends on the type of fault that you want to clear:

For a fault due to:	**Do this:**
execution of an instruction	Create a fault routine for the program: • In the controller organizer, right-click the program and select New Routine. a. In the name box, type a name for the fault routine. b. From the Type drop-down list, select Ladder. • Right-click the program and select Properties. a. Click the Configuration tab. b. From the Fault drop-down list, select the fault routine
power loss	Create a program and main routine for the Controller Fault Handler: • In the controller organizer, right-click Controller Fault Handler and select New Program. a. Enter the name of the program and a description. • Click the + sign next to Controller Fault Handler. • Right-click the program and select the New Routine a. Enter the name of the routine and a description. b. From the Type drop-down list, select the programming language for the routine. c. Right-click the program and select Properties. d. Click the Configuration tab. e. From the Main drop-down list, select the routine
I/O	
task watchdog	
mode change	
motion axis	

3. To clear a major fault that occurs during the execution of your project, use the following logic to:

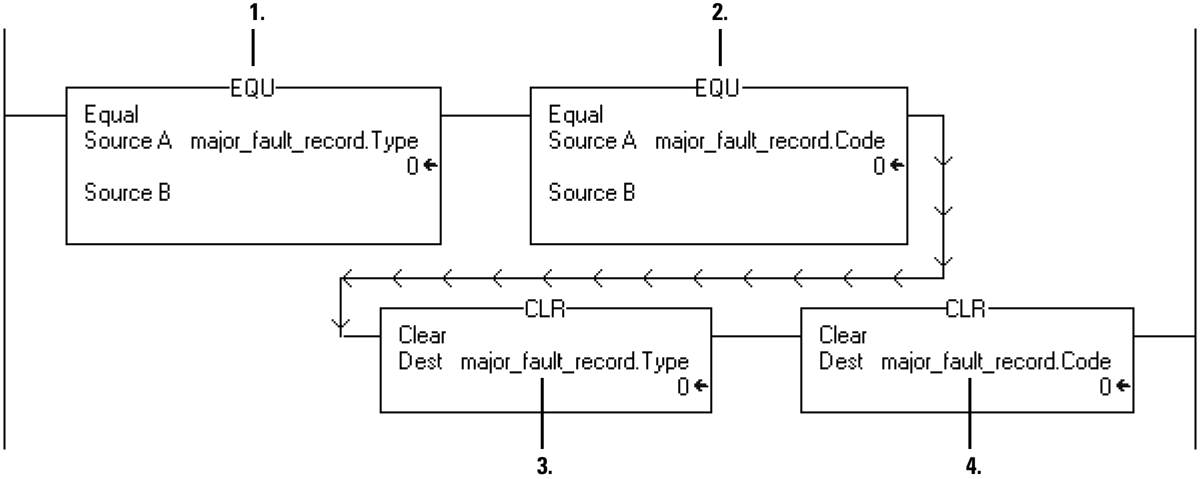
- Get the fault type and code



GSV
Get System Value
Class name PROGRAM
Instance name THIS
Attribute Name MAJORFAULTRECORD
Dest major_fault_record.Time_Low
0

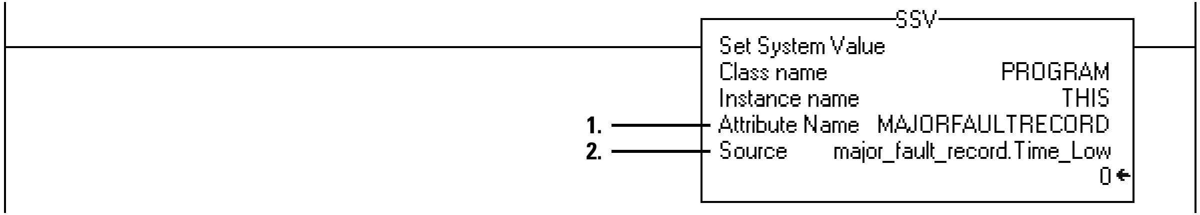
1. The GSV instruction accesses the MAJORFAULTRECORD attribute of this program.
2. The GSV instruction stores the fault information in the major_fault_record tag.

- Check for a specific fault


1.
2.
EQU
Equal
Source A major_fault_record.Type
0
Source B
EQU
Equal
Source A major_fault_record.Code
0
Source B
CLR
Clear
Dest major_fault_record.Type
0
CLR
Clear
Dest major_fault_record.Code
0
3.
4.

1. This EQU instruction checks for a specific type of fault, such as program, I/O. In Source B, enter the value for the type of fault that you want to clear.
2. This EQU instruction checks for a specific fault code. In Source B, enter the value for the code that you want to clear.
3. This CLR instruction sets to zero the value of the fault type in the major_fault_record tag.
4. This CLR instruction sets to zero the value of the fault code in the major_fault_record tag.

- Clear the fault



```
SSV
Set System Value
Class name                  PROGRAM
Instance name                  THIS
Attribute Name   MAJORFAULTRECORD
Source     major_fault_record.Time_Low
                                  0←
```

1. The SSV instruction writes new values to the MAJORFAULTRECORD attribute of this program.
2. The SSV instruction writes the values contained in the *major_fault_record* tag. Since the *Type* and *Code* member are set to zero, the fault clears and the controller resumes execution.

Major Fault Codes

Type:	Code:	Cause:	Recovery Method:
1	1	The controller powered on in Run mode.	Execute the power-loss handler.
1	60	On power-up, a non-recoverable fault occurred which resulted in loss of controller memory integrity. The controller has been reset and memory has been cleared.	Download the program to the controller. Contact Rockwell Automation for help in diagnosing the fault.
1	61	On power-up, a non-recoverable fault occurred which resulted in loss of controller memory integrity. The controller has been reset and memory has been cleared. Extended Diagnostic information was saved.	Download program to the controller. Contact Rockwell Automation for help in diagnosing the fault.
3	16	A required I/O module connection failed.	Check that the I/O module is in the chassis. Check electronic keying requirements. View the controller properties Major Fault tab and the module properties Connection tab for more information about the fault.
3	20	Possible problem with the ControlBus chassis.	Not recoverable - replace the chassis.
3	23	At least one required connection was not established before going to Run mode.	Wait for the controller I/O light to turn green before changing to Run mode.
4	16	Unknown instruction encountered.	Remove the unknown instruction. This probably happened due to a program conversion process.
4	20	Array subscript too big, control structure .POS or .LEN is invalid.	Adjust the value to be within the valid range. Don’t exceed the array size or go beyond dimensions defined.
4	21	Control structure .LEN or .POS < 0.	Adjust the value so it is > 0.
4	31	The parameters of the JSR instruction do not match those of the associated SBR or RET instruction.	Pass the appropriate number of parameters. If too many parameters are passed, the extra ones are ignored without any error.

Type:	Code:	Cause:	Recovery Method:
4	34	A timer instruction has a negative preset or accumulated value.	Fix the program to not load a negative value into timer preset or accumulated value.
4	42	JMP to a label that did not exist or was deleted.	Correct the JMP target or add the missing label.
4	82	A sequential function chart (SFC) called a subroutine and the subroutine tried to jump back to the calling SFC. Occurs when the SFC uses either a JSR or FOR instruction to call the subroutine.	Remove the jump back to the calling SFC.
4	83	The data tested was not inside the required limits.	Modify value to be within limits.
4	84	Stack overflow.	Reduce the subroutine nesting levels or the number of parameters passed.
4	89	In a SFR instruction, the target routine does not contain the target step.	Correct the SFR target or add the missing step.
4	user defined	A user-defined fault.	
6	1	Task watchdog expired. User task has not completed in specified period of time. A program error caused an infinite loop, or the program is too complex to execute as quickly as specified, or a higher priority task is keeping this task from finishing.	Increase the task watchdog, shorten the execution time, make the priority of this task “higher,” simplify higher priority tasks, or move some code to another controller.
7	40	Store to nonvolatile memory failed.	1. Try again to store the project to nonvolatile memory. 2. If the project fails to store to nonvolatile memory, replace the memory board.
7	41	Load from nonvolatile memory failed due to controller type mismatch.	Update the controller firmware to the with the correct firmware for the controller.

Type:	Code:	Cause:	Recovery Method:
7	42	Load from nonvolatile memory failed because the firmware revision of the project in nonvolatile memory does not match the firmware revision of the controller.	Update the controller firmware to the same revision level as the project that is in nonvolatile memory.
7	43	Load from nonvolatile memory failed due to bad checksum.	Contact Rockwell Automation support. See the back of this publication.
7	44	Failed to restore processor memory.	Contact Rockwell Automation support. See the back of this publication.
8	1	Attempted to place controller in Run mode with keyswitch during download.	Wait for the download to complete and clear fault.
11	1	Actual position has exceeded positive overtravel limit.	Move axis in negative direction until position is within overtravel limit and then execute Motion Axis Fault Reset.
11	2	Actual position has exceeded negative overtravel limit.	Move axis in positive direction until position is within overtravel limit and then execute Motion Axis Fault Reset.
11	3	Actual position has exceeded position error tolerance.	Move the position within tolerance and then execute Motion Axis Fault Reset.
11	4	Encoder channel A, B, or Z connection is broken.	Reconnect the encoder channel then execute Motion Axis Fault Reset.
11	5	Encoder noise event detected or the encoder signals are not in quadrature.	Fix encoder cabling then execute Motion Axis Fault Reset.
11	6	Drive Fault input was activated.	Clear Drive Fault then execute Motion Axis Fault Reset.
11	7	Synchronous connection incurred a failure.	First execute Motion Axis Fault Reset. If that doesn’t work, pull servo module out and plug back in. If all else fails replace servo module.

Type:	Code:	Cause:	Recovery Method:
11	8	Servo module has detected a serious hardware fault.	Replace the module.
11	9	Asynchronous Connection has incurred a failure.	First execute Motion Axis Fault Reset. If that doesn't work, pull servo module out and plug back in. If all else fails replace servo module.
11	32	The motion task has experienced an overlap.	The group's course update rate is too high to maintain correct operation. Clear the group fault tag, raise the group's update rate, and then clear the major fault.

Minor Faults

If a fault condition occurs that is *not* severe enough for the controller to shut down, the controller generates a **minor fault**.

- The controller continues to execute.
- You do not need to clear a minor fault.
- To optimize execution time and ensure program accuracy, you should monitor and correct minor faults.

To use ladder logic to capture information about a minor fault:

To check for a:	Do this:
periodic task overlap	1. Enter a GSV instructions that gets the FAULTLOG object, MinorFaultBits attribute. 2. Monitor bit 6.
load from nonvolatile memory	1. Enter a GSV instructions that gets the FAULTLOG object, MinorFaultBits attribute. 2. Monitor bit 7.
problem with the serial port	1. Enter a GSV instructions that gets the FAULTLOG object, MinorFaultBits attribute. 2. Monitor bit 9.
low battery	1. Enter a GSV instructions that gets the FAULTLOG object, MinorFaultBits attribute. 2. Monitor bit 10.

<table>
<tr><th>To check for a:</th><th>Do this:</th></tr>
<tr><td>problem with an instruction</td><td>
1. Create a user-defined data type that stores the fault information. Name the data type <i>FaultRecord</i> and assign the following members:
<table>
<tr><th>Name:</th><th>Data Type:</th><th>Style:</th></tr>
<tr><td>TimeLow</td><td>DINT</td><td>Decimal</td></tr>
<tr><td>TimeHigh</td><td>DINT</td><td>Decimal</td></tr>
<tr><td>Type</td><td>INT</td><td>Decimal</td></tr>
<tr><td>Code</td><td>INT</td><td>Decimal</td></tr>
<tr><td>Info</td><td>DINT[8]</td><td>Hex</td></tr>
</table>
2. Create a tag that will store the values of the MinorFaultRecord attribute.
3. Monitor S:MINOR.
4. If S:MINOR is on, use a GSV instruction to get the values of the MinorFaultRecord attribute.
5. To detect a minor fault that is caused by another instruction, reset S:MINOR. (S:MINOR remains set until the end of the scan.)
</td></tr>
</table>

Minor Fault Codes

Type:	Code:	Cause:	Recovery Method:
4	4	An arithmetic overflow occurred in an instruction.	Fix program by examining arithmetic operations (order) or adjusting values.
4	7	The GSV/SSV destination tag was too small to hold all of the data.	Fix the destination so it has enough space.
4	35	PID delta time ≤0.	Adjust the PID delta time so that it is > 0.
4	36	PID setpoint out of range	Adjust the setpoint so that it is within range.
4	51	The LEN value of the string tag is greater than the DATA size of the string tag.	1. Check that no instruction is writing to the LEN member of the string tag. 2. In the LEN value, enter the number of characters that the string contains.
4	52	The output string is larger than the destination.	Create a new string data type that is large enough for the output string. Use the new string data type as the data type for the destination.
4	53	The output number is beyond the limits of the destination data type.	Either: • Reduce the size of the ASCII value. • Use a larger data type for the destination.
4	56	The Start or Quantity value is invalid.	1. Check that the Start value is between 1 and the DATA size of the Source. 2. Check that the Start value plus the Quantity value is less than or equal to the DATA size of the Source.
4	57	The AHL instruction failed to execute because the serial port is set to no handshaking.	Either: • Change the Control Line setting of the serial port. • Delete the AHL instruction.
6	2	Periodic task overlap. Periodic task has not completed before it is time to execute again.	Simplify program(s), or lengthen period, or raise relative priority, etc.

Type:	Code:	Cause:	Recovery Method:
7	49	Project loaded from nonvolatile memory.	
9	0	Unknown error while servicing the serial port.	Contact Technical Support group.
9	1	The CTS line is not correct for the current configuration.	Disconnect and reconnect the serial port cable to the controller. Make sure the cable is wired correctly
9	2	Poll list error. A problem was detected with the DF1 master's poll list, such as specifying more stations than the size of the file, specifying more then 255 stations, trying to index past the end of the list, or polling the broadcast address (STN #255).	Check for the following errors in the poll list: • total number of stations is greater than the space in the poll list tag • total number of stations is greater than 255 • current station pointer is greater than the end of the poll list tag • a station number greater than 254 was encountered
9	5	DF1 slave poll timeout. The poll watchdog has timed out for slave. The master has not polled this controller in the specified amount of time.	Determine and correct delay for polling.
9	9	Modem contact was lost. DCD and/or DSR control lines are not being received in proper sequence and/or state.	Correct modem connection to the controller.
10	10	Battery not detected or needs to be replaced.	Install new battery.

User-Defined Faults

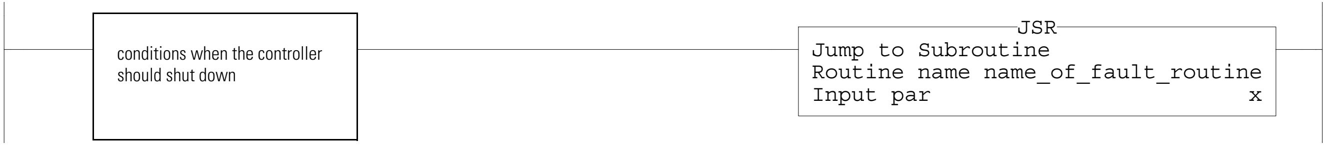
If you want to suspend (shut down) the controller based on conditions in your application, create a user-defined major fault. With a user-defined major fault:

- The fault type is always 4.
- You define a value for the fault code. Make sure it isn't a code that is already used by the predefined major faults.

  If you use a fault code that is already a predefined fault code, a major fault occurs.

- The controller handles the fault the same as other major faults:
  - The controller changes to the faulted mode (major fault) and stops executing the logic.
  - Outputs are set to their configured state or value for faulted mode.

In the main routine of the program, enter the following rung:


conditions when the controller should shut down
JSR
Jump to Subroutine
Routine name name_of_fault_routine
Input par x

Data Structures

Chapter **10**

Common Structures

The following structures are common structures used by several relay ladder instructions. Function block instructions also use structures, but they are more unique to individual types of instructions.

COMPARE Structure

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates that the instruction is enabled.
.DN	BOOL	The done bit is set when the instruction has operated on the last element (.POS = .LEN).
.FD	BOOL	The found bit is set each time the instruction records a mismatch (one-at-a-time operation) or after recording all mismatches (all-per-scan operation).
.IN	BOOL	The inhibit bit indicates the search mode. 0 = all mode 1 = one mismatch at a time mode
.ER	BOOL	The error bit is set if .POS < 0 or .LEN < 0. The instruction stops executing until the program clears the .ER bit.
.LEN	DINT	The length specifies the number of elements in the array.
.POS	DINT	The position contains the position of the current element.

CONTROL Structure

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates that the instruction is enabled.
.DN	BOOL	The done bit is set when the instruction has operated on the last element (.POS = .LEN).
.ER	BOOL	The error bit is set if the expression generates an overflow (S:V is set). The instruction stops executing until the program clears the .ER bit. The .POS value contains the position of the element that caused the overflow.
.LEN	DINT	The length specifies the number of elements in the array.
.POS	DINT	The position contains the position of the current element.

COUNTER Structure

Mnemonic:	Data Type:	Description:
.CD	BOOL	The count down enable bit indicates that the CTD instruction is enabled.
.CU	BOOL	The count up enable bit indicates that the CTU instruction is enabled.
.DN	BOOL	The done bit indicates that .ACC ≥ .PRE.
.OV	BOOL	The overflow bit indicates that the counter exceeded the upper limit of 2,147,483,647. The counter then rolls over to -2,147,483,648 and begins counting up again.
.UN	BOOL	The underflow bit indicates that the counter exceeded the lower limit of -2,147,483,648. The counter then rolls over to 2,147,483,647 and begins counting down again.
.PRE	DINT	The preset value specifies the value which the accumulated value must reach before the instruction sets the .DN bit.
.ACC	DINT	The accumulated value specifies the number of transitions the instruction has counted.

EXT_ROUTINE_CONTROL Structure *(SoftLogix5800 controller only)*

Mnemonic:	Data Type:	Description:
ErrorCode	SINT	If an error occurs, this value identifies the error. Valid values are from 0-255.
NumParams	SINT	This value indicates the number of parameters associated with this instruction.
ParameterDefs	EXT_ROUTINE_ PARAMETERS[10]	This array contains definitions of the parameters to pass to the external routine. The instruction can pass as many as 10 parameters.

Mnemonic:	**Data Type:**	**Description:**
ReturnParamDef	EXT_ROUTIN_ PARAMETERS	This value contains definitions of the return parameter from the external routine. There is only one return parameter.
EN	BOOL	When set, the enable bit indicates that the JXR instruction is enabled.
ReturnsValue	BOOL	If set, this bit indicates that a return parameter was entered for the instruction. If cleared, this bit indicates that no return parameter was entered for the instruction.
DN	BOOL	The done bit is set when the external routine has executed once to completion.
ER	BOOL	The error bit is set if an error occurs. The instruction stops executing until the program clears the error bit.
FirstScan	BOOL	This bit identifies whether this is the first scan after switching the controller to Run mode. Use FirstScan to initialize the external routine, if needed.
EnableOut	BOOL	Enable output.
EnableIn	BOOL	Enable input.
User1	BOOL	These bits are available for the user. The controller does not initialize these bits.
User0	BOOL	
ScanType1	BOOL	These bits identify the current scan type: **Bit Values:** **Scan Type:** 00 Normal 01 Pre Scan 10 Post Scan (not applicable to relay ladder programs)
ScanType0	BOOL	

MESSAGE Structure

Mnemonic:	**Data Type:**	**Description:**	
.FLAGS	INT	The .FLAGS member provides access to the status members (bits) in one, 16-bit word.	
		This bit:	**Is this member:**
		2	.EW
		4	.ER
		5	.DN
		6	.ST
		7	.EN
		8	.TO
		9	.EN_CC
		Important: Resetting any MSG status bits while a MSG is enabled can disrupt communications.	
.ERR	INT	If the .ER bit is set, the error code word identifies error codes for the MSG instruction.	
.EXERR	INT	The extended error code word specifies additional error code information for some error codes.	
.REQ_LEN	INT	The requested length specifies how many words the message instruction will attempt to transfer.	
.DN_LEN	INT	The done length identifies how many words actually transferred.	
.EW	BOOL	The enable waiting bit is set when the controller detects that a message request has entered the queue. The controller resets the.EW bit when the .ST bit is set.	

Mnemonic:	Data Type:	Description:
.ER	BOOL	The error bit is set when the controller detects that a transfer failed. The .ER bit is reset the next time the rung-condition-in goes from false to true.
.DN	BOOL	The done bit is set when the last packet of the message is successfully transferred. The .DN bit is reset the next time the rung-condition-in goes from false to true.
.ST	BOOL	The start bit is set when the controller begins executing the MSG instruction. The .ST bit is reset when the .DN bit or the .ER bit is set.
.EN	BOOL	The enable bit is set when the rung-condition-in goes true and remains set until either the .DN bit or the .ER bit is set and the rung-condition-in is false. If the rung-condition-in goes false, but the .DN bit and the .ER bit are cleared, the .EN bit remains set.
.TO	BOOL	If you manually set the .TO bit, the controller stops processing the message and sets the .ER bit.
.EN_CC	BOOL	The enable cache bit determines how to manage the MSG connection. Connections for MSG instructions going out the serial port are not cached, even if the .EN_CC bit is set.
.ERR_SRC	SINT	Used by RSLogix 5000 software to show the error path on the Message Configuration dialog box
.DestinationLink	INT	To change the Destination Link of a DH+ or CIP with Source ID message, set this member to the required value.
.DestinationNode	INT	To change the Destination Node of a DH+ or CIP with Source ID message, set this member to the required value.
.SourceLink	INT	To change the Source Link of a DH+ or CIP with Source ID message, set this member to the required value.
.Class	INT	To change the Class parameter of a CIP Generic message, set this member to the required value.
.Attribute	INT	To change the Attribute parameter of a CIP Generic message, set this member to the required value.
.Instance	DINT	To change the Instance parameter of a CIP Generic message, set this member to the required value.

Mnemonic:	Data Type:	Description:	
.LocalIndex	DINT	If you use an asterisk [*] to designate the element number of the local array, the LocalIndex provides the element number. To change the element number, set this member to the required value.	
		If the message:	**Then the local array is the:**
		reads data	Destination element
		writes data	Source element
.Channel	SINT	To send the message out a different channel of the 1756-DHRIO module, set this member to the required value. Use either the ASCII character A or B.	
.Rack	SINT	To change the rack number for a block transfer message, set this member to the required rack number (octal).	
.Group	SINT	To change the group number for a block transfer message, set this member to the required group number (octal).	
.Slot	SINT	To change the slot number for a block transfer message, set this member to the required slot number.	
		If the network is:	**Then specify the slot number in:**
		universal remote I/O	octal
		ControlNet	decimal (0-15)
.Path	STRING	To send the message to a different controller, set this member to the new path. • enter the path as hexadecimal values • omit commas [,]	

<table>
<tr><th>Mnemonic:</th><th>Data Type:</th><th colspan="2">Description:</th></tr>
<tr><td rowspan="4">.RemoteIndex</td><td rowspan="4">DINT</td><td colspan="2">If you use an asterisk [*] to designate the element number of the remote array, the RemoteIndex provides the element number. To change the element number, set this member to the required value.</td></tr>
<tr><td><b>If the message:</b></td><td><b>Then the remote array is the:</b></td></tr>
<tr><td>reads data</td><td>Source element</td></tr>
<tr><td>writes data</td><td>Destination element</td></tr>
<tr><td rowspan="4">.RemoteElement</td><td rowspan="4">STRING</td><td colspan="2">To specify a different tag or address in the controller to which the message is sent, set this member to the required value. Enter the tag or address as ASCII characters.</td></tr>
<tr><td><b>If the message:</b></td><td><b>Then the remote array is the:</b></td></tr>
<tr><td>reads data</td><td>Source element</td></tr>
<tr><td>writes data</td><td>Destination element</td></tr>
<tr><td>.UnconnnectedTimeout</td><td>DINT</td><td colspan="2">The time out for unconnected messages. The default value is 30 seconds.</td></tr>
<tr><td>.ConnectionRate</td><td>DINT</td><td colspan="2" rowspan="2">The ConnectionRate times the TimeoutMultiplier produces the time out for connected messages.
• the default ConnectionRate is 7.5 seconds
• the default TimeoutMultiplier is 0 (which equates to a multiplication factor of 4)
• the default time out for connected messages is 30 seconds (7.5 seconds x 4 = 30 seconds)
• to change the time out, change the ConnectionRate and leave the TimeoutMultiplier at the default value</td></tr>
<tr><td>.TimeoutMultiplier</td><td>SINT</td></tr>
</table>

RESULT Structure

Mnemonic:	Data Type:	Description:
.DN	BOOL	The done bit is set when the Result array is full.
.LEN	DINT	The length value identifies the number of storage locations in the Result array.
.POS	DINT	The position value identifies the current position in the Result array.

SERIAL_PORT_CONTROL Structure

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates that the instruction is enabled.
.EU	BOOL	The queue bit indicates that the instruction entered the ASCII queue.
.DN	BOOL	The done bit indicates when the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates that the instruction is executing.
.EM	BOOL	The empty bit indicates that the instruction is done, but it is synchronous to the logic scan.
.ER	BOOL	The error bit indicates when the instruction fails (errors).
.FD	BOOL	The found bit indicates that the instruction found the termination character or characters.
.POS	DINT	The position determines the number of characters in the buffer, up to and including the first set of termination characters. The instruction only returns this number after it finds the termination character or characters.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

STRING Structure

Every string data type includes these members:

Name:	Data Type:	Description:	Notes:
LEN	DINT	number of characters in the string	The LEN automatically updates to the new count of characters whenever you: • use the String Browser dialog box to enter characters • use instructions that read, convert, or manipulate a string The LEN shows the length of the current string. The DATA member may contain additional, old characters, which are not included in the LEN count.
DATA	SINT array	ASCII characters of the string	To access the characters of the string, address the name of the tag. Each element of the DATA array contains one character. You can create new string data types that store less or more characters.

You store ASCII characters in tags that use a string data type.

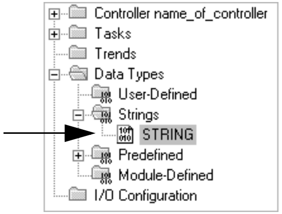
- You can use the default STRING data type. It stores up to 82 characters.
- You can create a new string data type that stores less or more characters.

IMPORTANT Use caution when you create a new string data type. If you later decide to change the size of the string data type, you may lose data in any tags that currently use that data type.

If you:	Then:
make a string data type smaller	• The data is truncated. • The LEN is unchanged.
make a string data type larger	The data and LEN is reset to zero.

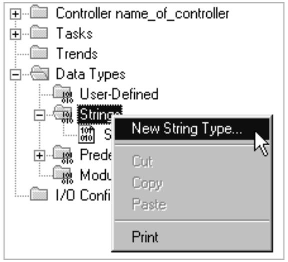
To create a string data type:

Use the default STRING data type. It stores as many as 82 characters.

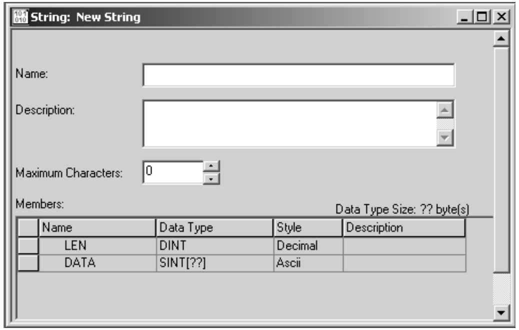


OR

Create a new string data type to store the number of characters that you define.



If you create a new string data type, define the number of characters in the string:


String: New String

Name:

Description:

Maximum Characters: 0

Members:

Data Type Size: ?? byte(s)

Name	Data Type	Style	Description
LEN	DINT	Decimal	
DATA	SINT[??]	Ascii	

TIMER Structure

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates that the instruction is enabled.
.TT	BOOL	The timing bit indicates that a timing operation is in process
.DN	BOOL	The done bit is set when .ACC ≥ .PRE.
.PRE	DINT	The preset value specifies the value (1 msec units) which the accumulated value must reach before the instruction sets the .DN bit.
.ACC	DINT	The accumulated value specifies the number of milliseconds that have elapsed since the instruction was enabled.

User-Defined Structure

You can also create your own structures, called a user-defined data type. A user-defined data type groups different types of data into a single named entity.

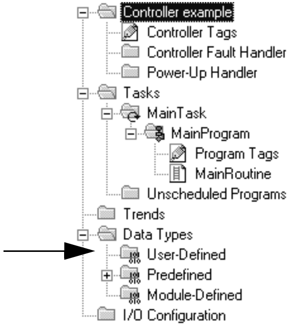
- Within a user-defined data type, you define the members.
- Like tags, members have a name and data type.
- You can include arrays and structures.
- Once you create a user-defined data type, you can create one or more tags using that data type.
- Minimize the use of these data type because they typically increase the memory requirements and execution time of your logic:
  - INT
  - SINT

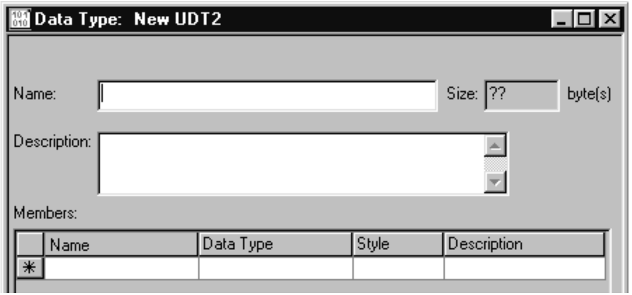
- If you include members that represent I/O devices, you must use ladder logic to copy the data between the members in the structure and the corresponding I/O tags.
- When you use the BOOL, SINT, or INT data types, place members that use the same data type in sequence:

more efficient	**less efficient**
BOOL	BOOL
BOOL	DINT
BOOL	BOOL
DINT	DINT
DINT	BOOL

- You can use single dimension arrays.
- You can create, edit, and delete user-defined data types only when programming offline.
- If you modify a user-defined data type and change its size, the existing values of any tags that use the data type are set to zero (0).
- To copy data to a structure, use the COP instruction.

To create a user-defined data type:





Name:

Size: ?? byte(s)

Description:

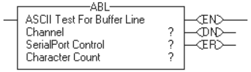
Members:

	Name	Data Type	Style	Description
*				

Notes:

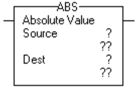
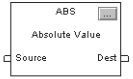
Instruction Set

Chapter 11

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ABL **ASCII Test for Buffer Line**	 ABL ASCII Test For Buffer Line Channel ? SerialPort Control ? Character Count ? EN DN ER	not available	`ABL(Channel SerialPortControl);`	The ABL instruction counts the characters in the buffer up to and including the first termination character.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	displays the number of characters in the buffer, including the first set of termination characters (relay ladder only)

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**
ABS **Absolute Value**			`dest := ABS(source);`	The ABS instruction takes the absolute value of the Source and places the result in the Destination.

Relay Ladder and Structured Text

Operand:	**Type:**	**Format:**	**Description:**
Source	SINT DINT INT REAL	immediate tag	value of which to take the absolute value
Destination	SINT DINT INT REAL	tag	tag to store the result

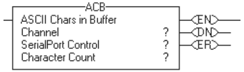
Function Block

Operand:	**Type:**	**Format:**	**Description:**
ABS tag	FBD_MATH_ ADVANCED	structure	ABS structure (default parameters):

Parameter:	**Type:**	**Description:**
Source	REAL	value of which to take the absolute value
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	**Major Faults:**
affected	none

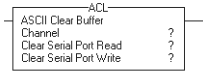
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

**ACB
ASCII
Characters in
Buffer**
	 ACB
	ASCII Chars in Buffer
	Channel ?
	SerialPort Control ?
	Character Count ?
	EN
	DN
	ER
		not available	`ACB(Channel`
			`SerialPortControl)`
				The ACB instruction counts the characters in the buffer.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	displays the number of characters in the buffer (relay ladder only)

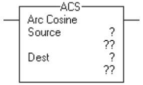
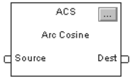
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

**ACL
ASCII Clear Buffer**
	 ACL
	ASCII Clear Buffer
	Channel ?
	Clear Serial Port Read ?
	Clear Serial Port Write ?
		not available	`ACL(Channel,`
			`ClearSerialPortRead,`
			`ClearSerialPortWrite);`
				The ACL instruction immediately clears the buffer and ASCII queue.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Clear Serial Port Read	BOOL	immediate tag	to empty the buffer and remove ARD and ARL instructions from the queue, enter Yes.
Clear Serial Port Write	BOOL	immediate tag	to remove AWA and AWT instructions from the queue, enter Yes.

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ACS **Arc Cosine**			`dest := ACOS(source);`	The ACS instruction takes the arc cosine of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

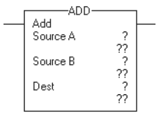
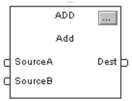
Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	find the arc cosine of this value
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
ACS tag	FBD_MATH_ ADVANCED	structure	ACS structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ADD **Add**	 ADD Add Source A ? ?? Source B ? ?? Dest ? ??		`dest := sourceA + sourceB;`	The ADD instruction adds Source A to Source B and places the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source A	SINT INT	DINT REAL	immediate tag	value to add to Source B
Source B	SINT INT	DINT REAL	immediate tag	value to add to Source A
Destination	SINT INT	DINT REAL	tag	tag to store the result

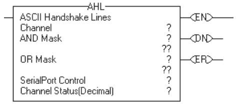
Function Block

Operand:	Type:	Format:	Description:
ADD tag	FBD_MATH	structure	ADD structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to add to SourceB
SourceB	REAL	value to add to SourceA
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

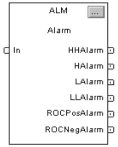
Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**
AFI **Always False**	—[AFI]—	not available	not available	The AFI instruction sets its rung-condition-out to false.
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
AHL ASCII Handshake Lines	 AHL ASCII Handshake Lines Channel ? AND Mask ? ?? OR Mask ? ?? SerialPort Control ? Channel Status(Decimal) ? EN DN ER	not available	`AHL(Channel,ANDMask ORMask, SerialPortControl);`	The AHL instruction obtains the status of control lines and turns on or off the DTR and RTS signals.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
ANDMask	DINT	immediate tag	
ORMask	DINT	immediate tag	
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Channel Status	DINT	immediate	displays the status of the control lines (relay ladder only)

To turn DTR:	And turn RTS:	ANDMask value:	ORMask value:	To turn DTR:	And turn RTS:	ANDMask value:	ORMask value:
off	off	3	0	unchanged	off	2	0
	on	1	2		on	0	2
	unchanged	1	0		unchanged	0	0
on	off	2	1				
	on	0	3				
	unchanged	0	1				

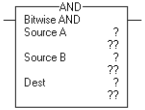
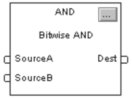
Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 57	The AHL instruction failed to execute because the serial port is set to no handshaking. Either change the Control Line setting of the serial port or delete the AHL instruction.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ALM **Alarm**	not available	 ALM ... Alarm In HHAlarm HAlarm LAlarm LLAlarm ROCPosAlarm ROCNegAlarm	`ALM(ALM_tag);`	The ALM instruction provides alarming for any analog signal.

Operand:	Type:	Format:	Description:
ALM tag	ALARM	structure	ALM structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input
HHAlarm	BOOL	high-high alarm indicator
HAlarm	BOOL	high alarm indicator
LAlarm	BOOL	low alarm indicator
LLAlarm	BOOL	low-low alarm indicator
ROCPosAlarm	BOOL	rate-of-change positive alarm indicator
ROCNegAlarm	BOOL	rate-of-change negative alarm indicator

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
AND **Bitwise AND**	 AND Bitwise AND Source A ? ?? Source B ? ?? Dest ? ??		`dest := sourceA AND sourceB`	The AND instruction performs a bitwise AND operation using the bits in Source A and Source B and places the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source A	SINT INT	DINT	immediate tag	value to AND with Source B
Source B	SINT INT	DINT	immediate tag	value to AND with Source A
Destination	SINT INT	DINT	tag	tag to store the result

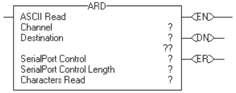
Function Block

Operand:	Type:	Format:	Description:
AND tag	FBD_LOGICAL	structure	AND structure (default parameters):

Parameter:	Type:	Description:
SourceA	DINT	value to AND with Source B
SourceB	DINT	value to AND with Source A
Dest	DINT	result of the instruction

Arithmetic Status Flags:	Major Faults:
affected	none

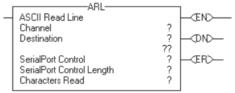
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

ARD
ASCII Read
	 ARD
	ASCII Read
	Channel ?
	Destination ?
	??
	SerialPort Control ?
	SerialPort Control Length ?
	Characters Read ?
	EN
	DN
	ER
		not available	`ARD(Channel,`
			`Destination,`
			`SerialPortControl);`
				The ARD instruction removes characters from the buffer and stores them in the Destination.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Destination	string SINT DINT INT	tag	tag into which the characters are moved (read): • for a string data type, enter the name of the tag • for a SINT, INT, or DINT array, enter the first element of the array
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Serial Port Control Length	DINT	immediate	displays the number of characters to move to the destination (relay ladder only)
Characters Read	DINT	immediate	during execution, displays the number of characters that were read (relay ladder only)

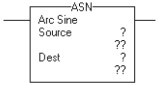
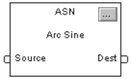
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

ARL
ASCII Read Line
	 ARL
	ASCII Read Line
	Channel ?
	Destination ?
	??
	SerialPort Control ?
	SerialPort Control Length ?
	Characters Read ?
	EN
	DN
	ER
		not available	`ARL(Channel,`
			`Destination,`
			`SerialPortControl);`
				The ARL instruction removes specified characters from the buffer and stores them in the Destination.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Destination	string SINT DINT INT	tag	tag into which the characters are moved (read): • for a string data type, enter the name of the tag • for a SINT, INT, or DINT array, enter the first element of the array
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Serial Port Control Length	DINT	immediate	displays the maximum number of characters to read if no termination characters are found (relay ladder only)
Characters Read	DINT	immediate	during execution, displays the number of characters that were read (relay ladder only)

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ASN **Arc Sine**			`dest := ASIN(source);`	The ASN instruction takes the arc sine of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

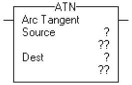
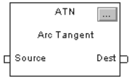
Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	find the arc sine of this value
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
ASN tag	FBD_MATH_ADVANCED	structure	ASN structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ATN **Arc Tangent**			`dest := ATAN(source);`	The ATN instruction takes the arc tangent of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	find the arc tangent of this value
Destination	SINT INT	DINT REAL	tag	tag to store the result

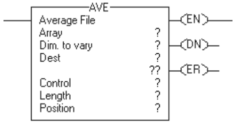
Function Block

Operand:	Type:	Format:	Description:
ATN tag	FBD_MATH_ ADVANCED	structure	ATN structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

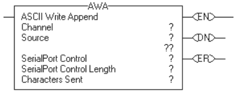
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

AVE
Average
	 AVE
	Average File
	Array ?
	Dim. to vary ?
	Dest ?
	??
	Control ?
	Length ?
	Position ?
	EN
	DN
	ER
		not available	`SIZE(array,0,length);`
			`sum := 0;`
			`FOR position = 0 TO length-1 DO`
			`    sum := sum + array[position];`
			`END_FOR;`
			`destination := sum / length;`
				The AVE instruction calculates the average of a set of values.

Operand:	Type:	Format:	Description:
Array	SINT DINT INT REAL	array tag	find the average of the values in this array; specify the first element of the group of elements to average **do not** use CONTROL.POS in the subscript
Dimension to vary	DINT	immediate (0, 1, 2)	which dimension to use the order is: array[dim_0,dim_1,dim_2] then array[dim_0,dim_1] then array[dim_0]
Destination	SINT DINT INT REAL	tag	result of the operation
Control	CONTROL	tag	control structure for the operation
Length	DINT	immediate	number of elements of the array to average
Position	DINT	immediate	current element in the array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	Dimension to vary does not exist for the specified array
	Type 4	Code 21	.POS < 0 or .LEN < 0

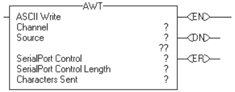
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

AWA
ASCII Write Append
		not available	`AWA(Channel,Source,`
			`SerialPortControl);`
				The AWA instruction sends a specified number of characters of the Source tag to a serial device and appends either one or two predefined characters.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Source	string SINT DINT INT	tag	tag that contains the characters to send: • for a string data type, enter the name of the tag. • for a SINT, INT, or DINT array, enter the first element of the array.
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Serial Port Control Length	DINT	immediate	displays the number of characters to send (relay ladder only)
Characters Sent	DINT	immediate	displays the number of characters that were sent (relay ladder only)

Arithmetic Status Flags:	Major Faults:
not affected	none

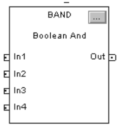
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

AWT
ASCII Write
	 AWT
	ASCII Write
	Channel ?
	Source ?
	??
	SerialPort Control ?
	SerialPort Control Length ?
	Characters Sent ?
	EN
	DN
	ER
		not available	`AWT(Channel,`
			`Source,`
			`SerialPortControl);`
				The AWT instruction sends a specified number of characters of the Source tag to a serial device.

Operand:	Type:	Format:	Description:
Channel	DINT	immediate tag	0
Source	SINT DINT INT string	tag	tag that contains the characters to send: • for a string data type, enter the name of the tag • for a SINT, INT, or DINT array, enter the first element of the array
Serial Port Control	SERIAL_PORT_ CONTROL	tag	tag that controls the operation
Serial Port Control Length	DINT	immediate	number of characters to send (relay ladder only)
Characters Sent	DINT	immediate	displays the number of characters that were sent (relay ladder only)

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

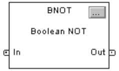
BAND
Boolean AND
	see AND	 BAND
		...
		Boolean And
		In1
		In2
		In3
		In4
		Out
			`IF operandA AND operandB THEN`
			`<statement>;`
			`END_IF;`
				The BAND instruction logically ANDs as many as 8 boolean inputs.

Operand:	Type:	Format:	Description:
BAND tag	FBD_BOOLEAN_AND	structure	BAND structure (default parameters):

Parameter:	Type:	Description:
In*x*	BOOL	boolean input; where *x* = 1-8
Out	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

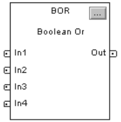
BNOT
Boolean NOT
	see NOT		`IF NOT operand THEN`
			`     <statement>;`
			`END_IF;`
				The BNOT instruction complements a boolean input.

Operand:	Type:	Format:	Description:
BNOT tag	FBD_BOOLEAN_BNOT	structure	BNOT structure (default parameters):

Parameter:	Type:	Description:
In	BOOL	boolean input
Out	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

BOR
Boolean OR
	see OR		`IF operandA OR operandB THEN`
			`    <statement>;`
			`END_IF;`
				The BOR instruction logically ORs as many as 8 boolean inputs.

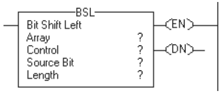
Operand:	Type:	Format:	Description:
BOR tag	FBD_BOOLEAN_OR	structure	BOR structure (default parameters):

Parameter:	Type:	Description:
In*x*	BOOL	boolean input; where *x* = 1-8
Out	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

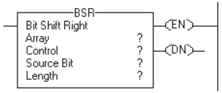
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
BRK **Break**	—(BRK)—	not available	`EXIT;`	The BRK instruction interrupts the execution of a routine that was called by a FOR instruction.

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
BSL **Bit Shift Left**		not available	not available	The BSL instruction shifts the specified bits within the Array one position left.

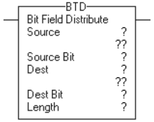
Operand:	Type:	Format:	Description:
Array	DINT	array tag	array to modify; specify the first element of the group of elements **do not** use CONTROL.POS in the subscript
Control	CONTROL	tag	control structure for the operation
Source bit	BOOL	tag	bit to shift
Length	DINT	immediate	number of bits in the array to shift

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
BSL **Bit Shift Right**	 BSR Bit Shift Right Array ? Control ? Source Bit ? Length ? EN DN	not available	not available	The BSR instruction shifts the specified bits within the Array one position right.

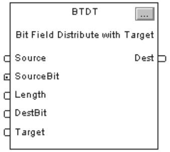
Operand:	Type:	Format:	Description:
Array	DINT	array tag	array to modify; specify the first element of the group of elements **do not** use CONTROL.POS in the subscript
Control	CONTROL	tag	control structure for the operation
Source bit	BOOL	tag	bit to shift
Length	DINT	immediate	number of bits in the array to shift

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
BTD **Bit Field** **Distribute**	 BTD Bit Field Distribute Source ? ?? Source Bit ? Dest ? ?? Dest Bit ? Length ?	see BTDT	see BTDT	The BTD instruction copies the specified bits from the Source, shifts the bits to the appropriate position, and writes the bits into the Destination.

Operand:	Type:	Format:	Description:
Source	SINT DINT INT	immediate tag	tag that contains the bits to move
Source Bit	DINT	immediate	number of the bit (lowest bit number) from where to start the move must be within the valid range for the Source data type (0-31 DINT, 0-15 INT, 0-7 SINT)
Destination	SINT DINT INT	immediate tag	tag where to move the bits
Destination bit	DINT	immediate	the number of the bit (lowest bit number) where to start copying bits from the Source must be within the valid range for the Destination data type (0-31 DINT, 0-15 INT, 0-7 SINT)
Length	DINT	tag	number of bits to move (1-32)

Arithmetic Status Flags:	Major Faults:
affected	none

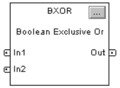
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
BTDT **Bit Field Distribute with Target**	see BTD	 BTDT ... Bit Field Distribute with Target Source SourceBit Length DestBit Target Dest	`BTDT(BTDT_tag);`	The BTDT instruction first copies the Target to the Destination. Then the instruction copies the specified bits from the Source, shifts the bits to the appropriate position, and writes the bits into the Destination. The Target and Source remain unchanged.

Operand:	Type:	Format:	Description:
BTDT tag	FBD_BIT_FIELD_ DISTRIBUTE	structure	BTDT structure (default parameters):

Parameter:	Type:	Description:
Source	DINT	input value containing the bits to move to Destination
SourceBit	DINT	the bit position in Source (lowest bit number where to start the move)
Length	DINT	number of bits to move (1-32)
DestBit	DINT	the bit position in Dest (lowest bit number to start copying bits into)
Target	DINT	input value to move to Dest prior to moving bits from the Source
Dest	DINT	result of the bit move operation

Arithmetic Status Flags:	Major Faults:
affected	none

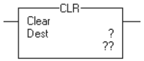
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

BXOR
Boolean
Exclusive XOR
	see XOR		`IF operandA XOR operandB`
			`THEN`
			`     <statement>;`
			`END_IF;`
				The BXOR performs an exclusive OR on two boolean inputs.

Operand:	Type:	Format:	Description:
BXOR tag	FBD_BOOLEAN_XOR	structure	BXOR structure (default parameters):

Parameter:	Type:	Description:
In1	BOOL	boolean input
In2	BOOL	boolean input
Out	BOOL	result of the instruction

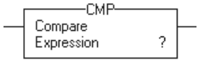
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CLR **Clear**		not available	`dest := 0;`	The CLR instruction clears all the bits of the Destination.

Operand:	Type:	Format:	Description:
Destination	SINT INT DINT REAL	tag	tag to clear

Arithmetic Status Flags:	Major Faults:
affected	none

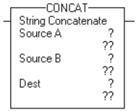
Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**

CMP
Compare
		not available	`IF BOOL_expression THEN`
			`<statement>;`
			`END_IF;`
				The CMP instruction performs a comparison on the arithmetic operations you specify in the expression.

Operand:	**Type:**	**Format:**	**Description:**
Expression	SINT INT DINT REAL string	immediate tag	an expression consisting of tags and/or immediate values separated by operators

Arithmetic Status Flags:	**Major Faults:**
affected if expressions uses operators that affect arithmetic status flags	none

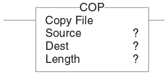
Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**

CONCAT
String
Concatenate
	 CONCAT
	String Concatenate
	Source A ?
	??
	Source B ?
	??
	Dest ?
	??
		not available	`CONCAT(SourceA,SourceB,`
			`Dest);`
				The CONCAT instruction adds ASCII characters to the end of a string.

Operand:	**Type:**	**Format:**	**Description:**
Source A	string	tag	tag that contains the initial characters
Source B	string	tag	tag that contains the end characters
Destination	string	tag	tag to store the result

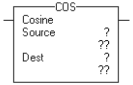
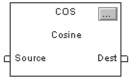
Arithmetic Status Flags:	**Major Faults:**		
not affected	Type 4	Code 51	The LEN value of the string tag is greater than the DATA size of the string tag. Check that no instruction is writing to the LEN member of the string tag and that in the LEN value, you entered the number of characters that the string contains.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

COP
Copy File
		not available	`COP(Source,Dest`
			`Length);`
				The COP instruction copies the value(s) in the Source to the Destination. The Source remains unchanged.
				The data can change during the copy operation

Operand:	Type:		Format:	Description:
Source	SINT INT DINT	REAL string structure	tag	initial element to copy the Source and Destination operands should be the same data type, or unexpected results may occur
Destination	SINT INT DINT	REAL string structure	tag	initial element to be overwritten by the Source the Source and Destination operands should be the same data type, or unexpected results may occur
Length	DINT		immediate tag	number of Destination elements to copy

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
COS **Cosine**			`dest := COS(source);`	The COS instruction takes the cosine of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

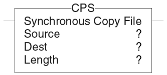
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	find the cosine of this value
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
COS tag	FBD_MATH_ADVANCED	structure	COS structure (default parameters):

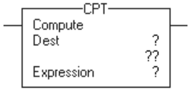
Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CPS Synchronous Copy File	 CPS Synchronous Copy File Source ? Dest ? Length ?	not available	`CPS(Source,Dest Length);`	The CPS instruction copies the value(s) in the Source to the Destination. The Source remains unchanged. The data cannot change during the copy operation.

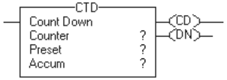
Operand:	Type:		Format:	Description:
Source	SINT INT DINT	REAL string structure	tag	initial element to copy the Source and Destination operands should be the same data type, or unexpected results may occur
Destination	SINT INT DINT	REAL string structure	tag	initial element to be overwritten by the Source the Source and Destination operands should be the same data type, or unexpected results may occur
Length	DINT		immediate tag	number of Destination elements to copy

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CPT **Compute**		not available	destination := *numeric_expresion;*	The CPT instruction performs the arithmetic operations you define in the expression.

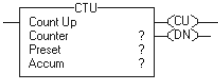
Operand:	Type:		Format:	Description:
Destination	SINT INT	DINT REAL	immediate tag	tag to store the result
Expression	SINT INT	DINT REAL	immediate tag	an expression consisting of tags and/or immediate values separated by operators

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CTD **Counter Down**	 CTD Count Down Counter ? Preset ? Accum ? CD DN	see CTUD	see CTUD	The CTD instruction counts downward.

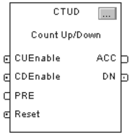
Operand:	Type:	Format:	Description:
Counter	COUNTER	tag	counter structure
Preset	DINT	immediate	how low to count
Accum	DINT	immediate	number of times the counter has counted; initial value is typically 0

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CTU **Counter Up**	 CTU Count Up Counter ? Preset ? Accum ? CU DN	see CTUD	see CTUD	The CTU instruction counts upward.

Operand:	Type:	Format:	Description:
Counter	COUNTER	tag	counter structure
Preset	DINT	immediate	how high to count
Accum	DINT	immediate	number of times the counter has counted; initial value is typically 0

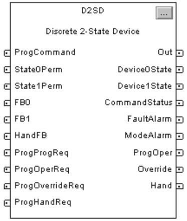
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
CTUD Count Up/Down	see CTU and CTD	 CTUD ... Count Up/Down CUEnable CDEnable PRE Reset ACC DN	`CTUD(CTUD_tag);`	The CTUD instruction counts up by one when CUEnable transitions from clear to set. The instruction counts down by one when CDEnable transitions from clear to set.

Operand:	Type:	Format:	Description:
CTUD tag	FBD_COUNTER	structure	CTUD structure (default parameters):

Parameter:	Type:	Description:
CUEnable	BOOL	enable up count When input toggles from clear to set, accumulator counts up by one.
CDEnable	BOOL	enable down count When input toggles from clear to set, accumulator counts down by one.
PRE	DINT	counter preset value
Reset	BOOL	request to reset the timer
ACC	DINT	accumulated value
DN	BOOL	counting done

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
D2SD **Discrete** **2-State Device**	not available		`D2SD(D2SD_tag);`	The D2SD instruction controls a discrete device which has only two possible states such as on/off, open/closed, etc.

Operand:	Type:	Format:	Description:
D2SD tag	DISCRETE_ 2STATE	structure	D2SD structure (default parameters):

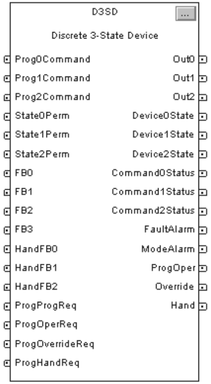
Parameter:	Type:	Description:
ProgCommand	BOOL	program state command
State*x*Perm	BOOL	state *x* permissive, where *x* = 0 or 1 unless in Hand or Override mode, this input must be set for the device to enter the state
FB*x*	BOOL	feedback input, where *x* = 0 or 1
HandFB	BOOL	hand feedback input when set, the field device is being requested to enter the 1 state; when cleared, the field device is being requested to enter the 0 state
ProgProgReq	BOOL	program program request
ProgOperReq	BOOL	program operator request
ProgOverrideReq	BOOL	program override request

continued

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

	Parameter:	Type:	Description:
D2SD **Discrete** **2-State Device** (continued)	ProgHandReq	BOOL	program hand request
	Out	BOOL	output of the instruction
	Device*x*State	BOOL	device *x* state output, where *x* = 0 or 1
	CommandStatus	BOOL	command status output
	FaultAlarm	BOOL	fault alarm output
	ModeAlarm	BOOL	mode alarm output
	ProgOper	BOOL	program/operator control indicator
	Override	BOOL	override mode indicator
	Hand	BOOL	hand mode indicator

Arithmetic Status Flags:	Major Faults:
not affected	none

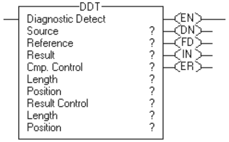
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
D3SD **Discrete** **3-State Device**	not available		`D3SD(D3SD_tag);`	The D3SD instruction controls a discrete device having three possible states such as fast/slow/off, forward/stop/reverse, etc.

Operand:	Type:	Format:	Description:
D3SD tag	DISCRETE_ 3STATE	structure	D3SD structure (default parameters):

Parameter:	Type:	Description:
Prog*x*Command	BOOL	program state x command, where *x* = 0, 1, or 2
State*x*Perm	BOOL	state *x* permissive, where *x* = 0, 1, or 2 unless in Hand or Override mode, this input must be set for the device to enter the state
FB*x*	BOOL	feedback input; where *x* = 0,1, 2, or 3

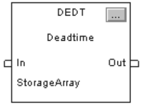
continued

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
D3SD **Discrete** **3-State Device** (continued)		**Parameter:**	**Type:**	**Description:**
		HandFB*x*	BOOL	hand feedback input, where *x* = 0, 1, or 2 when set, the field device is being requested to enter the 1 state; when cleared, the field device is being requested to enter some other state
		ProgProgReq	BOOL	program program request
		ProgOperReq	BOOL	program operator request
		ProgOverrideReq	BOOL	program override request
		ProgHandReq	BOOL	program hand request
		Out*x*	BOOL	output of the instruction, where *x* = 0, 1, or 2
		Device*x*State	BOOL	device *x* state output, where *x* = 0, 1, or 2
		Command*x*Status	BOOL	command status output, where *x* = 0, 1, or 2
		FaultAlarm	BOOL	fault alarm output
		ModeAlarm	BOOL	mode alarm output
		ProgOper	BOOL	program/operator control indicator
		Override	BOOL	override mode indicator
		Hand	BOOL	hand mode indicator
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**
DDT Diagnostic Detect		not available	not available	The DDT instruction compares bits in a Source array with bits in a Reference array to determine changes of state.

Operand:	**Type:**	**Format:**	**Description:**
Source	DINT	array tag	array to compare to the reference; **do not** use CONTROL.POS in the subscript
Reference	DINT	array tag	array to compare to the source; **do not** use CONTROL.POS in the subscript
Result	DINT	array tag	array to store the results; **do not** use CONTROL.POS in the subscript
Cmp control	CONTROL	structure	control structure for the compare
Length	DINT	immediate	number of bits to compare
Position	DINT	immediate	current position in the source; initial value typically 0
Result control	CONTROL	structure	control structure for the results
Length	DINT	immediate	number of storage locations in the result
Position	DINT	immediate	current position in the result; initial value typically 0

Arithmetic Status Flags:	**Major Faults:**		
not affected	Type 4	Code 20	Result.POS > size of Result array

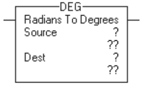
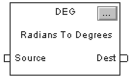
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DEDT **Deadtime**	not available		`DEDT(DEDT_tag,storage);`	The DEDT instruction performs a delay of a single input. You select the amount of deadtime delay.

Operand:	Type:	Format:	Description:
DEDT tag	DEADTIME	structure	DEDT structure (default parameters):
storage	REAL	array	deadtime buffer

DEDT structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DEG **Degrees**			`dest := DEG(source);`	The DEG instruction converts the Source (in radians) to degrees and stores the result in the Destination.

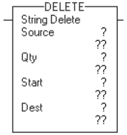
	Operand:	Type:		Format:	Description:
Relay Ladder and Structured Text					
	Source	SINT INT	DINT REAL	immediate tag	value to convert to degrees
	Destination	SINT INT	DINT REAL	tag	tag to store the result

	Operand:	Type:	Format:	Description:
Function Block				
	DEG tag	FBD_MATH_ ADVANCED	structure	DEG structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the conversion instruction
Dest	REAL	result of the conversion instruction

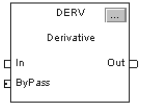
Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

DELETE
String Delete
	 DELETE
	String Delete
	Source ?
	??
	Qty ?
	??
	Start ?
	??
	Dest ?
	??
		not available	`DELETE(Source,Qty,`
			`Start,Dest);`
				The DELETE instruction removes ASCII characters from a string.

Operand:	Type:	Format:	Description:
Source	string	tag	tag that contains the string from which you want to delete characters
Quantity	SINT DINT INT	immediate tag	number of characters to delete; the Start plus the Quantity must be less than or equal to the DATA size of the Source
Start	SINT DINT INT	immediate tag	position of the first character to delete; enter a number between 1 and the DATA size of the Source
Destination	string	tag	tag to store the result

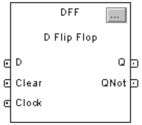
Arithmetic Status Flags:	Major Faults:		
not affected	4	51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	4	56	The Start or Quantity value is invalid. Check that: • the Start value is between 1 and the DATA size of the Source. • the Start value plus the Quantity value is less than or equal to the DATA size of the Source.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DERV Derivative	not available		`DERV(DERV_tag);`	The DERV instruction calculates the amount of change of a signal over time in per-second units.

Operand:	Type:	Format:	Description:
DERV tag	DERIVATIVE	structure	DERV structure (default parameters):

Parameter:	Type:	Description:
In	REAL	input to the instruction
ByPass	BOOL	request to bypass the algorithm; when set, the instruction sets Out = In
Out	REAL	calculated output of the algorithm

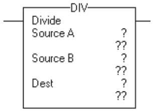
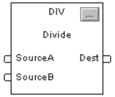
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DFF **D FLip-Flop**	not available	 DFF ... D Flip Flop D Q Clear QNot Clock	`DFF(DFF_tag);`	The DFF instruction sets the Q output to the state of the D input on a cleared to set transition of the Clock input. The QNot output is set to the opposite state of the Q output.

Operand:	Type:	Format:	Description:
DFF tag	FLIP_FLOP_D	structure	DFF structure (default parameters):

Parameter:	Type:	Description:
D	BOOL	input to the instruction
Clear	BOOL	clear input to the instruction; if set, the instruction clears Q and sets QNot
Clock	BOOL	Clock input to the instruction
Q	BOOL	output of the instruction
QNot	BOOL	complement of the Q output

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DIV **Divide**	 DIV Divide Source A ? ?? Source B ? ?? Dest ? ??		`dest := sourceA / sourceB;`	The DIV instruction divides Source A by Source B and places the result in the Destination.

Relay Ladder and Structured Text

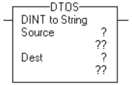
Operand:	Type:		Format:	Description:
Source A	SINT INT	DINT REAL	immediate tag	value of the dividend
Source B	SINT INT	DINT REAL	immediate tag	value of the divisor
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
DIV tag	FBD_MATH	structure	DIV structure (default parameters):

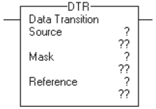
Parameter:	Type:	Description:
SourceA	REAL	value of the dividend
SourceB	REAL	value of the divisor
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 4	the divisor is 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DTOS **DINT to String**	 DTOS DINT to String Source ? ?? Dest ? ??	not available	`DTOS(Source,Dest);`	The DTOS instruction produces the ASCII representation of a value.

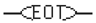
Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	tag	tag that contains the value; if the Source is a REAL, the instruction converts it to a DINT value
Destination	string		tag	tag to store the ASCII value

Arithmetic Status Flags:	Major Faults:		
not affected	4	51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	4	52	The output string is larger than the destination. Create a new string data type that is large enough for the output string. Use the new string data type as the data type for the destination.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
DTR **Data** **Transitional**		not available	not available	The DTR instruction passes the Source value through a Mask and compares the result with the Reference value.

Operand:	Type:	Format:	Description:
Source	DINT	immediate tag	array to compare to the reference
Mask	DINT	immediate tag	which bits to block or pass
Reference	DINT	tag	array to compare to the source

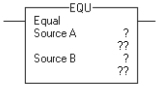
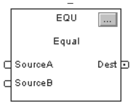
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
EOT End of Transition		not available	`EOT(data_bit);`	The EOT instruction returns a boolean state to an SFC transition.

Operand:	Type:	Format:	Description:
data bit	BOOL	tag	state of the transition (0=executing, 1=completed)

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

EQU
Equal To
			`IF sourceA = sourceB THEN`
			`    <statements>;`
				The EQU instruction tests whether Source A is equal to Source B.

Relay Ladder and Structured Text

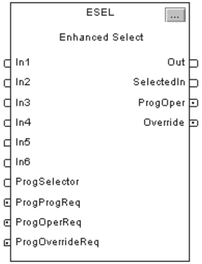
Operand:	Type:	Format:	Description:
Source A	SINT INT DINT REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT REAL string	immediate tag	value to test against Source A

Function Block

Operand:	Type:	Format:	Description:
EQU tag	FBD_COMPARE	structure	EQU structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ESEL Enhanced Select	not available	 ESEL ... Enhanced Select In1 In2 In3 In4 In5 In6 ProgSelector ProgProgReq ProgOperReq ProgOverrideReq Out SelectedIn ProgOper Override	`ESEL(ESEL_tag);`	The ESEL instruction lets you select one of as many as six inputs. Selection options include: • manual select (either by operator or by program) • high select • low select • median select • average (mean) select

Operand:	Type:	Format:	Description:
ESEL tag	SELECT_ ENHANCED	structure	ESEL structure (default parameters):

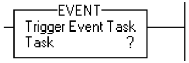
Parameter:	Type:	Description:
In*x*	REAL	analog signal inputs to the instruction, where *x* = 1-6
ProgSelector	DINT	program selector input
ProgProgReq	BOOL	program program request
ProgOperReq	BOOL	program operator request
ProgOverrideReq	BOOL	program override request

continued

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
ESEL Enhanced Select (continued)				

Parameter:	Type:	Description:
Out	REAL	calculated output of the algorithm
SelectedIn	DINT	number of inputs selected; if the selector mode is average select, the instruction sets SelectedIn = 0
ProgOper	BOOL	program/operator control indicator; set when in Program control; cleared when in Operator control
Override	BOOL	override mode; set when the instruction is in Override mode

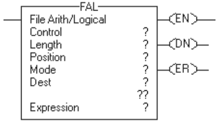
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
EVENT Trigger Event Task		not available	`EVENT(Task);`	The EVENT instruction triggers one execution of an event task.

Operand:	Type:	Format:	Description:
Task	na	task name	event task to execute

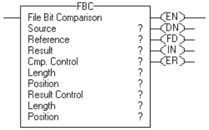
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

FAL
File Arithmetic and Logic
	 FAL
	File Arith/Logical
	Control ?
	Length ?
	Position ?
	Mode ?
	Dest ?
	??
	Expression ?
	EN
	DN
	ER
		not available	`SIZE(destination,0 length-1);`
			`FOR position = 0 TO length DO`
			`destination[position] :=` *`numeric_expression`*`;`
			`END_FOR;`
				The FAL instruction performs copy, arithmetic, logic, and function operations on data stored in an array.

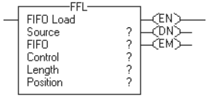
Operand:	Type:		Format:	Description:
Control	CONTROL		tag	control structure for the operation
Length	DINT		immediate	number of elements in the array to be manipulated
Position	DINT		immediate	current element in array; initial value is typically 0
Mode	DINT		immediate	how to distribute the operation; select INC, ALL, or enter a number
Destination	SINT INT	DINT REAL	tag	tag to store the result
Expression	SINT INT	DINT REAL	immediate tag	an expression consisting of tags and/or immediate values separated by operators

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 20	subscript is out of range
	Type 4	Code 21	.POS < 0 or .LEN < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
FBC **File Bit** **Compare**	 FBC File Bit Comparison Source ? Reference ? Result ? Cmp. Control ? Length ? Position ? Result Control ? Length ? Position ? EN DN FD IN ER	not available	not available	The FBC instruction compares bits in a Source array with bits in a Reference array.

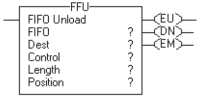
Operand:	Type:	Format:	Description:
Source	DINT	array tag	array to compare to the reference; **do not** use CONTROL.POS in the subscript
Reference	DINT	array tag	array to compare to the source; **do not** use CONTROL.POS in the subscript
Result	DINT	array tag	array to store the result; **do not** use CONTROL.POS in the subscripts
Cmp control	CONTROL	structure	control structure for the compare
Length	DINT	immediate	number of bits to compare
Position	DINT	immediate	current position in the source; initial value is typically 0
Result control	CONTROL	structure	control structure for the results
Length	DINT	immediate	number of storage locations in the result
Position	DINT	immediate	current position in the result initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	Result.POS > size of Result array

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
FFL **FIFO Load**	 FFL FIFO Load Source ? FIFO ? Control ? Length ? Position ? EN DN EM	not available	not available	The FFL instruction copies the Source value to the FIFO.

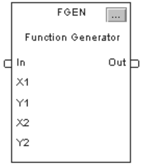
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL string structure	immediate tag	data to be stored in the FIFO
FIFO	SINT DINT INT REAL string structure	array tag	FIFO to modify; specify the first element of the FIFO **do not** use CONTROL.POS in the subscript
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the associated FFU
Length	DINT	immediate	maximum number of elements the FIFO can hold at one time
Position	DINT	immediate	next location in the FIFO where the instruction loads data; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	(starting element + .POS) > FIFO array size

Instruction:	**Relay Ladder:**	**Function Block:**	**Structured Text:**	**Description:**
FFU **FIFO Unload**	 FFU FIFO Unload FIFO ? Dest ? Control ? Length ? Position ? EU DN EM	not available	not available	The FFU instruction unloads the value from position 0 (first position) of the FIFO and stores that value in the Destination. The remaining data in the FIFO shifts down one position.

Operand:	**Type:**	**Format:**	**Description:**
FIFO	SINT DINT INT REAL string structure	array tag	FIFO to modify; specify the first element of the FIFO **do not** use CONTROL.POS in the subscript
Destination	SINT DINT INT REAL string structure	tag	value that exits the FIFO
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the associated FFL
Length	DINT	immediate	maximum number of elements the FIFO can hold at one time
Position	DINT	immediate	next location in the FIFO where the instruction unloads data; initial value is typically 0

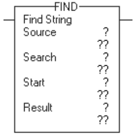
Arithmetic Status Flags:	**Major Faults:**		
not affected	Type 4	Code 20	Length > FIFO array size

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
FGEN Function Generator	not available	 FGEN ... Function Generator In Out X1 Y1 X2 Y2	`FGEN(FGEN_tag,X1,Y1,X2,Y2);`	The FGEN instruction converts an input based on a piece-wise linear function.

<table>
<tr><th>Operand:</th><th>Type:</th><th>Format:</th><th>Description:</th></tr>
<tr><td>FGEN tag</td><td>FUNCTION_
GENERATOR</td><td>structure</td><td>FGEN structure (default parameters):
<table>
<tr><th>Parameter:</th><th>Type:</th><th>Description:</th></tr>
<tr><td>In</td><td>REAL</td><td>analog signal input to the instruction</td></tr>
<tr><td>Out</td><td>REAL</td><td>calculated output of the algorithm</td></tr>
</table></td></tr>
<tr><td>X1</td><td>REAL</td><td>array</td><td>X-axis array, table one
combine with the Y-axis array, table one to define the points of the first piece-wise linear curve</td></tr>
<tr><td>Y1</td><td>REAL</td><td>array</td><td>Y-axis array, table one
combine with the X-axis array, table one to define the points of the first piece-wise linear curve</td></tr>
<tr><td>X2</td><td>REAL</td><td>array</td><td>(optional) X-axis array, table two
combine with the Y-axis array, table two to define the points of the second piece-wise linear curve</td></tr>
<tr><td>Y2</td><td>REAL</td><td>array</td><td>(optional) Y-axis array, table two
combine with the X-axis array, table two to define the points of the second piece-wise linear curve</td></tr>
</table>

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

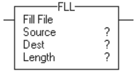
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

FIND
Find String
	 FIND
	Find String
	Source ?
	??
	Search ?
	??
	Start ?
	??
	Result ?
	??
		not available	`FIND(Source,Search,`
			`Start,Result);`
				The FIND instruction locates the starting position of a specified string within another string

Operand:	Type:		Format:	Description:
Source	string		tag	string to search in
Search	string		tag	string to find
Start	SINT INT	DINT	immediate tag	position in Source to start the search; enter a number between 1 and the DATA size of the Source.
Result	SINT INT	DINT	tag	tag that stores the starting position of the string to find

Arithmetic Status Flags:	Major Faults:		
not affected	4	51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	4	56	The Start value is invalid. Check that the Start value is between 1 and the DATA size of the Source.

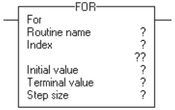
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

FLL
File Fill
		not available	`SIZE(destination,0`
			`length);`
			`FOR position = 0 TO length-1`
			`DO`
			`     destination[position]`
			`:= source;`
			`END_FOR;`
				The FLL instruction fills elements of an array with the Source value. The Source remains unchanged.

Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	element to copy the Source and Destination operands should be the same data type, or unexpected results may occur
Destination	SINT DINT INT REAL structure	tag	initial element to be overwritten by the Source the Source and Destination operands should be the same data type, or unexpected results may occur the preferred way to initialize a structure is to use the COP instruction
Length	DINT	immediate	number of elements to fill

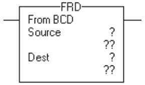
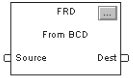
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

FOR
For
	 FOR
	For
	Routine name ?
	Index ?
	??
	Initial value ?
	Terminal value ?
	Step size ?
		not available	`FOR` *`count:= initial_value`* `TO`
			`final_value` `BY` *`increment`* `DO`
			`<statement>;`
			`END_FOR;`
				The FOR instruction executes a routine repeatedly.

Operand:	Type:	Format:	Description:
Routine name	ROUTINE	routine name	routine to execute
Index	DINT	tag	counts how many times the routine has been executed
Initial value	SINT DINT INT	immediate tag	value at which to start the index
Terminal value	SINT DINT INT	immediate tag	value at which to stop executing the routine
Step size	SINT DINT INT	immediate tag	amount to add to the index each time the FOR instruction executes the routine

Arithmetic Status Flags:	Major Faults:		
not affected	4	31	main routine contains a RET instruction

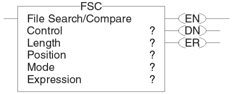
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
FRD **Convert to** **Integer**			not available	The FRD instruction converts a BCD value (Source) to an integer value and stores the result in the Destination.

	Operand:	Type:	Format:	Description:
Relay Ladder	Source	SINT DINT INT	immediate tag	value to convert
	Destination	SINT DINT INT	tag	tag to store the result

	Operand:	Type:	Format:	Description:
Function Block	FRD tag	FBD_CONVERT	structure	FRD structure (default parameters):

Parameter:	Type:	Description:
Source	DINT	input to the conversion instruction.
Dest	DINT	result of the math instruction.

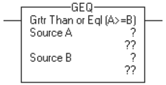
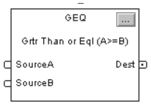
Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
FSC **File Search and Compare**	 FSC File Search/Compare Control ? Length ? Position ? Mode ? Expression ? EN DN ER	not available	not available	The FSC instruction compares values in an array, element by element.

Operand:	Type:	Format:	Description:
Control	CONTROL	tag	control structure for the operation
Length	DINT	immediate	number of elements in the array to be manipulated
Position	DINT	immediate	offset into array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
affected	4	21	.POS < 0 or .LEN < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

**GEQ
Greater Than or Equal To**
	 GEQ
	Grtr Than or Eql (A>=B)
	Source A
	?
	??
	Source B
	?
	??
		 GEQ
		Grtr Than or Eql (A>=B)
		SourceA
		Dest
		SourceB
			`IF sourceA >= sourceB THEN`
			`<statements>;`
				The GEQ instruction tests whether Source A is greater than or equal to Source B.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source A	SINT INT DINT	REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT	REAL string	immediate tag	value to test against Source A

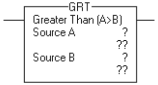
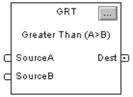
Function Block

Operand:	Type:	Format:	Description:
GEQ tag	FBD_COMPARE	structure	GEQ structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

GRT
Greater Than
			`IF sourceA > sourceB THEN`
			`     <statements>;`
				The GRT instruction tests whether Source A is greater than Source B.

Relay Ladder and Structured Text

Operand:	Type:	Format:	Description:
Source A	SINT INT DINT REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT REAL string	immediate tag	value to test against Source A

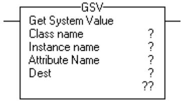
Function Block

Operand:	Type:	Format:	Description:
GRT tag	FBD_COMPARE	structure	GRT structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

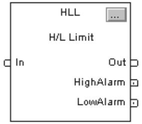
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

GSV
Get System Value
		not available	`GSV(ClassName,`
			`InstanceName,`
			`AttributeName,Dest);`
				The GSV instructions get s controller system data that is stored in objects.

Operand:	Type:		Format:	Description:
Class name	na		name	name of object
Instance name	na		name	name of specific object, when object requires name
Attribute Name	na		name	attribute of object; data type depends on the attribute you select
Destination	SINT INT	DINT REAL	tag	destination for attribute data

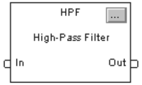
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 5	invalid object address
	Type 4	Code 6	• specified an object that does not support GSV/SSV • invalid attribute • did not supply enough information for an SSV instruction
	Type 4	Code 7	the GSV destination was not large enough to hold the requested data

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
HLL **High/Low Limit**	not available	 HLL ... H/L Limit In Out HighAlarm LowAlarm	`HLL(HLL_tag);`	The HLL instruction limits an analog input between two values. You can select high/low, high, or low limits.

Operand:	Type:	Format:	Description:
HLL tag	HL_LIMIT	structure	HLL structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm
HighAlarm	BOOL	high alarm indicator; set when In ≥ HighLimit
LowAlarm	BOOL	low alarm indicator; set when In ≤LowLimit

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

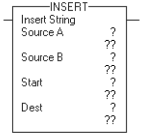
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
HPF **High Pass Filter**	not available		`HPF(HPF_tag);`	The HPF instruction provides a filter to attenuate input frequencies that are below the cutoff frequency.

Operand:	Type:	Format:	Description:
HPF tag	FILTER_HIGH_ PASS	structure	HPF structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

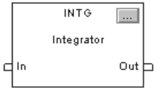
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

INSERT
Insert String
	 INSERT
	Insert String
	Source A ?
	??
	Source B ?
	??
	Start ?
	??
	Dest ?
	??
		not available	`INSERT(SourceA,SourceB,`
			`Start,Dest);`
				The INSERT instruction adds ASCII characters to a specified location within a string.

Operand:	Type:	Format:	Description:
Source A	string	tag	string to add the characters to
Source B	string	tag	string containing the characters to add
Start	SINT DINT INT	immediate tag	position in Source A to add the characters; enter a number between 1 and the DATA size of the Source.
Result	string	tag	string to store the result

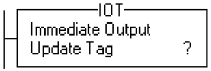
Arithmetic Status Flags:	Major Faults:		
not affected	4	51	The LEN value of the string tag is greater than the DATA size of the string tag. ChecK: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	4	56	The Start value is invalid. Check that the Start value is between 1 and the DATA size of the Source.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
INTG **Integrator**	not available		`INTG(INTG_tag);`	The INTG instruction implements an integral operation. This instruction is designed to execute in a task where the scan rate remains constant.

Operand:	Type:	Format:	Description:
INTG tag	INTEGRATOR	structure	INTG structure (default parameters):

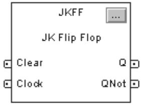
Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
IOT **Immediate** **Output**		not available	`IOT(output_tag);`	The IOT instruction immediately updates the specified output data (output tag or produced tag).

Operand:	Type:	Format:	Description:
Output tag	tag name	tag	tag that you want to update, either an output tag of an I/O module or a produced tag **do not** choose a member or element of a tag

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
JKFF **JK FLip-Flop**	not available		`JKFF(JKFF_tag);`	The JKFF instruction complements the Q and QNot outputs when the Clock input transitions from cleared to set.

Operand:	Type:	Format:	Description:
JKFF tag	FLIP_FLOP_JK	structure	JKFF structure (default parameters):

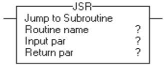
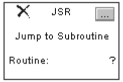
Parameter:	Type:	Description:
Clear	BOOL	clear input to the instruction; if set, the instruction clears Q and sets QNot
Clock	BOOL	Clock input to the instruction
Q	BOOL	output of the instruction
QNot	BOOL	complement of the Q output

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
JMP **Jump**		not available	not available	The JMP and LBL instructions skip portions of ladder logic.

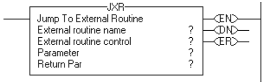
Operand:	Type:	Format:	Description:
Label name	na	name	name of associated LBL instruction

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 42	label does not exist

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
JSR Jump to Subroutine	 JSR Jump to Subroutine Routine name ? Input par ? Return par ?	 JSR Jump to Subroutine Routine: ?	`JSR(RoutineName InputCount, InputPar,ReturnPar);`	The JSR instruction jumps execution to a different routine.

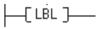
Operand:	Type:	Format:	Description:
Routine name	ROUTINE	name	routine to execute
Input parameter	BOOL DINT SINT REAL INT structure	immediate tag array tag	data from this routine that you want to copy to a tag in the subroutine • parameters are optional • enter multiple parameters, if needed
Return parameter	BOOL DINT SINT REAL INT structure	tag array tag	tag in this routine to which you want to copy a result of the subroutine • parameters are optional • enter multiple parameters, if needed
Input count	SINT DINT INT REAL	immediate	number of input parameters (structured text only)

Arithmetic Status Flags:	Major Faults:		
affected	4	31	• JSR instruction has fewer input parameters than SBR instruction • RET instruction has fewer return parameters than JSR instruction • main routine contains a RET instruction
	4	0	JSR instruction jumps to a fault routine

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
JXR **Jump to External Routine**		not available	not available	The JXR instruction executes an external routine. This instruction is only supported by the SoftLogix5800 controllers.

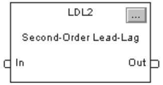
Operand:	Type:	Format:	Description:
External routine name	ROUTINE	name	external routine to execute
External routine control	EXT_ROUTINE_ CONTROL	tag	control structure
Parameter	BOOL DINT SINT REAL INT structure	immediate tag array tag	data from this routine that you want to copy to a variable in the external routine • parameters are optional • enter multiple parameters, if needed • you can have as many as 10 parameters
Return parameter	BOOL DINT SINT REAL INT	tag	tag in this routine to which you want to copy a result of the external routine • the return parameter is optional. • you can have only one return parameter

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LBL **Label**		not available	not available	The JMP and LBL instructions skip portions of ladder logic.

Operand:	Type:	Format:	Description:
Label name	na	name	execution jumps to LBL instruction with referenced label name

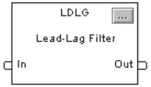
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 42	label does not exist

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LDL2 Second-Order Lead Lag	not available		`LDL2(LDL2_tag);`	The LDL2 instruction provides a filter with a pole pair and a zero pair. The frequency and damping of the pole and zero pairs are adjustable. The pole or zero pairs can be either complex (damping less than unity) or real (damping greater than or equal to unity).

Operand:	Type:	Format:	Description:
LDL2 tag	LEAD_LAG_SEC_ ORDER	structure	LDL2 structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

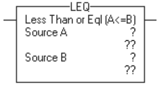
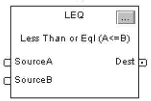
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LDLG **Lead Lag**	not available		`LDLG(LDLG_tag);`	The LDLG instruction provides a phase lead-lag compensation for an input signal. This instruction is typically used for feedforward PID control or for process simulations.

Operand:	Type:	Format:	Description:
LDLG tag	LEAD_LAG	structure	LDLG structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

LEQ
Less Than or Equal To
	 LEQ
	Less Than or Eql (A<=B)
	Source A ?
	??
	Source B ?
	??
		 LEQ
		...
		Less Than or Eql (A<=B)
		SourceA
		Dest
		SourceB
			`IF sourceA <= sourceB THEN`
			`    <statements>;`
				The LEQ instruction tests whether Source A is less than or equal to Source B.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source A	SINT INT DINT	REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT	REAL string	immediate tag	value to test against Source A

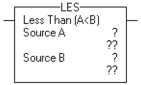
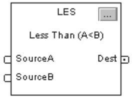
Function Block

Operand:	Type:	Format:	Description:
LEQ tag	FBD_COMPARE	structure	LEQ structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

LES
Less Than
	 LES
	Less Than (A<B)
	Source A
	?
	??
	Source B
	?
	??
			`IF sourceA < sourceB THEN`
			`    <statements>;`
				The LES instruction tests whether Source A is less than Source B.

Relay Ladder and Structured Text

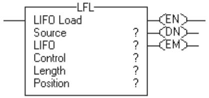
Operand:	Type:	Format:	Description:
Source A	SINT INT DINT REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT REAL string	immediate tag	value to test against Source A

Function Block

Operand:	Type:	Format:	Description:
LES tag	FBD_COMPARE	structure	LES structure (default parameters):

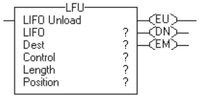
Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LFL **LIFO Load**	 LFL LIFO Load Source ? LIFO ? Control ? Length ? Position ? EN DN EM	not available	not available	The LFL instruction copies the Source value to the LIFO.

Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL string structure	immediate tag	data to be stored in the LIFO
LIFO	SINT DINT INT REAL string structure	array tag	LIFO to modify; specify the first element of the LIFO **do not** use CONTROL.POS in the subscript
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the associated LFU
Length	DINT	immediate	maximum number of elements the LIFO can hold at one time
Position	DINT	immediate	next location in the LIFO where the instruction loads data; initial value is typically 0

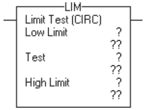
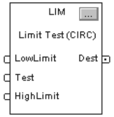
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	(starting element + .POS) > LIFO array size

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LFU **LIFO Unload**	 LFU LIFO Unload LIFO ? Dest ? Control ? Length ? Position ? EU DN EM	not available	not available	The LFU instruction unloads the value at .POS of the LIFO and stores 0 in that location.

Operand:	Type:	Format:	Description:
LIFO	SINT DINT INT REAL string structure	array tag	LIFO to modify; specify the first element of the LIFO **do not** use CONTROL.POS in the subscript
Destination	SINT DINT INT REAL string structure	tag	value that exits the LIFO
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the associated LFL
Length	DINT	immediate	maximum number of elements the LIFO can hold at one time
Position	DINT	immediate	next location in the LIFO where the instruction unloads data; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	Length > LIFO array size

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

LIM
Limit
	 LIM
	Limit Test (CIRC)
	Low Limit ?
	??
	Test ?
	??
	High Limit ?
	??
		 LIM
		...
		Limit Test (CIRC)
		LowLimit
		Dest
		Test
		HighLimit
			`IF (LowLimit <= HighLimit`
			`AND`
			`     (Test >= LowLimit AND`
			`Test <= HighLimit)) OR`
			`     (LowLimit >= HighLimit`
			`AND`
			`     (Test <= LowLimit OR`
			`Test >= HighLimit)) THEN`
			`     <statement>;`
			`END_IF;`
				The LIM instruction tests whether the Test value is within the range of the Low Limit to the High Limit.

Relay Ladder and Structured Text

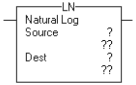
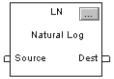
Operand:	Type:		Format:	Description:
Low Limit	SINT INT	DINT REAL	immediate tag	value of lower limit
Test	SINT INT	DINT REAL	immediate tag	value to test
High Limit	SINT INT	DINT REAL	immediate tag	value of upper limit

Function Block

Operand:	Type:	Format:	Description:
LIM tag	FBD_LIMIT	structure	LIM structure (default parameters):

Parameter:	Type:	Description:
LowLimit	REAL	value of lower limit
Test	REAL	value to test against limits
HighLimit	REAL	value of upper limit
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LN **Natural Log**			`dest := LN(source);`	The LN instruction takes the natural log of the Source and stores the result in the Destination.

Relay Ladder and Structured Text

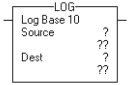
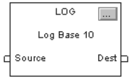
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	find the natural log of this value
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
LN tag	FBD_MATH_ ADVANCED	structure	LN structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LOG **Log Base 10**			`dest := LOG(source);`	The LOG instruction takes the log base 10 of the Source and stores the result in the Destination.

Relay Ladder and Structured Text

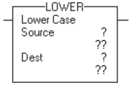
Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	find the log of this value
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
LOG tag	FBD_MATH_ ADVANCED	structure	LOG structure (default parameters):

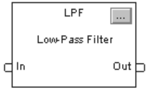
Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LOWER **Lower Case**		not available	`LOWER(Source,Dest);`	The LOWER instruction converts the alphabetical characters in a string to lower case characters.

Operand:	Type:	Format:	Description:
Source	string	tag	tag that contains the characters that you want to convert to lower case
Destination	string	tag	tag to store the characters in lower case

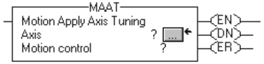
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
LPF **Low Pass Filter**	not available		`LPF(LPF_tag);`	The LPF instruction provides a filter to attenuate input frequencies that are above the cutoff frequency.

Operand:	Type:	Format:	Description:
LPF tag	FILTER_LOW_ PASS	structure	LPF structure (default parameters):

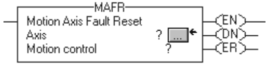
Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAAT **Motion Apply Axis Tuning**	 MAAT Motion Apply Axis Tuning Axis ? Motion control ? EN DN ER	not available	`MAAT(Axis,MotionControl);`	The MAAT computes a complete set of servo gains and dynamic limits based on the results of a previously run MRAT instruction and updates the motion module with these new gain parameters.

Operand:	Type:	Format:	Description:
Axis	AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion control	MOTION_INSTRUCTION	tag	motion structure

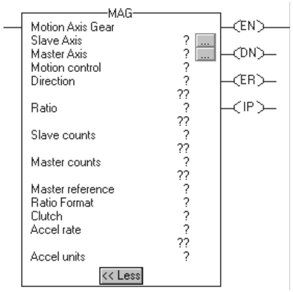
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAFR **Motion Axis Fault Reset**	 MAFR Motion Axis Fault Reset Axis ? Motion control ? EN DN ER	not available	`MAFR(Axis,MotionControl);`	The MAFR instruction clears all motion faults for an axis. This is the only method for clearing axis motion faults.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

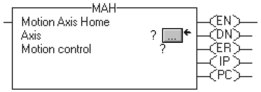
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MAG
Motion Axis Gear
	
	MAG
	Motion Axis Gear
	Slave Axis ?
	Master Axis ?
	Motion control ?
	Direction ? ??
	Ratio ? ??
	Slave counts ? ??
	Master counts ? ??
	Master reference ?
	Ratio Format ?
	Clutch ?
	Accel rate ? ??
	Accel units ?
	<< Less
	EN
	DN
	ER
	IP
		not available	`MAG(SlaveAxis,MasterAxis,`
			`MotionControl,Direction,`
			`Ratio,SlaveCounts,`
			`MasterCounts,`
			`MasterReference,`
			`RatioFormat,Clutch,`
			`AccelRate,AccelUnits);`
				The MAG instruction provides electronic gearing between any two axes in a specified direction and at a specified ratio

Operand:	Type:	Format:	Description:
Slave axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Master axis	AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	axis that the slave axis follows
Motion control	MOTION_INSTRUCTION	tag	motion structure

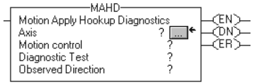
continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MAG Motion Axis Gear (continued)	Direction	UINT32	immediate tag	relative direction that the Slave axis tracks the Master Axis: • 0 = slave axis moves in the same direction as the master axis • 1 = slave axis moves in the opposite direction of its current direction • 2 = slave axis reverses from current or previous • 3 = slave axis to continue its current or previous direction
	Ratio	REAL	immediate tag	signed Real value establishing the gear ratio in Slave User Units per Master User Unit
	Slave counts	UINT32	immediate tag	slave counts
	Master counts	UINT32	immediate tag	master counts
	Master reference	BOOL	immediate	master position reference: 0 = actual position, 1 = command position
	Ratio format	BOOL	immediate	ratio format: • 0 = real gear ratio • 1 = integer fraction of slave encoder counts to master encoder counts
	Clutch	BOOL	immediate	whether Clutch is enabled or disabled
	Accel rate	BOOL	immediate tag	acceleration rate of the Slave Axis in% or Acceleration Units
	Accel units	DINT	immediate	units used to display the Acceleration value: 0 = units per sec^2; 1 =% of maximum acceleration
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAH Motion Axis Home	 MAH Motion Axis Home Axis ? Motion control ? EN DN ER IP PC	not available	`MAH(Axis,MotionControl);`	The MAH instruction homes an axis.

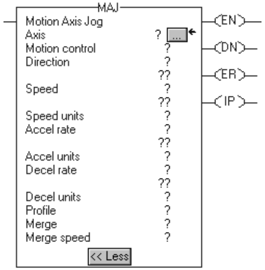
Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion control	MOTION_INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAHD Motion Apply Hookup Diagnostics	 MAHD Motion Apply Hookup Diagnostics Axis ? ... Motion control ? Diagnostic Test ? Observed Direction ? EN DN ER	not available	`MAHD(Axis,MotionControl, DiagnosticTest, ObservedDirection);`	The MAHD instruction applies the results of a previously run MRHD instruction to generate a new set of encoder and servo polarities based on the observed direction of motion during the test.

Operand:	Type:	Format:	Description:
Axis	AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Diagnostic test	UDINT	immediate	test for the motion module to run: • 0 = motor/encoder hookup test • 1 = encoder hookup test • 2 = encoder marker test
Observed direction	BOOL	immediate	direction of the test motion: 0 = forward; 1 = reverse

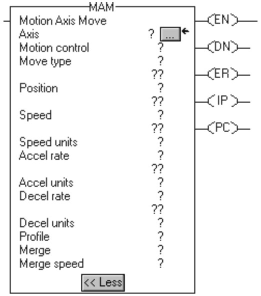
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAJ **Motion Axis Jog**	MAJ Motion Axis Jog Axis ? Motion control ? Direction ? ?? Speed ? ?? Speed units ? Accel rate ? ?? Accel units ? Decel rate ? ?? Decel units ? Profile ? Merge ? Merge speed ? << Less EN DN ER IP	not available	`MAJ(Axis,MotionControl, Direction,Speed,SpeedUnits, AccelRate,AccelUnits, DecelRate,DecelUnits, Profile,Merge,MergeSpeed);`	The MAJ instruction initiates a jog motion profile for the specified axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Direction	UDINT	immediate tag	direction of jog: 0 = forward jog; 1 = reverse jog
Speed	REAL	immediate tag	speed to move the axis in% or Speed Units
Speed units	UDINT	immediate	engineering units for the Speed value: 0 = units per sec; 1 =% of maximum speed

continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MAJ Motion Axis Jog (continued)	Accel units	UDINT	immediate	engineering units for the Acceleration value: 0 = units per sec^2; 1 =% of maximum acceleration
	Accel rate	REAL	immediate tag	acceleration rate of the axis in% or Acceleration Units
	Decel rate	REAL	immediate or tag	deceleration rate of the axis in% or Deceleration Units
	Decel units	UDINT	immediate	engineering units for the Deceleration value: 0 = units per sec^2; 1 =% of maximum deceleration
	Profile	UDINT	immediate	select the velocity profile to run the jog: 0 = trapezoidal; 1 = S-curve
	Merge	UDINT	immediate	instructs the motion control to turn all current axis motion
	Merge speed	UDINT	immediate	determines whether the speed is the specified Speed value of this instruction or the Current axis speed: • 0 = programmed value in the speed field • 1 = current axis speed
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

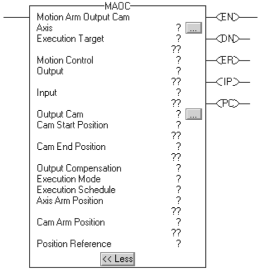
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAM Motion Axis Move	 MAM Motion Axis Move Axis ? ... Motion control ? Move type ? ?? Position ? ?? Speed ? ?? Speed units ? Accel rate ? ?? Accel units ? Decel rate ? ?? Decel units ? Profile ? Merge ? Merge speed ? << Less EN DN ER IP PC	not available	`MAM(Axis,MotionControl, MoveType,Position,Speed, SpeedUnits,AccelRate, AccelUnits,DecelRate, DecelUnits,Profile,Merge, MergeSpeed);`	The MAM instruction initiates a move profile for the specified axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Move type	UDINT	immediate or tag	type of move operation: 0 = Absolute Move; 1 = Incremental Move; 2 = Rotary Shortest Path Move; 3 = Rotary Positive Move; 4 = Rotary Negative Move; 5 = Absolute Master Offset; 6 = Incremental Master Offset
Position /Distance	REAL	immediate tag	value of the absolute command position to move to, or for incremental movement, the value of the distance to move from the current command position.
Speed	REAL	immediate tag	speed to move the axis in either% or Speed units.

continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MAM Motion Axis Move (continued)	Speed Units	BOOL	immediate	units for the Speed value:0 =units per sec; 1 =% of maximum speed
	Accel rate	REAL	immediate or tag	acceleration rate of the axis in% or Acceleration units
	Accel units	BOOL	immediate	units for the Accel value: 0 = units per sec^2; 1 =% of maximum acceleration
	Decel rate	REAL	immediate or tag	deceleration rate of the axis in% or Deceleration units
	Decel units	BOOLEAN	immediate	units for the Deceleration value: 0 = units per sec^2; 1 =% of maximum acceleration
	Profile	UDINT	immediate	velocity profile to run for the move: 0 = Trapezoidal; 1 = S-curve
	Merge	BOOL	immediate	instructs the motion control to turn all current axis motion, regardless of the motion instructions currently in process, into a pure move governed by this instruction
	Merge speed	DINT	immediate	determines whether the speed of the move profile is going to be the specified Speed value of this instruction or the Current axis speed: • 0 = programmed value in the speed field • 1 = current axis speed
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

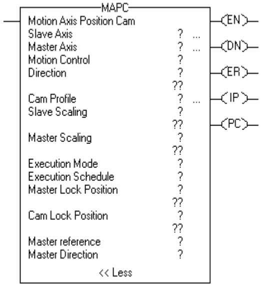
MAOC
Motion Arm
Output Cam
	 MAOC
	Motion Arm Output Cam
	Axis ?
	Execution Target ? ??
	Motion Control ?
	Output ? ??
	Input ? ??
	Output Cam ?
	Cam Start Position ? ??
	Cam End Position ? ??
	Output Compensation ?
	Execution Mode ?
	Execution Schedule ?
	Axis Arm Position ? ??
	Cam Arm Position ? ??
	Position Reference ?
	<< Less
	EN
	DN
	ER
	IP
	PC
		not available	`MAOC(Axis,ExecutionTarget,`
			`MotionControl,Output,Input,`
			`OutputCam,CamStartPosition,`
			`CamEndPosition,`
			`OutputCompensation,`
			`ExecutionMode,`
			`ExecutionSchedule,`
			`AxisArmPosition,`
			`CamArmPosition,Reference);`
				The MAOC instruction sets and resets output bits based on an axis position.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Execution Target	UNIT32	immediate tag	defines the specific output cam: • 0...8 – Output Cams executed in the Logix controller. • 9...31 – Reserved for future use.
Motion Control	MOTION_INSTRUCTION	tag	motion structure

continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MAOC **Motion Arm** **Output Cam** (continued)	Output	DINT	tag	32 output bits that are set or reset based on the specified output cam
	Input	DINT	tag	32 input bits that can be used as enable bits depending on the specified output cam
	Output Cam	OUTPUT_CAM	array tag	array of OUTPUT_CAM elements
	Cam Start Position	SINT DINT INT REAL	immediate tag	cam start position with the cam end position define the left and right boundaries of the output cam range
	Cam End Position	SINT DINT INT REAL	immediate tag	cam end position with the cam start position define the left and right boundaries of the output cam range
	Output Compensation	OUTPUT_ COMPENSATION	array tag	array of 1 to 32 OUTPUT_COMPENSATION elements
	Execution Mode	UINT32	immediate	execution mode: once (0); continuous (1); persistent (2)
	Execution Schedule	UINT32	immediate	when to arm the output cam: 0 = immediate; 1 = pending; 2 = forward only; 3 = reverse only; 4 = bi-directional
	Axis Arm Position	SINT DINT INT REAL	immediate tag	axis position where the output cam is armed when the execution schedule is set to forward only, reverse only, or bi-directional and the axis moves in the specified direction
	Cam Arm Position	SINT DINT INT REAL	immediate tag	cam position associated with the axis arm position when the output cam is armed
	Reference	UINT32	immediate	whether the output cam is connected to either 0 = actual position, 1 = command position
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

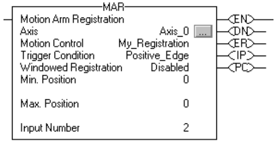
MAPC
Motion Axis Position Cam
	 MAPC
	Motion Axis Position Cam
	Slave Axis ? ...
	Master Axis ? ...
	Motion Control ?
	Direction ?
	??
	Cam Profile ? ...
	Slave Scaling ?
	??
	Master Scaling ?
	??
	Execution Mode ?
	Execution Schedule ?
	Master Lock Position ?
	??
	Cam Lock Position ?
	??
	Master reference ?
	Master Direction ?
	<< Less
	EN
	DN
	ER
	IP
	PC
		not available	`MAPC(SlaveAxis,MasterAxis,`
			`MotionControl,Direction,`
			`CamProfile,SlaveScaling,`
			`MasterScaling,`
			`ExecutionMode,`
			`ExecutionSchedule,`
			`MasterLockPosition,`
			`CamLockPosition,`
			`MasterReference,`
			`MasterDirection);`
				The MAPC instruction provides electronic camming between any two axes according to the specified cam profile.

Operand:	Type:	Format:	Description:
Slave Axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Master Axis	AXIS_FEEDBACK AXIS_CONSUME AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	axis that the slave axis follows according to the cam profile
Motion Control	MOTION_ INSTRUCTION	tag	motion structure

continued

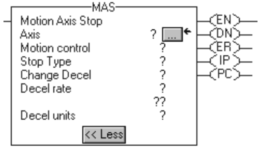
Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MAPC Motion Axis Position Cam (continued)	Direction	UINT32	immediate tag	relative direction of the slave axis: same, opposite, revers, or unchanged
	Cam Profile	CAM_PROFILE	array	calculated cam profile array used to establish the master/slave position relationship
	Slave Scaling	REAL	immediate tag	scales the total distance covered by the slave axis through the cam profile
	Master Scaling	REAL	immediate tag	scales the total distance covered by the master axis through the cam profile
	Execution Mode	UINT32	immediate	determines if the cam profile is executed: 0 = once, 1 = continuous, 2 = persistent
	Execution Schedule	UINT32	immediate	method to execute the cam profile: 0 = immediate, 1 = pending, 2 = forward only, 3 = reverse only, 4 = bi-directional
	Master Lock Position	REAL	immediate tag	master axis absolute position where the slave axis locks to the master axis
	Cam Lock Position	REAL	immediate tag	starting location in the cam profile
	Master Reference	UINT32	immediate	master position reference: 0 = actual position, 1 = command position
	Master Direction	UINT32	immediate	direction of the master axis that generates slave motion according to the cam profile: bi-directional (0), forward only (1), reverse only (2)
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MAR
Motion Arm Registration
	
	MAR
	Motion Arm Registration
	Axis Axis_0
	Motion Control My_Registration
	Trigger Condition Positive_Edge
	Windowed Registration Disabled
	Min. Position 0
	Max. Position 0
	Input Number 2
	
		not available	`MAR(Axis,MotionControl,`
			`TriggerCondition,`
			`WindowedRegistration,`
			`MinimumPosition,`
			`MaximumPosition,`
			`InputNumber);`
				The MAR instruction arms servo-module registration event-checking for the specified axis.

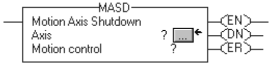
Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion control	MOTION_INSTRUCTION	tag	motion structure
Trigger condition	BOOL	immediate	registration input transition trigger: 0 = on positive edge, 1 = on negative edge
Windowed registration	BOOL	immediate	whether registration is to be windowed, meaning that the computed registration position must fall within the specified minimum and maximum position limits
Minimum position	REAL	immediate or tag	registration position must be greater than minimum position limit
Maximum position	REAL	immediate or tag	registration position must be less than maximum position limit
Input Number	UINT32	1 or 2	registration input: 1 = Registration 1 Position, 2 = Registration 2 Position

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAS Motion Axis Stop	 MAS Motion Axis Stop Axis ? Motion control ? Stop Type ? Change Decel ? Decel rate ? ?? Decel units ? << Less EN DN ER IP PC	not available	`MAS(Axis,MotionControl, StopType,ChangeDecel, DecelRate,DecelUnits);`	The MAS instruction initiates a controlled stop of any motion process on the designated axis.

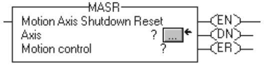
Operand:	Type:	Format:	Description:
Axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Stop type	UNIT32	immediate	determines motion process: 0 = stop all motion; 1 = stop jogging; 2 = stop moving; 3 = stop gearing; 4 = stop homing 5 = stop tuning; 6 = stop test; 7 = stop position camming; 8 = stop time camming; 9 = stop a Master Offset Move
Change Decel	BOOL	immediate	set to enable use of Decel value rather then the current configured Max Deceleration rate
Decel rate	REAL	immediate tag	deceleration rate of the axis in% or Deceleration Units
Decel units	BOOL	immediate	engineering units for Decel value: 0 = units per sec^2; 1 =% of maximum

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MASD **Motion Axis Shutdown**	 MASD Motion Axis Shutdown Axis ? Motion control ? EN DN ER	not available	`MASD(Axis,MotionControl);`	The MASD instruction forces a specified axis into the Shutdown state. The Shutdown state of an axis is when the drive output is disabled, servo loop deactivated, and any available or associated OK solid-state relay contacts are open. The axis remains in the Shutdown state until either an Axis or Group Shutdown Reset is executed.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

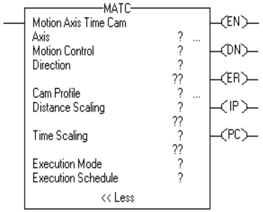
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MASR Motion Axis Shutdown Reset	 MASR Motion Axis Shutdown Reset Axis ? Motion control ? EN DN ER	not available	`MASR(Axis,MotionControl);`	The MASR instruction transitions an axis from an existing Shutdown state to an Axis Ready state.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

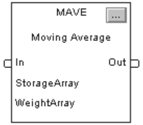
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MATC
Motion Axis Time Cam
	
	MATC
	Motion Axis Time Cam
	Axis ? ...
	Motion Control ?
	Direction ?
	??
	Cam Profile ? ...
	Distance Scaling ?
	??
	Time Scaling ?
	??
	Execution Mode ?
	Execution Schedule ?
	<< Less
	EN
	DN
	ER
	IP
	PC
		not available	`MATC(Axis,MotionControl,`
			`Direction,CamProfile,`
			`DistanceScaling,`
			`TimeScaling,`
			`ExecutionMode,`
			`ExecutionSchedule);`
				The MATC instruction provides electronic camming of an axis as a function of time, according to the specified Cam Profile.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion Control	MOTION_INSTRUCTION	tag	motion structure
Direction	UINT32	immediate tag	relative direction of the slave axis to the master axis: same, opposite, reverse, unchanged
Cam Profile	CAM_PROFILE	array	calculated cam profile array
Distance Scaling	REAL	immediate tag	scales the total distance covered by the axis through the cam profile

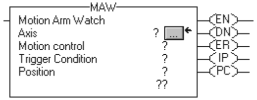
continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MATC Motion Axis Time Cam (continued)	Time Scaling	REAL	immediate tag	scales the time interval covered by the cam profile
	Execution Mode	UINT32	immediate	how the cam motion behaves when the time moves beyond the end point of the cam profile: once (0), continuous (1)
	Execution Schedule	UNIT32	immediate	method to execute the cam profile: 0 = immediate, 1 = pending
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAVE Moving Average	not available	 MAVE ... Moving Average In Out StorageArray WeightArray	`MAVE(MAVE_tag,storage, weight);`	The MAVE instruction calculates a time average value for the In signal. This instruction optionally supports user-specified weights.

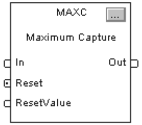
<table>
<tr><th>Operand:</th><th>Type:</th><th>Format:</th><th colspan="3">Description:</th></tr>
<tr><td rowspan="4">MAVE tag</td><td rowspan="4">MOVING_
AVERAGE</td><td rowspan="4">structure</td><td colspan="3">MAVE structure (default parameters):</td></tr>
<tr><th>Parameter:</th><th>Type:</th><th>Description:</th></tr>
<tr><td>In</td><td>REAL</td><td>analog signal input to the instruction</td></tr>
<tr><td>Out</td><td>REAL</td><td>calculated output of the algorithm</td></tr>
<tr><td>storage</td><td>REAL</td><td>array</td><td colspan="3">holds the moving average samples; this array must be at least as large as NumberOfSamples</td></tr>
<tr><td>weight</td><td>REAL</td><td>array</td><td colspan="3">(optional) used for weighted averages; this array must be at least as large as NumberOfSamples
element [0] is used for the newest sample; element [n] is used for the oldest sample</td></tr>
</table>

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAW Motion Arm Watch	 MAW Motion Arm Watch Axis ? Motion control ? Trigger Condition ? Position ? ?? EN DN ER IP PC	not available	`MAW(Axis,MotionControl,TriggerCondition,Position);`	The MAW instruction arms watch-position event-checking for the specified axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Trigger condition	BOOL	immediate	watch-event trigger condition: 0 = forward; 1 = reverse
Position	REAL	immediate tag	new value for the watch position

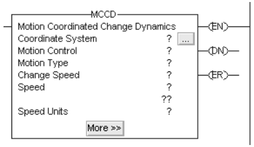
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MAXC Maximum Capture	not available		`MAXC(MAXC_tag);`	The MAXC instruction finds the maximum of the Input signal over time.

Operand:	Type:	Format:	Description:
MAXC tag	MAXIMUM_ CAPTURE	structure	MAXC structure (default parameters):

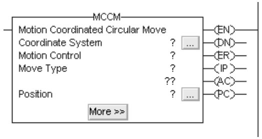
Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Reset	BOOL	request to reset control algorithm the instruction sets Out = ResetValue as long as Reset is set
ResetValue	REAL	reset value for instruction the instruction sets Out = ResetValue as long as Reset is set
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCCD Motion Coordinated Change Dynamics	 MCCD Motion Coordinated Change Dynamics Coordinate System ? Motion Control ? Motion Type ? Change Speed ? Speed ? ?? Speed Units ? More >> EN DN ER	not available	MCCD(CoordinateSystem, MotionControl,MotionType, ChangeSpeed,Speed, SpeedUnits);	The MCCD instruction initiates a change in path dynamics for coordinate motion active on the specified coordinate system

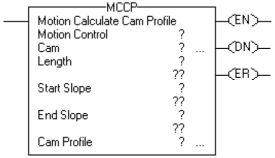
Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure
Motion Type	SINT DINT INT	immediate	1 = coordinated move
Change Speed	SINT DINT INT	immediate tag	whether to change speed: 0 = no; 1 = yes
Speed	SINT DINT INT REAL	immediate tag	coordination units
Speed Units	SINT DINT INT	immediate	0 = units per second; 1 = % of maximum

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCCM Motion Coordinated Circular Move	 MCCM Motion Coordinated Circular Move Coordinate System ? Motion Control ? Move Type ? ?? Position ? More >> EN DN ER IP AC PC	not available	`MCCM(CoordinateSystem, MotionControl,MotionType, Position);`	The MCCM instruction initiates a 2- or 3-dimensional circular coordinated move for the specified axes within the coordinate system.

Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure
Motion Type	SINT DINT INT	immediate tag	type of move: 0 = absolute; 1 = incremental
Position	REAL	array	coordination units

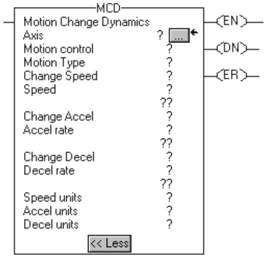
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCCP Motion Calculate Cam Profile		not available	`MCCP(MotionControl,Cam, Length,StartSlope,EndSlope, CamProfile);`	The MCCP instruction calculates a cam profile based on an array of cam points.

Operand:	Type:	Format:	Description:
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Cam	CAM	array	cam array
Length	UINT	immediate tag	number of cam elements in the array
Start Slope	REAL	immediate tag	boundary condition for the initial slope of the profile
End Slope	REAL	immediate tag	boundary condition for the ending slope of the profile
Cam Profile	CAM_PROFILE	array	calculated cam profile array

Arithmetic Status Flags:	Major Faults:
not affected	none

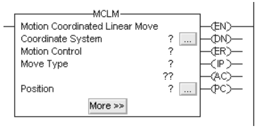
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MCD
Motion Change Dynamics
	
	MCD
	Motion Change Dynamics
	Axis ? ...
	Motion control ?
	Motion Type ?
	Change Speed ?
	Speed ?
	??
	Change Accel ?
	Accel rate ?
	??
	Change Decel ?
	Decel rate ?
	??
	Speed units ?
	Accel units ?
	Decel units ?
	<< Less
	EN
	DN
	ER
		not available	`MCD(Axis,MotionControl,`
			`MotionType,ChangeSpeed,`
			`Speed,ChangeAccel,`
			`AccelRate,ChangeDecel,`
			`DecelRate,SpeedUnits,`
			`AccelUnits,DecelUnits);`
				The MCD instruction selectively changes the speed, acceleration rate, or deceleration rate of a move profile or a jog profile in process

Operand:	Type:	Format:	Description:
Axis	AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion control	MOTION_INSTRUCTION	tag	motion structure
Motion type	UDINT	immediate	motion profile to change: 0 = jog; 1 = move
Change speed	BOOL	immediate	whether to enable a change of speed
Speed	REAL	immediate tag	new Speed to move the axis in% or Speed Units

continued

Instruction:	Relay Ladder:		Function Block:	Structured Text: Description:
MCD **Motion Change Dynamics** (continued)	Change accel	BOOL	immediate	whether to enable an acceleration change
	Accel rate	REAL	immediate tag	acceleration rate of the axis in% or Acceleration units
	Change decel	BOOL	immediate	whether to enable a deceleration change
	Decel rate	REAL	immediate tag	deceleration rate of the axis in% or Deceleration units
	Speed units	BOOL	immediate	units used to display the Speed value: 0 = units per sec; 1 =% of maximum speed
	Accel units	BOOL	immediate	units used to display the Acceleration value: 0 = units per sec^2; 1 =% of maximum acceleration
	Decel units	BOOL	immediate	units used to display the Deceleration value: 0 = units per sec^2; 1 =% of maximum acceleration
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCLM **Motion Coordinated Linear Move**	 MCLM Motion Coordinated Linear Move Coordinate System ? Motion Control ? Move Type ? ?? Position ? ... More >> EN DN ER IP AC PC	not available	`MCLM(CoordinateSystem, MotionControl,MotionType, Position);`	The MCLM instruction initiates a single- or multi-dimensional linear coordinated move for the specified axes within the coordinate system.

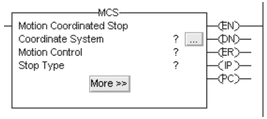
Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure
Motion Type	SINT DINT INT	immediate tag	type of move: 0 = absolute; 1 = incremental
Position	REAL	array	coordination units

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCR **Master Control Reset**	—(MCR)—	not available	not available	The MCR instruction, used in pairs, creates a program zone that can disable all rungs within the MCR instructions.

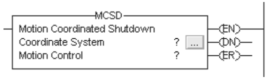
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

**MCS
Motion Coordinated Stop**
	 MCS
	Motion Coordinated Stop
	Coordinate System ?
	Motion Control ?
	Stop Type ?
	...
	More >>
	EN
	DN
	ER
	IP
	PC
		not available	`MCS(CoordinateSystem,`
			`MotionControl,StopType);`
				The MCS instruction initiates a controlled stop of the coordinated motion profile.

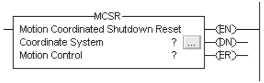
Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure
Stop Type	SINT DINT INT	immediate	type of stop: 2 = coordinated move

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCSD Motion Coordinated Shutdown	 MCSD Motion Coordinated Shutdown Coordinate System ? ... Motion Control ? EN DN ER	not available	`MCSD(CoordinateSystem, MotionControl);`	The MCSD instruction initiates a controlled shutdown of all axes in the specified coordinate system.

Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure

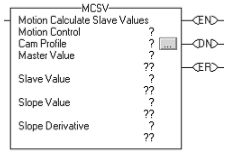
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MCSR Motion Coordinated Shutdown Reset	 MCSR Motion Coordinated Shutdown Reset Coordinate System ? ... Motion Control ? EN DN ER	not available	`MCSR(CoordinateSystem, MotionControl);`	The MCSD instruction resets all axes in the specified coordinate system.

Operand:	Type:	Format:	Description:
Coordinate System	COORDINATE_ SYSTEM	tag	coordinate group of axes
Motion Control	MOTION_ INSTRUCTION	tag	motion structure

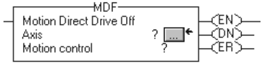
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MCSV
Motion
Calculate
Slave Value
	 MCSV
	Motion Calculate Slave Values
	Motion Control ?
	Cam Profile ? ...
	Master Value ?
	??
	Slave Value ?
	??
	Slope Value ?
	??
	Slope Derivative ?
	??
	EN
	DN
	ER
		not available	`MCSV(MotionControl,`
			`CamProfile,MasterValue,`
			`SlaveValue,SlopValue,`
			`SlopeDerivative`
				The MCSV instruction calculates the slave value, the slope value, and the derivative of the slope for a given cam profile and master value. As an extension to the position and time camming functionality it supplies the values essential for the recovery from faults during camming operations.

Operand:	Type:	Format:	Description:
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Cam profile	CAM_PROFILE	array	defines the cam profile used in calculating the slave values
Master value	SINT DINT INT REAL	immediate or tag	value along the master axis of the cam profile that is used in calculating the slave values
Slave value	REAL	tag	value along the slave axis of the cam profile with the master at the specified master value
Slope value	REAL	tag	first derivative of the value along the slave axis of the cam profile with the master at the specified master value
Slope derivative	REAL	tag	second derivative of the value along the slave axis of the cam profile with the master at the specified master value

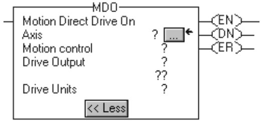
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MDF **Motion Direct Drive Off**	 MDF Motion Direct Drive Off Axis Motion control ? ? ... EN DN ER	not available	`MDF(Axis,MotionControl);`	The MDF instruction deactivates the servo drive and sets the servo output voltage to the output offset voltage.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_SERVO	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

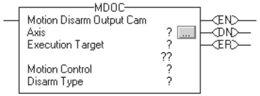
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MDO
Motion Direct Drive On
	 MDO
	Motion Direct Drive On
	Axis ?
	Motion control ?
	Drive Output ?
	??
	Drive Units ?
	<< Less
	EN
	DN
	ER
		not available	`MDO(Axis,MotionControl,`
			`DriveOutput,DriveUnits);`
				The MDO instruction works in conjunction with motion modules that support an external analog servo drive interface. The MDO instruction activates the module's Drive Enable, enabling the external servo drive, and also sets the servo module's output voltage of the drive to the specified voltage level.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_SERVO	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Drive Output	REAL	tag	voltage to output in% of servo output limit or in volts
Drive Units	BOOL	tag	units for drive output value: 0 = volts, 1 = %

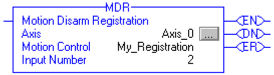
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MDOC **Motion Disarm Output Cam**		not available	`MDOC(Axis,ExecutionTarget,MotionControl,DisarmType);`	The MDOC instruction initiates the disarming of one or more output cams connected to the specified axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_CONSUME AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Execution Target	SINT DINT INT	immediate tag	output cam from the set connected to the named axis: • 0...8 – Output Cams executed in the Logix controller. • 9...31 – Reserved for future use.
Motion Control	MOTION_ INSTRUCTION	tag	motion structure
Disarm Type	DINT	immediate	output cam(s) to be disarmed: 0 = all, 1 = specific

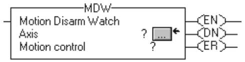
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MDR
Motion Disarm Registration
	MDR
	Motion Disarm Registration
	Axis Axis_0 ...
	Motion Control My_Registration
	Input Number 2
	EN
	DN
	ER
		not available	`MDR(Axis,MotionControl,`
			`InputNumber);`
				The MDR instruction disarms the registration input event-checking for the specified axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	tag	name of the axis
Motion control	MOTION_INSTRUCTION	tag	motion structure
Input Number	UINT32	1 or 2	registration input: 1 = Registration 1 Position, 2 = Registration 2 Position

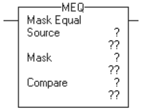
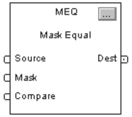
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MDW **Motion Disarm Watch**	 MDW Motion Disarm Watch Axis ? Motion control ? EN DN ER	not available	`MDW(Axis,MotionControl);`	The MDW instruction disarms watch-position event-checking for an axis.

Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

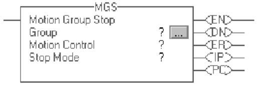
MEQ
Masked Equal To
	 MEQ
	Mask Equal
	Source
	?
	??
	Mask
	?
	??
	Compare
	?
	??
		 MEQ
		...
		Mask Equal
		Source
		Dest
		Mask
		Compare
			`IF (Source AND Mask) =`
			`(Compare AND Mask) THEN`
			`    <statement>;`
			`END_IF;`
				The MEQ instruction passes the Source and Compare values through a Mask and compares the results.

Relay Ladder and Structured Text	Operand:	Type:	Format:	Description:
	Source	SINT DINT INT	immediate tag	value to test against Compare
	Mask	SINT DINT INT	immediate tag	defines which bits to block or pass
	Compare	SINT DINT INT	immediate tag	value to test against Source

Function Block	Operand:	Type:	Format:	Description:
	MEQ tag	FBD_MASK_ EQUAL	structure	MEQ structure (default parameters):

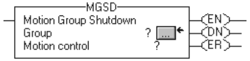
Parameter:	Type:	Description:
Source	DINT	value to test against Compare
Mask	DINT	defines which bits to block (mask)
Compare	DINT	compare value
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MGS **Motion Group Stop**	 MGS Motion Group Stop Group ? ... Motion Control ? Stop Mode ? EN DN ER IP PC	not available	`MGS(Group,MotionControl, StopMode);`	The MGS instruction initiates a stop of all motion in progress on all axes in the specified group by a method configured individually for each axis or as a group via the stop mode of the MGS instruction.

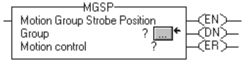
Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group of axes
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Stop Mode	UDINT	immediate	how the axes in the group are stopped: 0 = programmed, 1 = fast stop, 2 = fast disable

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MGSD **Motion Group Shutdown**	 MGSD Motion Group Shutdown Group ? ... Motion control ? EN DN ER	not available	`MGSD(Group,MotionControl);`	The MGSD instruction forces all axes in the designated group into a Shutdown state.

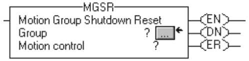
Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group of axes
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MGSP Motion Group Strobe Position		not available	`MGSP(Group,MotionControl);`	The MGSP instruction latches the current command and actual position of all axes in the specified group at a single point in time.

Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group of axes
Motion control	MOTION_ INSTRUCTION	tag	motion structure

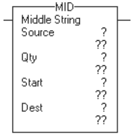
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MGSR Motion Group Shutdown Reset		not available	`MGSR(Group,MotionControl);`	The MGSR instruction transitions a group of axes from the shutdown operating state to the axis ready operating state.

Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group of axes
Motion control	MOTION_ INSTRUCTION	tag	motion structure

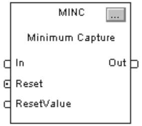
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MID
Middle String
	 MID
	Middle String
	Source ?
	??
	Qty ?
	??
	Start ?
	??
	Dest ?
	??
		not available	`MID(Source,Qty,`
			`Start,Dest);`
				The MID instruction copies a specified number of ASCII characters from a string and stores them in another string.

Operand:	Type:	Format:	Description:
Source	string	tag	string to copy characters from
Quantity	SINT DINT INT	immediate tag	number of characters to copy; the Start plus the Quantity must be less than or equal to the DATA size of the Source
Start	SINT DINT INT	immediate tag	position of the first character to copy; enter a number between 1 and the DATA size of the Source
Destination	string	tag	string to copy the characters to

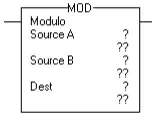
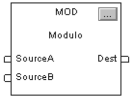
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag • in the LEN value, you entered the number of characters that the string contains
	Type 4	Code 56	The Start or Quantity value is invalid. Check that the: • Start value is between 1 and the DATA size of the Source • Start value plus the Quantity value is less than or equal to the DATA size of the Source

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MINC **Minimum** **Capture**	not available		`MINC(MINC_tag);`	The MINC instruction finds the minimum of the Input signal over time.

Operand:	Type:	Format:	Description:
MINC tag	MINIMUM_ CAPTURE	structure	MINC structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Reset	BOOL	request to reset control algorithm the instruction sets Out = ResetValue as long as Reset is set
ResetValue	REAL	reset value for instruction the instruction sets Out = ResetValue as long as Reset is set.
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MOD **Modulo**	 MOD Modulo Source A ? ?? Source B ? ?? Dest ? ??		`dest := sourceA MOD sourceB;`	The MOD instruction divides Source A by Source B and places the remainder in the Destination.

Relay Ladder and Structured Text

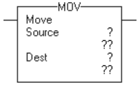
Operand:	Type:	Format:	Description:
Source A	SINT DINT INT REAL	immediate tag	value of the dividend
Source B	SINT DINT INT REAL	immediate tag	value of the divisor
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
MOD tag	FBD_MATH	structure	MOD structure (default parameters):

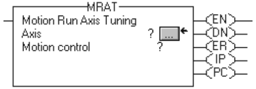
Parameter:	Type:	Description:
SourceA	REAL	value of the dividend
SourceB	REAL	value of the divisor
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 4	the divisor is 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MOV **Move**		not available	`dest := source;`	The MOV instruction copies the Source to the Destination. The Source remains unchanged.

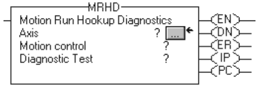
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	value to move (copy)
Destination	SINT DINT INT REAL	tag	an expression consisting of tags and/or immediate values separated by operators

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MRAT **Motion Run** **Axis Tuning**	 MRAT Motion Run Axis Tuning Axis ? ... Motion control ? EN DN ER IP PC	not available	`MRAT(Axis,MotionControl);`	The MRAT instruction commands the motion module to run a tuning profile for the specified axis.

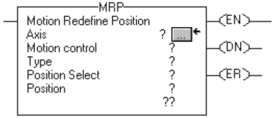
Operand:	Type:	Format:	Description:
Axis	AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MRHD **Motion Run Hookup Diagnostics**	MRHD Motion Run Hookup Diagnostics Axis ? Motion control ? Diagnostic Test ? EN DN ER IP PC	not available	`MRHD(Axis,MotionControl, DiagnosticTest);`	The MRHD instruction commands the motion module to run any one of three different diagnostics on the specified axis.

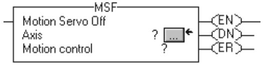
Operand:	Type:	Format:	Description:
Axis	AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Diagnostic test	DINT	immediate	test for the motion module to run: • 0 = motor/encoder hookup test • 1 = encoder hookup test • 2 = encoder marker test

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MRP Motion Redefine Position	 MRP Motion Redefine Position Axis ? Motion control ? Type ? Position Select ? Position ? ?? ... EN DN ER	not available	`MRP(Axis,MotionControl, Type,PositionSelect, Position);`	The MRP instruction changes the command or actual position of an axis.

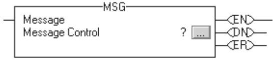
Operand:	Type:	Format:	Description:
Axis	AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Type	BOOL	immediate	how the redefinition operation should work: 0 = absolute, 1 = relative
Position select	BOOL	immediate	what position to perform the redefinition operation on: 0 = actual position, 1 = command position
Position	REAL	immediate tag	value to use to change the axis position to or offset to current position

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MSF Motion Servo Off		not available	`MSF(Axis,MotionControl);`	The MSF instruction deactivates the drive output for the specified axis and to deactivate the axis' servo loop. If you execute an MSF instruction while the axis is moving, the axis coasts to an uncontrolled stop.

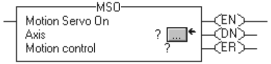
Operand:	Type:	Format:	Description:
Axis	AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MSG Message		not available	`MSG(MessageControl);`	The MSG instruction asynchronously reads or writes a block of data to another module on a network.

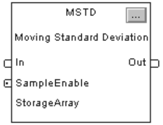
Operand:	Type:	Format:	Description:
message control	MESSAGE	tag	message structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MSO **Motion Servo On**	 MSO Motion Servo On Axis ? Motion control ? EN DN ER	not available	`MSO(Axis,MotionControl);`	The MSO instruction activates the drive amplifier for the specified axis and to activate the axis' servo control loop.

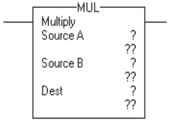
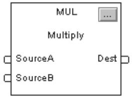
Operand:	Type:	Format:	Description:
Axis	AXIS_GENERIC AXIS_SERVO AXIS_SERVO_ DRIVE	tag	name of the axis
Motion control	MOTION_ INSTRUCTION	tag	motion structure

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MSTD Moving Standard Deviation	not available	 MSTD ... Moving Standard Deviation In Out SampleEnable StorageArray	`MSTD(MSTD_tag,storage);`	The MSTD instruction calculates a moving standard deviation and average for the In signal.

<table>
<tr><th>Operand:</th><th>Type:</th><th>Format:</th><th>Description:</th></tr>
<tr><td>MSTD tag</td><td>MOVING_STD_ DEV</td><td>structure</td><td>MSTD structure (default parameters):
<table>
<tr><th>Parameter:</th><th>Type:</th><th>Description:</th></tr>
<tr><td>In</td><td>REAL</td><td>analog signal input to the instruction</td></tr>
<tr><td>SampleEnable</td><td>BOOL</td><td>enable for taking a sample of In
When set, the instruction enters the value of In into the storage array and calculates a new Out and Average value.
When cleared and Initialize is cleared, the instruction holds Out and Average at their current values.</td></tr>
<tr><td>Out</td><td>REAL</td><td>calculated output of the algorithm</td></tr>
</table></td></tr>
<tr><td>storage</td><td>REAL</td><td>array</td><td>holds the In samples; this array must be at least as large as NumberOfSamples</td></tr>
</table>

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MUL **Multiply**	 MUL Multiply Source A ? ?? Source B ? ?? Dest ? ??		`dest := sourceA * sourceB;`	The MUL instruction multiplies Source A with Source B and places the result in the Destination.

Relay Ladder and Structured Text

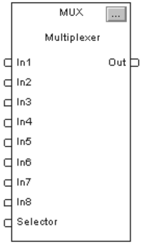
Operand:	Type:		Format:	Description:
Source A	SINT INT	DINT REAL	immediate tag	value of the multiplicand
Source B	SINT INT	DINT REAL	immediate tag	value of the multiplier
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
MUL tag	FBD_MATH	structure	MUL structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value of the multiplicand
SourceB	REAL	value of the multiplier
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

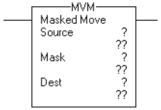
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MUX Multiplexer	not available		not available	The MUX instruction selects one of eight inputs based on the selector input.

Operand:	Type:	Format:	Description:
MUX tag	MULTIPLEXER	structure	MUX structure (default parameters):

Parameter:	Type:	Description:
In*x*	REAL	analog signal input to the instruction where *x* = 1-8
Selector	DINT	selector input to the instruction
Out	REAL	selected output of the algorithm

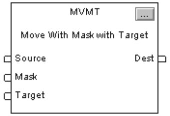
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

MVM
Masked Move
	 MVM
	Masked Move
	Source ?
	??
	Mask ?
	??
	Dest ?
	??
		see MVMT	`dest := (Dest AND NOT (Mask))`
			`OR (Source AND Mask);`
				The MVM instruction copies the Source to a Destination and allows portions of the data to be masked.

Operand:	Type:	Format:	Description:
Source	SINT DINT INT	immediate tag	value to move
Mask	SINT DINT INT	immediate tag	which bits to block or pass
Destination	SINT DINT INT	tag	an expression consisting of tags and/or immediate values separated by operators

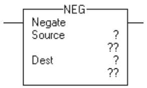
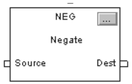
Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
MVMT Masked Move with Target	see MVM		`MVMT(MVMT_tag);`	The MVMT instruction first copies the Target to the Destination. Then the instruction compares the masked Source to the Destination and makes any required changes to the Destination. The Target and the Source remain unchanged.

Operand:	Type:	Format:	Description:
MVMT tag	FBD_MASKED_ MOVE	structure	MVMT structure (default parameters):

Parameter:	Type:	Description:
Source	DINT	input value to move to Destination based on value of Mask
Mask	DINT	mask of bits to move from Source to Dest. All bits set to one cause the corresponding bits to move from Source to Dest. All bits that are set to zero cause the corresponding bits not to move from Source to Dest
Target	DINT	input value to move to Dest prior to moving Source bits through the Mask
Dest	DINT	result of masked move instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
NEG **Negate**			`dest := -source;`	The NEG instruction changes the sign of the Source and places the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	value to negate
Destination	SINT INT	DINT REAL	tag	tag to store the result

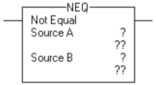
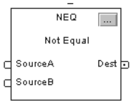
Function Block

Operand:	Type:	Format:	Description:
NEG tag	FBD_MATH_ ADVANCED	structure	NEG structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	value to negate
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

NEQ
Not Equal To
			`IF sourceA <> sourceB THEN`
			`      <statements>;`
				The NEQ instruction tests whether Source A is not equal to Source B.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source A	SINT INT DINT	REAL string	immediate tag	value to test against Source B
Source B	SINT INT DINT	REAL string	immediate tag	value to test against Source A

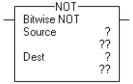
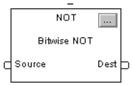
Function Block

Operand:	Type:	Format:	Description:
NEQ tag	FBD_COMPARE	structure	NEQ structure (default parameters):

Parameter:	Type:	Description:
SourceA	REAL	value to test against SourceB
SourceB	REAL	value to test against SourceA
Dest	BOOL	result of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
NOP **No Operation**	-[NOP]-	not available	not available	The NOP instruction functions as a placeholder
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

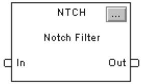
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
NOT **Bitwise NOT**			`dest := NOT source`	The NOT instruction performs a bitwise NOT operation using the bits in the Source and places the result in the Destination.

	Operand:	Type:	Format:	Description:
Relay Ladder and Structured Text	Source	SINT DINT INT	immediate tag	value to NOT
	Destination	SINT DINT INT	tag	tag to store the result

	Operand:	Type:	Format:	Description:
Function Block	NOT tag	FBD_LOGICAL	structure	NOT structure (default parameters):

Parameter:	Type:	Description:
Source	DINT	value to NOT
Dest	DINT	result of the instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
NTCH **Notch Filter**	not available		`NTCH(NTCH_tag);`	The NTCH instruction provides a filter to attenuate input frequencies that are at the notch frequency.

Operand:	Type:	Format:	Description:
NTCH tag	FILTER_NOTCH	structure	NTCH structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

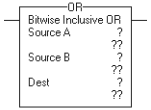
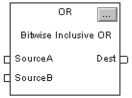
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

ONS
One Shot
	—[ONS]—	not available	`IF` *`BOOL_expression`* `AND NOT storage_bit THEN`
			`<statement>;`
			`END_IF;`
			`storage_bit :=` *`BOOL_expression;`*
				The ONS instruction enables or disables the remainder of the rung, depending on the status of the storage bit.

Operand:	Type:	Format:	Description:
storage bit	BOOL	tag	internal storage bit stores the rung-condition-in from the last time the instruction was executed

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
OR **Bitwise OR**	 OR Bitwise Inclusive OR Source A ? ?? Source B ? ?? Dest ? ??	 OR ... Bitwise Inclusive OR SourceA Dest SourceB	`dest := sourceA OR sourceB`	The OR instruction performs a bitwise OR operation using the bits in Source A and Source B and places the result in the Destination.

Relay Ladder and Structured Text

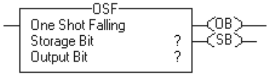
Operand:	Type:	Format:	Description:
Source A	SINT DINT INT	immediate tag	value to OR with Source B
Source B	SINT DINT INT	immediate tag	value to OR with Source A
Destination	SINT DINT INT	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
OR tag	FBD_LOGICAL	structure	OR structure (default parameters):

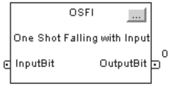
Parameter:	Type:	Description:
SourceA	DINT	value to OR with Source B
SourceB	DINT	value to OR with Source A
Dest	DINT	result of the instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
OSF **One Shot Falling**	 OSF One Shot Falling Storage Bit ? Output Bit ? OB SB	see OSFI	see OSFI	The OSF instruction sets or clears the output bit depending on the status of the storage bit.

Operand:	Type:	Format:	Description:
storage bit	BOOL	tag	internal storage bit stores the rung-condition-in from the last time the instruction was executed
output bit	BOOL	tag	bit to be set

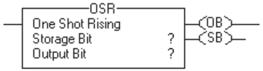
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
OSFI **One Shot Falling with Input**	see OSF	 OSFI ... One Shot Falling with Input InputBit OutputBit 0	`OSFI(OSFI_tag);`	The OSFI instruction sets the OutputBit for one execution cycle when the InputBit toggles from set to cleared.

Operand:	Type:	Format:	Description:
OSFI tag	FBD_ONESHOT	structure	OSFI structure (default parameters):

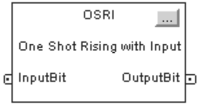
Parameter:	Type:	Description:
InputBit	BOOL	input bit
OutputBit	BOOL	output bit

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
OSR **One Shot Rising**	 OSR One Shot Rising Storage Bit ? Output Bit ? OB SB	see OSRI	see OSRI	The OSR instruction sets or clears the output bit, depending on the status of the storage bit.

Operand:	Type:	Format:	Description:
storage bit	BOOL	tag	internal storage bit stores the rung-condition-in from the last time the instruction was executed
output bit	BOOL	tag	bit to be set

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
OSRI **One Shot Rising with Input**	see OSR	 OSRI ... One Shot Rising with Input InputBit OutputBit	`OSRI(OSRI_tag);`	The OSRI instruction sets the output bit for one execution cycle when the input bit toggles from cleared to set.

Operand:	Type:	Format:	Description:
OSRI tag	FBD_ONESHOT	structure	OSRI structure (default parameters):

Parameter:	Type:	Description:
InputBit	BOOL	input bit
OutputBit	BOOL	output bit

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

OTE
Output
Energize
	—( )—	not available	`data_bit [:=]`
			`BOOL_expression;`
				The OTE instruction sets or clears the data bit.

Operand:	Type:	Format:	Description:
data bit	BOOL	tag	bit to be set or cleared

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

OTL
Output Latch
	—(L)—	not available	`IF` *`BOOL_expression`* `THEN`
			`     data_bit := 1;`
			`END_IF;`
				The OTL instruction sets (latches) the data bit.

Operand:	Type:	Format:	Description:
data bit	BOOL	tag	bit to be set

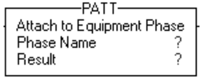
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

OTU
Output Unlatch
	—(U)—	not available	`IF` *`BOOL_expression`* `THEN`
			`     data_bit := 0;`
			`END_IF;`
				The OTU instruction clears (unlatches) the data bit.

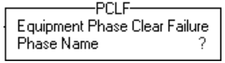
Operand:	Type:	Format:	Description:
data bit	BOOL	tag	bit to be cleared

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PATT **Attach to Equipment Phase**	 PATT Attach to Equipment Phase Phase Name ? Result ?	not available	`PATT(Phase_Name, Result);`	The PATT instruction lets a program take ownership of an equipment phase.

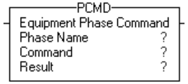
Relay Ladder and Structured Text	Operand:	Type:	Format:	Description:
	Phase Name	phase	name of the equipment phase	Equipment phase that you want to own
	Result	DINT	immediate tag	To let the instruction return a code for its success/failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

	Arithmetic Status Flags:	Major Faults:
	not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PCLF **Equipment Phase Clear Failure**	 PCLF Equipment Phase Clear Failure Phase Name ?	not available	`PCLF(Phase_Name);`	The PCLF instruction clears the failure code for an equipment phase.

Relay Ladder and Structured Text	Operand:	Type:	Format:	Description:
	Phase Name	phase	name of the equipment phase	Equipment phase that you *no longer* want to own

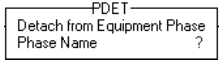
	Arithmetic Status Flags:	Major Faults:
	not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PCMD Equipment Phase Command	 PCMD Equipment Phase Command Phase Name ? Command ? Result ?	not available	`PCMD(PhaseName, Command, Result);`	The PCMD instruction transitions an equipment phase to the next state or substate.

Relay Ladder and Structured Text

Operand:	Type:	Format:	Description:
Phase Name	phase	name of the equipment phase	Equipment phase that you want to change to a different state or substate
Command	command	name of the command	Command that you want to send to the equipment phase to change its state
Result	DINT	immediate tag	To let the instruction return a code for its success/failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PDET Detach from Equipment Phase	 PDET Detach from Equipment Phase Phase Name ?	not available	`PDET(Phase_Name);`	After a program executes a PDET instruction, the program no longer owns the equipment phase. This frees the equipment phase for ownership by another program or by RSBizWare Batch software. Use the PDET instruction only if the program previously took ownership of an equipment phase via an Attach to Equipment Phase (PATT) instruction.

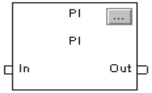
Relay Ladder and Structured Text

Operand:	Type:	Format:	Description:
Phase Name	phase	name of the equipment phase	Equipment phase that you *no longer* want to own

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PFL Equipment Phase Failure	? —<PFL>—	not available	`PFL(Failure_Code);`	The PFL instruction sets the value of the failure code for an equipment phase. Use the instruction to signal a specific failure for an equipment phase, such as a specific device has faulted.

Relay Ladder and Structured Text	**Operand:**	**Type:**	**Format:**	**Description:**
	Failure_Code	DINT	immediate tag	value to which you want to set the failure code for the equipment phase
	Arithmetic Status Flags:		**Major Faults:**	
	not affected		none	

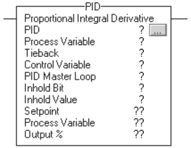
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PI **Proportional + Integral**	not available		`PI(PI_tag);`	The PI instruction provides two methods of operation. The first method follows the conventional PI algorithm in that the proportional and integral gains remain constant over the range of the input signal (error). The second method uses a non-linear algorithm where the proportional and integral gains vary over the range of the input signal. The input signal is the deviation between the setpoint and feedback of the process.

Operand:	Type:	Format:	Description:
PI tag	PROP_INT	structure	PI structure (default parameters):

Parameter:	Type:	Description:
In	REAL	process error signal input
Out	REAL	calculated output of the PI algorithm

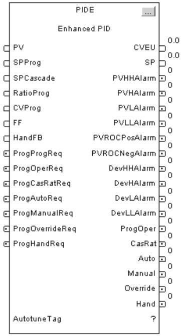
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

PID Proportional, Integral, Derivative	
	PID
	Proportional Integral Derivative
	PID ?
	Process Variable ?
	Tieback ?
	Control Variable ?
	PID Master Loop ?
	Inhold Bit ?
	Inhold Value ?
	Setpoint ??
	Process Variable ??
	Output % ??
		not available	`PID(PID,`
			`ProcessVariable,Tieback,`
			`ControlVariable,`
			`PIDMasterLoop,`
			`InholdBit,`
			`InHoldValue);`
				The PID instruction controls a process variable such as flow, pressure, temperature, or level.

Operand:	Type:	Format:	Description:
PID	PID	structure	PID structure
Process variable	SINT DINT INT REAL	tag	value you want to control
Tieback	SINT DINT INT REAL	immediate tag	*(optional)* output of a hardware hand/auto station which is bypassing the output of the controller Enter 0 if you don't want to use this parameter.
Control variable	SINT DINT INT REAL	tag	value which goes to the final control device (valve, damper, etc.) If you are using the deadband, the Control variable must be REAL or it will be forced to 0 when the error is within the deadband.
PID master loop	PID	structure	(*optional*) PID tag for the master PID Enter 0 if you don't want to use this parameter.
Inhold bit	BOOL	tag	(*optional*) current status of the inhold bit from a 1756 analog output channel to support bumpless restart Enter 0 if you don't want to use this parameter.
Inhold value	SINT DINT INT REAL	tag	(*optional*) data readback value from a 1756 analog output channel to support bumpless restart Enter 0 if you don't want to use this parameter.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PID Proportional, Integral, Derivative (continued)	Setpoint	na	na	displays current value of the setpoint
	Process variable	na	na	displays current value of the scaled process variable
	Output %	na	na	displays current output percentage value
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	Type 4	Code 35	.UPD =0
		Type 4	Code 36	setpoint out of range

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PIDE Enhanced PID	not available		`PIDE(PIDE_tag);`	The PIDE instruction 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.

Operand:	Type:	Format:	Description:
PIDE tag	PIDE_ENHANCED	structure	PIDE structure (default parameters):

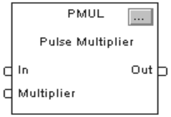
Parameter:	Type:	Description:
PV	REAL	scaled process variable input
SPProg	REAL	SP program value, scaled in PV units
SPCascade	REAL	SP Cascade value, scaled in PV units
RatioProg	REAL	ratio program multiplier.
CVProg	REAL	CV program manual value

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

PIDE
Enhanced PID
(continued)

Parameter:	Type:	Description:
FF	REAL	feed forward value
HandFB	REAL	CV hand feedback value
ProgProgReq	BOOL	program program request
ProgOperReq	BOOL	program operator request
ProgCasRatReq	BOOL	program cascade/ratio mode request
ProgAutoReq	BOOL	program auto mode request
ProgManualReq	BOOL	program manual mode request
ProgOverrideReq	BOOL	program override mode request
ProgHandReq	BOOL	program hand mode request
CVEU	REAL	scaled control variable output
SP	REAL	current setpoint value
PVHHAlarm	BOOL	PV high-high alarm indicator
PVHAlarm	BOOL	PV high alarm indicator
PVLAlarm	BOOL	PV low alarm indicator
PVLLAlarm	BOOL	PV low-low alarm indicator
PVROCPosAlarm	BOOL	PV rate-of-change positive alarm indicator
PVROCNegAlarm	BOOL	PV rate-of-change negative alarm indicator

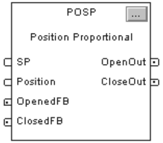
Instruction:	Relay Ladder:		Function Block:		Structured Text:	Description:
PIDE Enhanced PID (continued)				**Parameter:**	**Type:**	**Description:**
				DevHHAlarm	BOOL	deviation high-high alarm indicator
				DevHAlarm	BOOL	deviation high alarm indicator
				DevLAlarm	BOOL	deviation low alarm indicator
				DevLLAlarm	BOOL	deviation low-low alarm indicator
				ProgOper	BOOL	program/operator control indicator set when in program mode; cleared when in operator mode
				CasRat	BOOL	cascade ration mode indicator
				Auto	BOOL	auto mode indicator
				Manual	BOOL	manual mode indicator
				Override	BOOL	override mode indicator
				Hand	BOOL	hand mode indicator
	autotune	PIDE_AUTOTUNE	structure	(optional) autotune structure (function block only)		
	Arithmetic Status Flags:		**Major Faults:**			
	set for the CVEU parameter		none			

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PMUL Pulse Multiplier	not available		`PMUL(PMUL_tag);`	The PMUL instruction provides an interface from a position input module, such as a resolver or encoder feedback module, to the digital system by computing the change in input from one scan to the next. By selecting a specific word size, you configure the PMUL instruction to differentiate through the rollover boundary in a continuous and linear fashion.

Operand:	Type:	Format:	Description:
PMUL tag	PULSE_ MULTIPLIER	structure	PMUL structure (default parameters):

Parameter:	Type:	Description:
In	DINT	analog signal input to the instruction
Multiplier	DINT	multiplier; divide this value by 100,000 to control the ratio of In to Out
Out	REAL	output of the instruction

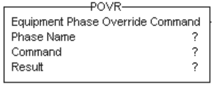
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
POSP **Position Proportional**	not available	 POSP ... Position Proportional SP Position OpenedFB ClosedFB OpenOut CloseOut	`POSP(POSP_tag);`	The POSP instruction 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.

Operand:	Type:	Format:	Description:
POSP tag	POSITION_PROP	structure	POSP structure (default parameters):

Parameter:	Type:	Description:
SP	REAL	setpoint value; must use the same engineering units as Position
Position	REAL	position feedback
OpenedFB	BOOL	opened feedback; when set, the open output is not allowed to turn on
ClosedFB	BOOL	closed feedback; when set, the close output is not allowed to turn on
OpenOut	BOOL	output is pulsed to open the device
CloseOut	BOOL	output is pulsed to close the device

Arithmetic Status Flags:	Major Faults:
set for the PositionPercent parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
POVR Equipment Phase Override Command	 POVR Equipment Phase Override Command Phase Name ? Command ? Result ?	not available	`POVR(PhaseName, Command, Result);`	Gives the hold, stop, or abort command to an equipment phase. Overrides all owners of the equipment phase. The command works even if RSLogix 5000 software, RSBizWare Batch software, or another program already own the equipment phase.

Relay Ladder and Structured Text

Operand:	Type:	Format:	Description:
Phase Name	phase	name of the equipment phase	Equipment phase that you want to change to a different state
Command	command	name of the command	One of these commands for the equipment phase: • hold • stop • abort
Result	DINT	immediate tag	To let the instruction return a code for its success/failure, enter a DINT tag in which to store the result code. Otherwise, enter 0.

Arithmetic Status Flags:	Major Faults:
not affected	none

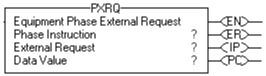
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PPD Equipment Phase Paused	-[PPD]-	not available	`PPD( );`	The PPD instruction lets you stop execution at a specific step (breakpoint) to test and troubleshoot your logic.

Arithmetic Status Flags:	Major Faults:
not affected	none

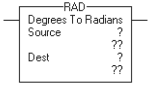
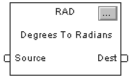
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PRNP Equipment Phase New Parameters		not available	`PRNP( );`	The PRNP instruction clears the NewInputParameters bit of the equipment phase.
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
PSC Phase State Complete		not available	`PSC( );`	The PSC instruction signals the completion of a phase state routine.
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

**PXRQ
Equipment Phase External Request**
	 PXRQ
	Equipment Phase External Request
	Phase Instruction ?
	External Request ?
	Data Value ?
	EN
	ER
	IP
	PC
		not available	`PXRQ(Phase_Instruction,`
			`External_Request,`
			`Data_Value);`
				The PXRQ instruction sends a request to RSBizWare Batch software.

Relay Ladder and Structured Text	**Operand:**	**Type:**	**Format:**	**Description:**
	Phase Instruction	PHASE_INSTRUC TION	tag	tag that controls the operation
	External Request	request	name	type of request
	Data Value	DINT	array tag	parameters of the request
	Arithmetic Status Flags:	**Major Faults:**		
	not affected	none		

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RAD **Radians**			`dest := RAD(source);`	The RAD instruction converts the Source (in degrees) to radians and stores the result in the Destination.

Relay Ladder and Structured Text	Operand:	Type:	Format:	Description:
	Source	SINT DINT INT REAL	immediate tag	value to convert to radians
	Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block	Operand:	Type:	Format:	Description:
	RAD tag	FBD_MATH_ ADVANCED	structure	RAD structure (default parameters):

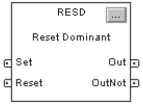
Parameter:	Type:	Description:
Source	REAL	input to the conversion instruction
Dest	REAL	result of the conversion instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RES **Reset**	—(RES)—	not available	not available	The RES instruction resets a TIMER, COUNTER, or CONTROL structure.

Operand:	Type:	Format:	Description:
structure	TIMER CONTROL COUNTER	tag	structure to reset

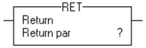
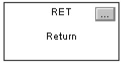
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RESD **Reset** **Dominant**	not available	 RESD ... Reset Dominant Set Out Reset OutNot	`RESD(RESD_tag);`	The RESD instruction uses Set and Reset inputs to control latched outputs. The Reset input has precedence over the Set input.

Operand:	Type:	Format:	Description:
RESD tag	DOMINANT_ RESET	structure	RESD structure (default parameters):

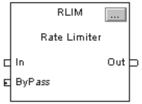
Parameter:	Type:	Description:
Set	BOOL	set input to the instruction
Reset	BOOL	reset input to the instruction
Out	BOOL	output of the instruction
OutNot	BOOL	inverted output of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RET **Return**			`RET(ReturnPar);`	The RET instruction is an optional instruction that exchanges data with the JSR instruction.

Operand:	Type:	Format:	Description:
Return parameter	BOOL DINT SINT REAL INT structure	immediate tag array tag	data from this routine that you want to copy to the corresponding return parameter in the JSR instruction

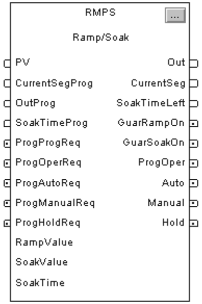
Arithmetic Status Flags:	Major Faults:		
affected	4	31	• JSR instruction has fewer input parameters than SBR instruction • RET instruction has fewer return parameters than JSR instruction • main routine contains a RET instruction
	4	0	JSR instruction jumps to a fault routine

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RLIM **Rate Limiter**	not available		`RLIM(RLIM_tag);`	The RLIM instruction limits the amount of change of a signal over time.

Operand:	Type:	Format:	Description:
RLIM tag	RATE_LIMITER	structure	RLIM structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
ByPass	BOOL	request to bypass the algorithm; when set, Out = In
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RMPS Ramp/Soak	not available	 RMPS Ramp/Soak PV CurrentSegProg OutProg SoakTimeProg ProgProgReq ProgOperReq ProgAutoReq ProgManualReq ProgHoldReq RampValue SoakValue SoakTime Out CurrentSeg SoakTimeLeft GuarRampOn GuarSoakOn ProgOper Auto Manual Hold	`RMPS(RMPS_tag,RampValue, SoakValue,SoakTime);`	The RMPS instruction provides for a number of segments of alternating ramp and soak periods.

Operand:	Type:	Format:	Description:
RMPS tag	RAMP_SOAK	structure	RMPS structure (default parameters):

Parameter:	Type:	Description:
PV	REAL	scaled analog temperature signal input to the instruction
CurrentSegProg	DINT	current segment program value
OutProg	REAL	output program value
SoakTimeProg	REAL	soak time program value
ProgProgReq	BOOL	program program request
ProgOperReq	BOOL	program operator request

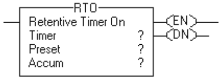
continued

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RMPS Ramp/Soak (continued)				

Parameter:	Type:	Description:
ProgAutoReq	BOOL	program auto mode request
ProgManualReq	BOOL	program manual mode request
ProgHoldReq	BOOL	program hold mode request
Out	REAL	output of the instruction
CurrentSeg	DINT	current segment number
SoakTimeLeft	REAL	soak time left
GuarRampOn	BOOL	guaranteed ramp status
GuarSoakOn	BOOL	guaranteed soak status
ProgOper	BOOL	program/operator control indicator
Auto	BOOL	auto mode indicator
Manual	BOOL	manual mode indicator
Hold	BOOL	hold mode indicator

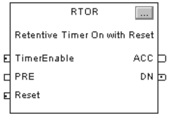
RampValue	REAL	array	ramp value array; enter a ramp value (time in minutes) for each segment (0 to NumberOfSegs-1)
SoakValue	REAL	array	soak value array; enter a soak value for each segment (0 to NumberOfSegs-1); the array must be at least as large as NumberOfSegs
SoakTime	REAL	array	soak time array; enter a soak time (time in minutes) for each segment (0 to NumberOfSegs-1)

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RTO **Retentive** **Timer On**	 RTO Retentive Timer On Timer ? Preset ? Accum ? EN DN	see RTOR	see RTOR	The RTO instruction is a retentive timer that accumulates time when the instruction is enabled.

Operand:	Type:	Format:	Description:
Timer	TIMER	tag	timer structure
Preset	DINT	immediate	how long to delay (accumulate time)
Accum	DINT	immediate	number of msec the timer has counted; initial value is typically 0

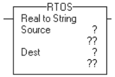
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 34	• .PRE < 0 • .ACC < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RTOR Retentive Timer On with Reset	see RTO		`RTOR(RTOR_tag);`	The RTOR instruction is a retentive timer that accumulates time when TimerEnable is set.

Operand:	Type:	Format:	Description:
RTOR tag	FBD_TIMER	structure	RTOR structure (default parameters):

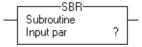
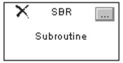
Parameter:	Type:	Description:
TimerEnable	BOOL	if cleared, enables the timer to run and accumulate time
PRE	DINT	timer preset value in 1msec units
Reset	BOOL	request to reset the timer
ACC	BOOL	accumulated time in milliseconds
DN	BOOL	timing done output. Indicates when the ACC ≥ PRE

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
RTOS **REAL to String**		not available	`RTOS(Source,Dest);`	The RTOS instruction produces the ASCII representation of a REAL value.

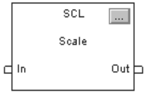
Operand:	Type:	Format:	Description:
Source	REAL	tag	tag that contains the REAL value
Destination	string	tag	tag to store the ASCII value

Arithmetic Status Flags:	Major Faults:		
not affected	4	51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	4	52	The output string is larger than the destination. Create a new string data type that is large enough for the output string. Use the new string data type as the data type for the destination.

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SBR Subroutine			`SBR(InputPar);`	The SBR instruction is an optional instruction that exchanges data with the JSR instruction.

Operand:	Type:	Format:	Description:
Input parameter	BOOL DINT SINT REAL INT structure	tag array tag	tag in this routine into which you want to copy the corresponding input parameter from the JSR instruction

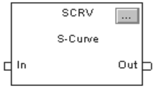
Arithmetic Status Flags:	Major Faults:		
affected	4	31	• JSR instruction has fewer input parameters than SBR instruction • RET instruction has fewer return parameters than JSR instruction • main routine contains a RET instruction
	4	0	JSR instruction jumps to a fault routine

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SCL **Scale**	not available		`SCL(SCL_tag);`	The SCL instruction converts an unscaled input value to a floating point value in engineering units.

Operand:	Type:	Format:	Description:
SCL tag	SCALE	structure	SCL structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	output that represents the scaled value of the analog input

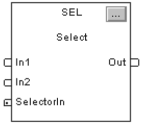
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

<table>
<tr><th>Instruction:</th><th>Relay Ladder:</th><th>Function Block:</th><th>Structured Text:</th><th>Description:</th></tr>
<tr><td><b>SCRV
S-Curve</b></td><td>not available</td><td>
</td><td><code>SCRV(SCRV_tag);</code></td><td>The SCRV instruction performs a ramp function with an added jerk rate. The jerk rate is the maximum rate of change of the rate used to ramp output to input.</td></tr>
</table>

Operand:	Type:	Format:	Description:
SCRV tag	S_CURVE	structure	SCRV structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	output of the instruction

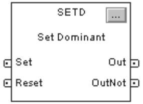
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SEL **Selector**	not available		not available	The SEL instruction uses a digital input to select one of two inputs.

Operand:	Type:	Format:	Description:
SEL tag	SELECT	structure	SEL structure (default parameters):

Parameter:	Type:	Description:
In1	REAL	first analog signal input to the instruction
In2	REAL	second analog signal input to the instruction
SelectorIn	BOOL	input that selects between In1 and In2
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

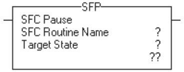
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SETD **Set Dominant**	not available	 SETD ... Set Dominant Set Out Reset OutNot	`SETD(SETD_tag);`	The SETD instruction uses Set and Reset inputs to control latched outputs. The Set input has precedence over the Reset input.

Operand:	Type:	Format:	Description:
SETD tag	DOMINANT_SET	structure	SETD structure (default parameters):

Parameter:	Type:	Description:
Set	BOOL	Set input to the instruction
Reset	BOOL	Reset input to the instruction
Out	BOOL	output of the instruction
OutNot	BOOL	inverted output of the instruction

Arithmetic Status Flags:	Major Faults:
not affected	none

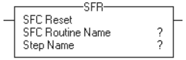
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

SFP
Pause SFC
	 SFP
	SFC Pause
	SFC Routine Name ?
	Target State ?
	??
		not available	`SFP(SFCRoutineName,`
			`TargetState);`
				The SFP instruction pauses an SFC routine.

Operand:	Type:	Format:	Description:
SFCRoutine Name	ROUTINE	name	SFC routine to pause
TargetState	DINT	immediate tag	select executing (enter 0) or paused (enter 1)

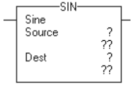
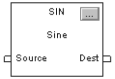
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 85	the routine type is not an SFC routine

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

SFR
Reset SFC
		not available	`SFR(SFCRoutineName`
			`StepName);`
				The SFR instruction resets the execution of a SFC routine at a specified step.

Operand:	Type:	Format:	Description:
SFCRoutine Name	ROUTINE	name	SFC routine to reset
Step Name	SFC_STEP	tag	target step where to resume execution

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 85	the routine type is not an SFC routine
	Type 4	Code 89	specified target step does not exist in the SFC routine

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SIN **Sine**			`dest := SIN(source);`	The SIN instruction takes the sine of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

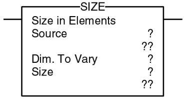
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	find the sine of this value
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
SIN tag	FBD_MATH_ ADVANCED	structure	SIN structure (default parameters):

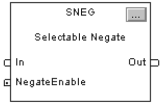
Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SIZE Size in Elements	 SIZE Size in Elements Source ? ?? Dim. To Vary ? Size ? ??	not available	not available	The SIZE instruction finds the size of a dimension of an array.

Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL structure string	array tag	array on which the instruction is to operate
Dimension to vary	DINT	immediate (0, 1, 2)	which dimension to use enter 0 (first dimension), 1 (second dimension), or 2 (third dimension)
Size	SINT DINT INT REAL	tag	tag to store the number of elements in the specified dimension of the array

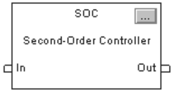
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SNEG Selected Negate	not available		`SNEG(SNEG_tag);`	The SNEG instruction uses a digital input to select between the input value and the negative of the input value.

Operand:	Type:	Format:	Description:
SNEG tag	SELECTABLE_ NEGATE	structure	SNEG structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
NegateEnable	BOOL	when NegateEnable is set, the instruction sets Out to the negative value of In
Out	REAL	calculated output of the algorithm

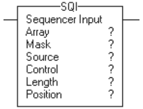
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SOC Second-Order Controller	not available		`SOC(SOC_tag);`	The SOC instruction is designed for use in closed loop control systems in a similar manner to the PI instruction. The SOC instruction provides a gain term, a first order lag, and a second order lead.

Operand:	Type:	Format:	Description:
SOC tag	SEC_ORDER_ CONTROLLER	structure	SOC structure (default parameters):

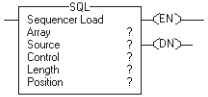
Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SQI Sequencer Input		not available	not available	The SQI instruction detects when a step is complete in a sequence pair of SQO/SQI instructions.

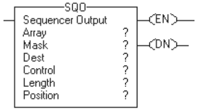
Operand:	Type:	Format:	Description:
Array	DINT	array tag	sequencer array; specify the first element of the sequencer array **do not** use CONTROL.POS in the subscript
Mask	SINT DINT INT	tag immediate	which bits to block or pass
Source	SINT DINT INT	tag	input data for the sequencer array
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the SQO and SQL instructions
Length	DINT	immediate	number of elements in the Array (sequencer table) to compare
Position	DINT	immediate	current position in the array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SQL Sequencer Load	 SQL Sequencer Load Array ? Source ? Control ? Length ? Position ? EN DN	not available	not available	The SQL instruction loads reference conditions into a sequencer array.

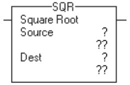
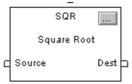
Operand:	Type:	Format:	Description:
Array	DINT	array tag	sequencer array; specify the first element of the sequencer array **do not** use CONTROL.POS in the subscript
Source	SINT DINT INT	tag immediate	input data to load into the sequencer array
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the SQI and SQO instructions
Length	DINT	immediate	number of elements in the Array (sequencer table) to load
Position	DINT	immediate	current position in the array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 20	Length > size of Array

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SQO Sequencer Output	 SQO Sequencer Output Array ? Mask ? Dest ? Control ? Length ? Position ? EN DN	not available	not available	The SQO instruction sets output conditions for the next step of a sequence pair of SQO/SQI instructions.

Operand:	Type:	Format:	Description:
Array	DINT	array tag	sequencer array; specify the first element of the sequencer array **do not** use CONTROL.POS in the subscript
Mask	SINT DINT INT	tag immediate	which bits to block or pass
Destination	DINT	tag	output data from the sequencer array
Control	CONTROL	tag	control structure for the operation; typically use the same CONTROL as the SQI and SQL instructions
Length	DINT	immediate	number of elements in the Array (sequencer table) to output
Position	DINT	immediate	current position in the array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SQR **Square Root**			`dest := SQRT(source);`	The SQR instruction computes the square root of the Source and places the result in the Destination.

Relay Ladder and Structured Text

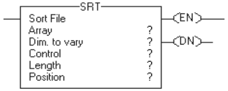
Operand:	Type:	Format:	Description:
Source	SINT DINT INT REAL	immediate tag	find the square root of this value
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
SQR tag	FBD_MATH_ ADVANCED	structure	SQR structure (default parameters):

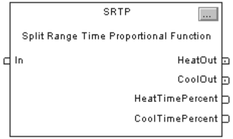
Parameter:	Type:	Description:
Source	REAL	find the square root of this value
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SRT **File Sort**	 SRT Sort File Array ? Dim. to vary ? Control ? Length ? Position ? EN DN	not available	`SRT(Array,Dimtovary, Control);`	The SRT instruction sorts a set of values in one dimension (Dim to vary) of the Array into ascending order.

Operand:	Type:	Format:	Description:
Array	SINT DINT INT REAL	array tag	array to sort; specify the first element of the group of elements to sort **do not** use CONTROL.POS in the subscript
Dimension to vary	DINT	immediate (0, 1, 2)	which dimension to use the order is: array[dim_0,dim_1,dim_2] then array[dim_0,dim_1] then array[dim_0]
Control	CONTROL	tag	control structure for the operation
Length	DINT	immediate	number of elements of the array to sort
Position	DINT	immediate	current element in the array; initial value is typically 0

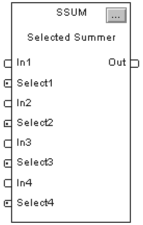
Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 20	• instruction tries to access data outside of the array boundaries • dimension to vary does not exist for the specified array
	Type 4	Code 21	.POS < 0 or .LEN < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SRTP **Split Range** **Proportional**	not available	 SRTP ... Split Range Time Proportional Function In HeatOut CoolOut HeatTimePercent CoolTimePercent	`SRTP(SRTP_tag);`	The SRTP instruction takes the 0-100% output of a PID loop and drives heating and cooling digital output contacts with a periodic pulse. This instruction controls applications such as barrel temperature control on extrusion machines.

Operand:	Type:	Format:	Description:
SRTP tag	SPLIT_RANGE	structure	SRTP structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input asking for heating or cooling
HeatOut	BOOL	heating output pulse
CoolOut	BOOL	cooling output pulse
HeatTimePercent	REAL	calculated percent of the current cycle that the HeatOut will be on
CoolTimePercent	REAL	calculated percent of the current cycle that the CoolOut will be on

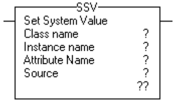
Arithmetic Status Flags:	Major Faults:
set for the HeatTimePercent and CoolTimePercent parameters	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SSUM Selected Summer	not available	 SSUM ... Selected Summer In1 Out Select1 In2 Select2 In3 Select3 In4 Select4	`SSUM(SSUM_tag);`	The SSUM instruction uses boolean inputs to select real inputs to be algebraically summed.

Operand:	Type:	Format:	Description:
SSUM tag	SELECTABLE_ SUMMER	structure	SSUM structure (default parameters):

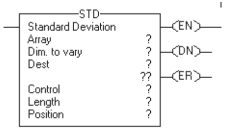
Parameter:	Type:	Description:
In*x*	REAL	input, where *x* = 1-4
Select*x*	BOOL	selector signal for associated input, where *x* = 1-4
Out	REAL	calculated output of the algorithm

Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SSV **Set System Value**		not available	`SSV(ClassName, InstanceName, AttributeName,Source);`	The GSV/SSV instructions get and set controller system data that is stored in objects.

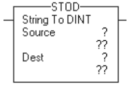
Operand:	Type:	Format:	Description:
Class name	na	name	name of object
Instance name	na	name	name of specific object, when object requires name
Attribute Name	na	name	attribute of object; data type depends on the attribute you select
Source	SINT DINT INT REAL	tag	tag that contains data you want to copy to the attribute

Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 5	invalid object address
	Type 4	Code 6	• specified an object that does not support GSV/SSV • invalid attribute • did not supply enough information for an SSV instruction
	Type 4	Code 7	the GSV destination was not large enough to hold the requested data

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
STD **Standard Deviation**	 STD Standard Deviation Array ? Dim. to vary ? Dest ? ?? Control ? Length ? Position ? EN DN ER	not available	not available	The STD instruction calculates the standard deviation of a set of values in one dimension of the Array and stores the result in the Destination.

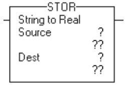
Operand:	Type:	Format:	Description:
Array	SINT DINT INT REAL	array tag	find the standard deviation of the values in this array specify the first element of the group of elements to use in calculating the standard deviation **do not** use CONTROL.POS in the subscript
Dimension to vary	DINT	immediate (0, 1, 2)	which dimension to use the order is: array[dim_0,dim_1,dim_2] then array[dim_0,dim_1] then array[dim_0]
Destination	REAL	tag	result of the operation
Control	CONTROL	tag	control structure for the operation
Length	DINT	immediate	number of elements of the array to use in calculating the standard deviation
Position	DINT	immediate	current element in the array; initial value is typically 0

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 20	dimension to vary does not exist for the specified array
	Type 4	Code 21	.POS < 0 or .LEN < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
STOD **String to DINT**		not available	`STOD(Source,Dest);`	The STOD instruction converts the ASCII representation of an integer to an integer or REAL value.

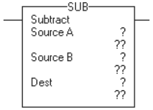
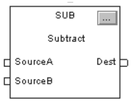
Operand:	Type:	Format:	Description:
Source	string	tag	tag that contains the value in ASCII
Destination	SINT DINT INT REAL	tag	tag to store the integer value; if the Source value is a floating-point number, the instruction converts only the non-fractional part of the number (regardless of the destination data type).

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	Type 4	Code 53	The output number is beyond the limits of the destination data type. Either: • reduce the size of the ASCII value • use a larger data type for the destination

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
STOR **String to REAL**	 STOR String to Real Source ? ?? Dest ? ??	not available	`STOR(Source,Dest);`	The STOR instruction converts the ASCII representation of a floating-point value to a REAL value.

Operand:	Type:	Format:	Description:
Source	string	tag	tag that contains the value in ASCII
Destination	REAL	tag	tag to store the REAL value

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 51	The LEN value of the string tag is greater than the DATA size of the string tag. Check: • that no instruction is writing to the LEN member of the string tag. • in the LEN value, you entered the number of characters that the string contains.
	Type 4	Code 53	The output number is beyond the limits of the destination data type. Either: • reduce the size of the ASCII value • use a larger data type for the destination

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SUB **Subtract**			`dest := sourceA - sourceB;`	The SUB instruction subtracts Source B from Source A and places the result in the Destination.

Relay Ladder and Structured Text

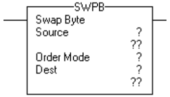
Operand:	Type:		Format:	Description:
Source A	SINT INT	DINT REAL	immediate tag	value from which to subtract Source B
Source B	SINT INT	DINT REAL	immediate tag	value to subtract from Source A
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
SUB tag	FBD_MATH	structure	SUB structure (default parameters):

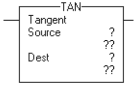
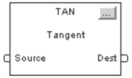
Parameter:	Type:	Description:
SourceA	REAL	value from which to subtract Source B
SourceB	REAL	value to subtract from Source A
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
SWPB **Swap Byte**	 SWPB Swap Byte Source ? ?? Order Mode ? Dest ? ??	not available	`SWPB(Source,OrderMode, Dest);`	The SWPB instruction rearranges the bytes of a value.

Operand:	Type:	Format:	Description:
Source	INT REAL DINT	tag	tag that contains the bytes that you want to rearrange
Order Mode	na	REVERSE WORD HIGH/LOW	how you want to change the order of the bytes
Destination	INT REAL DINT	tag	tag to store the bytes in the new order

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TAN **Tangent**			`dest := TAN(source);`	The TAN instruction takes the tangent of the Source value (in radians) and stores the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:		Format:	Description:
Source	SINT INT	DINT REAL	immediate tag	find the tangent of this value
Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block

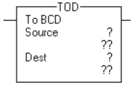
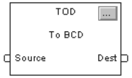
Operand:	Type:	Format:	Description:
TAN tag	FBD_MATH_ ADVANCED	structure	TAN structure (default parameters):

Parameter:	Type:	Description:
Source	REAL	input to the math instruction
Dest	REAL	result of the math instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TND **Temporary End**	—(TND)—	not available	`TND();`	The TND instruction acts as a boundary.

Arithmetic Status Flags:	Major Faults:
not affected	none

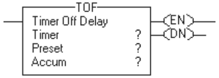
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TOD **Convert to BCD**	 TOD To BCD Source ? ?? Dest ? ??	 TOD ... To BCD Source Dest	not available	The TOD instruction converts a decimal value (0 ≤Source ≤ 99,999,999) to a BCD value and stores the result in the Destination.

Relay Ladder	Operand:	Type:		Format:	Description:
	Source	SINT INT	DINT	immediate tag	value to convert
	Destination	SINT INT	DINT	tag	tag to store the result

Function Block	Operand:	Type:	Format:	Description:
	TOD tag	FBD_CONVERT	structure	TOD structure (default parameters):

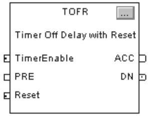
Parameter:	Type:	Description:
Source	DINT	input to the conversion instruction
Dest	DINT	result of the conversion instruction

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 4	Source < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TOF **Timer Off Delay**	 TOF Timer Off Delay Timer ? Preset ? Accum ? EN DN	see TOFR	see TOFR	The TOF instruction is a non-retentive timer that accumulates time when the instruction is enabled (rung-condition-in is false).

Operand:	Type:	Format:	Description:
Timer	TIMER	tag	timer structure
Preset	DINT	immediate	how long to delay (accumulate time)
Accum	DINT	immediate	number of msec the timer has counted; initial value is typically 0

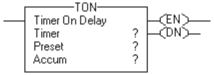
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 34	• .PRE < 0 • .ACC < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TOFR **Timer Off Delay with Reset**	see TOF	 TOFR ... Timer Off Delay with Reset TimerEnable PRE Reset ACC DN	`TOFR(TOFR_tag);`	The TOFR instruction is a non-retentive timer that accumulates time when TimerEnable is cleared.

Operand:	Type:	Format:	Description:
TOFR tag	FBD_TIMER	structure	TOFR structure (default parameters):

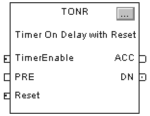
Parameter:	Type:	Description:
TimerEnable	BOOL	if cleared, enables the timer to run and accumulate time
PRE	DINT	timer preset value in 1msec units
Reset	BOOL	request to reset the timer
ACC	BOOL	accumulated time in milliseconds
DN	BOOL	timing done output. Indicates when the ACC ≥ PRE

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TON **Timer On Delay**	 TON Timer On Delay Timer ? Preset ? Accum ? EN DN	see TONR	see TONR	The TON instruction is a non-retentive timer that accumulates time when the instruction is enabled (rung-condition-in is true).

Operand:	Type:	Format:	Description:
Timer	TIMER	tag	timer structure
Preset	DINT	immediate	how long to delay (accumulate time)
Accum	DINT	immediate	number of msec the timer has counted; initial value is typically 0

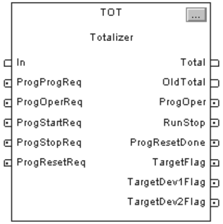
Arithmetic Status Flags:	Major Faults:		
not affected	Type 4	Code 34	• .PRE < 0 • .ACC < 0

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TONR **Timer On Delay with Reset**	see TON	 TONR Timer On Delay with Reset TimerEnable PRE Reset ACC DN	`TONR(TONR_tag);`	The TONR instruction is a non-retentive timer that accumulates time when TimerEnable is set.

Operand:	Type:	Format:	Description:
TONR tag	FBD_TIMER	structure	TONR structure (default parameters):

Parameter:	Type:	Description:
TimerEnable	BOOL	if cleared, enables the timer to run and accumulate time
PRE	DINT	timer preset value in 1msec units
reset	BOOL	request to reset the timer
ACC	BOOL	accumulated time in milliseconds
DN	BOOL	timing done output. Indicates when the ACC ≥ PRE

Arithmetic Status Flags:	Major Faults:
not affected	none

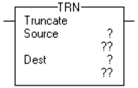
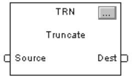
Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TOT Totalizer	not available	 TOT Totalizer In ProgProgReq ProgOperReq ProgStartReq ProgStopReq ProgResetReq Total OldTotal ProgOper RunStop ProgResetDone TargetFlag TargetDev1Flag TargetDev2Flag	`TOT(TOT_tag);`	The TOT instruction provides a time-scaled accumulation of an analog input value.

Operand:	Type:	Format:	Description:
TOT tag	TOTALIZER	structure	TOT structure (default parameters):

Parameter:	Type:	Description:
In	REAL	analog signal input to the instruction
ProgProgReq	BOOL	program program request
ProgOperReq	BOOL	program operator request
ProgStartReq	BOOL	program start request
ProgStopRequest	BOOL	program stop request
ProgResetReq	BOOL	program reset request

continued

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TOT **Totalizer** (continued)		**Parameter:**	**Type:**	**Description:**
		Total	REAL	the totalized value if In
		OldTotal	REAL	the value of the total before a reset occurred
		ProgOper	BOOL	program/operator control indicator
		RunStop	BOOL	the indicator of the operational state of the totalizer
		ProgResetDone	BOOL	the indicator that the TOT instruction has completed a program reset request
		TargetFlag	BOOL	the flag for Total; set when Total ≥ Target.
		TargetDev1Flag	BOOL	the flag for TargetDev1; set when Total ≥ Target - TargetDev1
		TargetDev2Flag	BOOL	the flag for TargetDev2; set when Total ≥ Target - TargetDev2
	Arithmetic Status Flags:	**Major Faults:**		
	set for the Total parameter	none		

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
TRN **Truncate**			`dest := TRUNC(source);`	The TRN instruction removes (truncates) the fractional part of the Source and stores the result in the Destination.

Relay Ladder and Structured Text

Operand:	Type:	Format:	Description:
Source	REAL	immediate tag	value to truncate
Destination	SINT DINT INT REAL	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
TRN tag	FBD_ TRUNCATE	structure	TRN structure (default parameters):

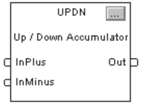
Parameter:	Type:	Description:
Source	REAL	input to the conversion instruction.
Dest	DINT	result of the math instruction.

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

UID
User Interrupt Disable
 	
UIE
User Interrupt Enable
		not available	`UID();`
			
			`UIE();`
				The UID instruction and the UIE instruction work together to prevent a small number of critical rungs from being interrupted by other tasks.

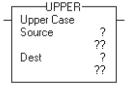
Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
UPDN Up/Down Accumulator	not available	 UPDN ... Up / Down Accumulator InPlus Out InMinus	`UPDN(UPDN_tag);`	The UPDN instruction adds and subtracts two inputs into an accumulated value.

Operand:	Type:	Format:	Description:
UPDN tag	UP_DOWN_ ACCUM	structure	UPDN structure (default parameters):

Parameter:	Type:	Description:
InPlus	REAL	input added to the accumulator
InMinus	REAL	input subtracted from the accumulator
Out	REAL	output of the instruction

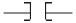
Arithmetic Status Flags:	Major Faults:
set for the Out parameter	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
UPPER **Upper Case**		not available	`UPPER(Source,Dest);`	The UPPER instruction converts the alphabetical characters in a string to upper case characters.

Operand:	Type:	Format:	Description:
Source	string	tag	tag that contains the characters that you want to convert to upper case
Destination	string	tag	tag to store the characters in upper case

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:

XIC
Examine If
Closed
		not available	`IF data_bit THEN`
			`    <statement>;`
			`END_IF;`
				The XIC instruction examines the data bit to see if it is set.

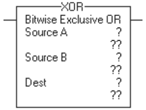
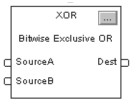
Operand:	Type:	Format:	Description:
data bit	BOOL	tag	bit to be tested

Arithmetic Status Flags:	Major Faults:
not affected	none

Instruction:	Relay Ladder:		Function Block:		Structured Text:	Description:

XIO Examine If Open	—] / [—		not available		`IF NOT data_bit THEN`
					`     <statement>;`
					`END_IF;`
						The XIO instruction examines the data bit to see if it is cleared.

	Operand:	**Type:**	**Format:**	**Description:**		
	data bit	BOOL	tag	bit to be tested		
	Arithmetic Status Flags:		**Major Faults:**			
	not affected		none			

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
XOR Bitwise Exclusive OR	 XOR Bitwise Exclusive OR Source A ? ?? Source B ? ?? Dest ? ??	 XOR ... Bitwise Exclusive OR SourceA Dest SourceB	`dest := sourceA XOR sourceB`	The XOR instruction performs a bitwise XOR operation using the bits in Source A and Source B and places the result in the Destination.

Relay Ladder and Structured Text

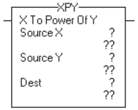
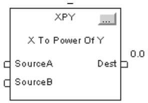
Operand:	Type:	Format:	Description:
Source A	SINT DINT INT	immediate tag	value to XOR with Source B
Source B	SINT DINT INT	immediate tag	value to XOR with Source A
Destination	SINT DINT INT	tag	tag to store the result

Function Block

Operand:	Type:	Format:	Description:
XOR tag	FBD_LOGICAL	structure	XOR structure (default parameters):

Parameter:	Type:	Description:
SourceA	DINT	value to XOR with Source B
SourceB	DINT	value to XOR with Source A
Dest	DINT	result of the instruction

Arithmetic Status Flags:	Major Faults:
affected	none

Instruction:	Relay Ladder:	Function Block:	Structured Text:	Description:
XPY **X to the** **Power of Y**	 XPY X To Power Of Y Source X ? ?? Source Y ? ?? Dest ? ??	 XPY ... X To Power Of Y SourceA SourceB Dest 0.0	`dest := sourceX ** sourceY;`	The XPY instruction takes Source A (X) to the power of Source B (Y) and stores the result in the Destination.

Relay Ladder and Structured Text	Operand:	Type:		Format:	Description:
	Source X	SINT INT	DINT REAL	immediate tag	base value
	Source Y	SINT INT	DINT REAL	immediate tag	exponent
	Destination	SINT INT	DINT REAL	tag	tag to store the result

Function Block	Operand:	Type:	Format:	Description:
	XPY tag	FBD_MATH	structure	LOXPY structure (default parameters):

Parameter:	Type:	Description:	
Source X	REAL	immediate tag	base value
Source Y	REAL	immediate tag	exponent
Dest	REAL	tag	tag to store the result

Arithmetic Status Flags:	Major Faults:		
affected	Type 4	Code 4	Source X is negative and Source Y is not an integer value

Rockwell Automation Support

Rockwell Automation provides technical information on the web to assist you in using our 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 our products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs 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

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-QR107C-EN-P - June 2005
Supersedes Publication 1756-QR107B-EN-P - June 2003

PN 957955-66