Logix 5000 Controllers General Instructions

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, Emulate 5570

Rockwell Automation

Reference Guide

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

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

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

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

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

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

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

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

These labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

The following icon may appear in the text of this document.

Tip: Identifies information that is useful and can help to make a process easier to do or easier to understand.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Summary of changes

This manual includes new and updated information. Use these reference tables to locate changed information.

Global changes

None for this release.

New or enhanced features

Subject	Reason
Access the TimeSynchronize object on page 252	Anomaly resolution; corrected link to the *Deploying Scalable Time Distribution within a Converged Plantwide Ethernet Architecture Design Guide*.
File Search and Compare (FSC) on page 536	Anomaly resolution; in the Ladder diagram table, removed BOOL from the list of data types supported for 5x80 controllers.

Contents

Preface

This manual provides a programmer with details about the available General, Motion, Process, and Drives instruction set for a Logix-based controller.

If you design, program, or troubleshoot safety applications that use GuardLogix controllers, refer to the GuardLogix Safety Application Instruction Set Safety Reference Manual, publication 1756-RM095.

This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system. Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Instruction Locator

Use this locator to find the applicable Logix5000 controllers instruction manual for each instruction.

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	**Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006**	**Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002**	**PlantPAx Process Control Instructions PROCES-RM215**
Absolute Value (ABS)	Alarm (ALM)	Master Driven Coordinated Control (MDCC)	Process Analog HART (PAH)
Add (ADD)	Attach to Equipment Phase (PATT)	Motion Apply Axis Tuning (MAAT)	Process Analog Input (PAI)
Analog Alarm (ALMA)	Attach to Equipment Sequence (SATT)	Motion Apply Hookup Diagnostics (MAHD)	Process Dual Sensor Analog Input (PAID)
Always False (AFI)	Coordinated Control (CC)	Motion Arm Output Cam (MAOC)	Process Multi Sensor Analog Input (PAIM)
Arc Cosine (ACOS)	D Flip-Flop (DFF)	Motion Arm Registration (MAR)	Process Analog Output (PAO)
Arc Sine (ASIN)	Deadtime (DEDT)	Motion Arm Watch (MAW)	Process Boolean Logic (PBL)
Arc Tangent (ATAN)	Derivative (DERV)	Motion Axis Fault Reset (MAFR)	Process Command Source (PCMDSRC)

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
ASCII Chars in Buffer (ACB)	Detach from Equipment Phase (PDET)	Motion Axis Gear (MAG)	Process Discrete 2-, 3-, or 4-State Device (PD4SD)
ASCII Clear Buffer (ACL)	Detach from Equipment Sequence (SDET)	Motion Axis Home (MAH)	Process Deadband Controller (PDBC)
ASCII Handshake Lines (AHL)	Discrete 3-State Device (D3SD)	Motion Axis Jog (MAJ)	Process Discrete Input (PDI)
ASCII Read (ARD)	Discrete 2-State Device (D2SD)	Motion Axis Move (MAM)	Process Discrete Output (PDO)
ASCII Read Line (ARL)	Enhanced PID (PIDE)	Motion Axis Position Cam (MAPC)	Process Dosing (PDOSE)
ASCII Test for Buffer Line (ABL)	Enhanced Select (ESEL)	Motion Axis Stop (MAS)	Process Analog Fanout (PFO)
ASCII Write (AWT)	Equipment Phase Clear Failure (PCLF)	Motion Axis Time Cam (MATC)	Process High or Low Selector (PHLS)
ASCII Write Append (AWA)	Equipment Phase Command (PCMD)	Motion Axis Shutdown (MASD)	Process Interlocks (PINTLK)
Bit Field Distribute (BTD)	Equipment Phase External Request (PXRQ)	Motion Axis Shutdown Reset (MASR)	Process Lead Lag Standby Motor Group (PLLS)
Bit Field Distribute with Target (BTDT)	Equipment Phase Failure (PFL)	Motion Calculate Cam Profile (MCCP)	Process Motor (PMTR)
Bit Shift Left (BSL)	Equipment Phase New Parameters (PRNP)	Motion Coordinated Path Move (MCPM)	Process Permissives (PPERM)
Bit Shift Right (BSR)	Equipment Phase Override Command (POVR)	Motion Calculate Slave Values (MCSV)	Process Proportional + Integral + Derivative (PPID)
Bitwise And (AND)	Equipment Phase Paused (PPD)	Motion Coordinated Transform with Orientation (MCTO)	Process Pressure/Temperature Compensated Flow (PPTC)
Bitwise (NOT)	Equipment Sequence Assign Sequence Identifier (SASI)	Motion Calculate Transform Position (MCTP)	Process Restart Inhibit (PRI)
Bitwise (OR)	Equipment Sequence Clear Failure (SCLF)	Motion Calculate Transform Position with Orientation (MCTPO)	Process Run Time and Start Counter (PRT)
Boolean AND (BAND)	Equipment Sequence command (SCMD)	Motion Change Dynamics (MCD)	Process Tank Strapping Table (PTST)
Boolean Exclusive OR (BXOR)	Equipment Sequence Override (SOVR)	Motion Coordinated Change Dynamics (MCCD)	Process Valve (PVLV)
Boolean NOT (BNOT)	Function Generator (FGEN)	Motion Coordinated Circular Move (MCCM)	Process Valve Statistics (PVLVS)
Boolean OR (BOR)	High Pass Filter (HPF)	Motion Coordinated Linear Move (MCLM)	
Break (BRK)	High/Low Limit (HLL)	Motion Coordinated Shutdown (MCSD)	
Breakpoints (BPT)	HMI Button Control (HMIBC)	Motion Coordinated Shutdown Reset (MCSR)	
Clear (CLR)	Integrator (INTG)	Motion Coordinated Stop (MCS)	
Compare (CMP)	Internal Model Control (IMC)	Motion Coordinated Transform (MCT)	
Convert to BCD (TO_BCD)	JK Flip-Flop (JKFF)	Motion Direct Drive Off (MDF)	
Convert to Integer (BCD_TO)	Lead-Lag (LDLG)	Motion Direct Drive On (MDO)	
Copy File (COP), Synchronous Copy File (CPS)	Low Pass Filter (LPF)	Motion Direct Start (MDS)	
Cosine (COS)	Maximum Capture (MAXC)	Motion Disarm Output Cam (MDOC)	
Compute (CPT)	Minimum Capture (MINC)	Motion Disarm Registration (MDR)	
Count down (CTD)	Modular Multivariable Control (MMC)	Motion Disarm Watch (MDW)	
Count up (CTU)	Moving Average (MAVE)	Motion Group Shutdown (MGSD)	
Count up/down CTUD	Moving Standard Deviation (MSTD)	Motion Group Shutdown Reset (MGSR)	
Data Transition (DTR)	Multiplexer (MUX)	Motion Group Stop (MGS)	
Degrees (DEG)	Notch Filter (NTCH)	Motion Group Strobe Position (MGSP)	
Diagnostic Detect (DDT)	Phase State Complete (PSC)	Motion Redefine Position (MRP)	
Digital Alarm (ALMD)	Position Proportional (POSP)	Motion Run Axis Tuning (MRAT)	
DINT To String (DTOS)	Proportional + Integral (PI)	Motion Run Hookup Diagnostics (MRHD)	
Divide (DIV)	Pulse Multiplier (PMUL)	Motion Servo Off (MSF)	
End of Transition (EOT)	Ramp/Soak (RMPS)	Motion Servo On (MSO)	
Equal to (EQ)	Rate Limiter (RLIM)		
File Arithmetic (FAL)	Reset Dominant (RESD)		
File Bit Comparison (FBC)	Scale (SCL)		
FIFO Load (FFL)	S-Curve (SCRV)		
FIFO Unload (FFU)	Second-Order Controller (SOC)		
File Average (AVE)	Second-Order Lead Lag (LDL2)		
File Standard Deviation (STD)	Select (SEL)		
File Fill (FLL)	Selected Negate (SNEG)		
File Sort (SRT)	Selected Summer (SSUM)		
Find String (FIND)	Set Dominant (SETD)		
For (FOR)	Split Range Time Proportional (SRTP)		
File Search and Compare (FSC)	Totalizer (TOT)		

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Get System Value (GSV) and Set System Value (SST)	Up/Down Accumulator (UPDN)		
Greater Than or Equal to (GE)			
Greater than (GT)			
Insert String (INSERT)			
Immediate Output (IOT)			
Is Infinity (IsINF)			
Is Not a Number (IsNAN)			
Jump to Label (JMP) and Label (LBL)			
Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)			
Jump to External Routine (JXR)			
Less Than (LT)			
Less Than or Equal to (LE)			
LIFO Load (LFL)			
LIFO Unload (LFU)			
License Validation (LV)			
Limit (LIMIT)			
Log Base (LOG)			
Lower to Case (LOWER)			
Masked Move (MVM)			
Masked Move with Target (MVMT)			
Master Control Reset (MCR)			
Masked Equal to (MEQ)			
Message (MSG)			
Middle String (MID)			
Modulo (MOD)			
Move (MOVE)			
Multiply (MUL)			
Natural Log (LN)			
Negate (NEG)			
Not Equal to (NE)			
No Operation (NOP)			
One Shot (ONS)			
One Shot Falling (OSF)			
One Shot Falling with Input (OSFI)			
One Shot Rising (OSR)			
One Shot Rising with Input (OSRI)			
Output Energize (OTE)			
Output Latch (OTL)			
Output Unlatch (OTU)			
Proportional Integral Derivative (PID)			
Radian (RAD)			
Real to String (RTOS)			
Reset (RES)			
Reset SFC (SFR)			
Return (RET)			
Retentive Timer On (RTO)			
Retentive Timer On with Reset (RTOR)			
Pause SFC (SFP)			
Size In Elements (SIZE)			
Sequencer Input (SQI)			
Sequencer Load (SQL)			
Sequencer Output (SQO)			

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Sine (SIN)			
Square Roost (SQRT)			
String Concatenate (CONCAT)			
String Delete (DELETE)			
String to DINT (STOD)			
String to REAL (STOR)			
Swap Byte (SWPB)			
Subtract (SUB)			
Tangent (TAN)			
Timer Off Delay (TOF)			
Timer Off Delay with Reset (TOFR)			
Timer On Delay (TON)			
Timer On Delay with Reset (TONR)			
Temporary End (TND)			
Tracepoints (TPT)			
Trigger Event Task (EVENT)			
Truncate (TRUNC)			
Unknown Instruction (UNK)			
Upper Case (UPPER)			
User Interrupt Disable (UID)/User Interrupt Enable (UIE)			
X to the Power of Y (EXPT)			
Examine if Closed (XIC)			
Examine If Open (XIO)			
Bitwise Exclusive (XOR)			

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication, 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Rockwell Automation product certifications	Provides declarations of conformity, certificates, and other certification details.

View or download publications at https://www.rockwellautomation.com/en-us/support/documentation/literature-library.html. To order paper copies of technical documentation, contact a local Rockwell Automation distributor or sales representative.

Legal notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

Software and Cloud Services Agreement

Review the Rockwell Automation Software and Cloud Services Agreement here.

Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses.

You can view a full list of all open source software used in this product and their corresponding licenses at this URL:

Studio 5000 Logix Designer Open Source Attribution List

You may obtain Corresponding Source code for open source packages included in this product from their respective project web site(s). Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page. Please include "Open Source" as part of the request text.

Alarm Instructions

Use the alarm instructions to monitor and control alarm conditions.

The Logix-based alarm instructions instructions integrate alarming between the RSView® SE applications and Logix 5000 controllers.

Available Instructions

Ladder Diagram

ALMD on page 50	ALMA on page 22	ASO on page 19

Function Block

ALMD on page 50

Structured Text

ALMD on page 50	ALMA on page 22	ASO on page 19

If:	Use the:
Providing alarming for any discrete Boolean value for a ladder diagram, function block, or structured text	Digital Alarm (ALMD) instruction
Providing level and rate-of-change alarming for any analog signal for ladder diagram, function block, diagram and structured text	Analog Alarm (ALMA) instruction
Issuing a specified operation to all alarm conditions of the specified alarm set	Alarm Set Operation (ASO) instruction

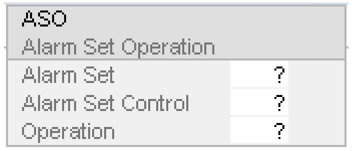
Alarm Set Operation (ASO)

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Alarm Set Operation (ASO) instruction issues a specified operation to all alarm conditions of the specified alarm set. The Alarm Set Operation instruction is used to initiate asynchronous execution of an alarm operation for all alarm conditions of the specified alarm set. The instruction iterates through alarm conditions of the specified alarm set and sets an internal flag requesting the operation execution for each of the conditions. The internal flags have the same purpose and priority as the existing user accessible Progxxx bits and will be processed for all the alarm conditions of the specified alarm set during the next periodic evaluation of each particular alarm condition from the set.

Available Languages

Ladder Diagram



Function Block Diagram

This instruction is not available in Function Block Diagram.

Structured Text

ASO (Alarm Set, Alarm Set Control, Operation)Operands

IMPORTANT: Unexpected operation may occur if:

- The same tag (ALARM_SET_CONTROL) is used as a parameter for more than one instruction invocation.
- The .LastState structure member is modified by a user application program.

WARNING: The Alarm Set Control structure contains internal state information. If any of the configuration operands are changed while in run mode, accept the pending edits and cycle the controller mode from Program to Run for the changes to take effect.

The following table provides operands used for configuring the instruction.

Operand	Data Type	Format	Description
Alarm Set	ALARM_SET	AlarmSet	The ALARM_SET structure represents alarm conditions that are operated on by this instruction.
Alarm Set Control	ALARM_SET_CONTROL	tag	This data type contains three BOOL flags: • EnableIn • EnableOut • LastState The instruction reacts to the edge (transition of .EnableIn from false to true) instead of the level. EnableOut is always set to .EnableIn. The request to perform the instruction operation have the same priority as ProgXXX flags.

Operand	Data Type	Format	Description
Operation		immediate	This operand can be selected from the list or entered as an integer value: 0 - Acknowledge 1 - Reset 2 - Enable 3 - Disable 4- Unshelve 5 - Suppress 6 - Unsuppress 7 - ResetAlarmCount

Affects Math Status FlagsNoMajor/Minor FaultsNone specific to this instruction. See Index through arrays on page 863 for array-indexing faults.Execution

Condition/State	Action Taken
Prescan	The instruction clears all ALARM_SET structure members.
Rung-condition -in is false	The instruction clears .EnableOut and .LastState structure members.
Rung-condition-in is true	If .LastState is false then the instruction initiates the operation and sets .LastState structure member to true. The .EnableOut structure member is always set to true.
Postscan	The instruction clears all ALARM_SET structure members.

OperationThe Alarm Set Operation instruction initiates asynchronous execution of one of the following alarm operations on the specified alarm set:

- Acknowledge
- Reset
- Enable
- Disable
- Unshelve
- Suppress
- Unsuppress
- ResetAlarmCount

The instruction iterates through all alarm conditions which are included in the specified alarm set or in the nested alarm sets to set an internal flag representing the request to perform the required operation on a particular alarm condition. The operation is initiated for all alarm conditions which are iterated by the instruction with the following exceptions:

- Alarm Conditions which are configured not to support alarm operations
- Alarm Conditions which are configured as not used

When an alarm operation is initiated for a particular alarm condition by the instruction, the operation is performed during the next alarm periodic evaluation of the alarm condition.When the instruction is called multiple times for the same Alarm Set to initiate contradictory alarm operations, the last requested operation is always applied to all alarm conditions in the Alarm Set. The alarm operations initiated for the Alarm Set may be applied to the conditions before the last requested operation is performed.When an Alarm Condition is periodically evaluated, the requests to perform particular alarm operations have the same priority as the requests to perform alarm operations initiated via user accessible Progxxx flags. It means that if a request to perform an alarm operation is generated by the instruction, then it is handled as if the corresponding Progxxx flag is set and the same rules used to resolve conflicting requests specified for ProgXXX flags are used to resolve conflicts between the instruction requests and requests made via Progxxx flags.The Alarm Set Operation instruction initiates the required alarm operation only when it detects the transition of .EnableIn value from false to true. In order to detect the transition, .LastState structure member is used to store .EnableIn value from the previous instruction execution. See the Execution section above.

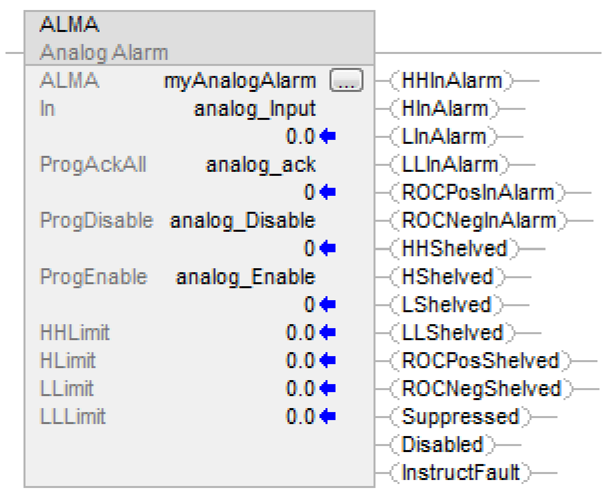
Tip: If the Alarm set provided as the instruction parameter contains an excessive number of alarm conditions, then the execution time of the ASO instruction can increase significantly.

Analog Alarm (ALMA)

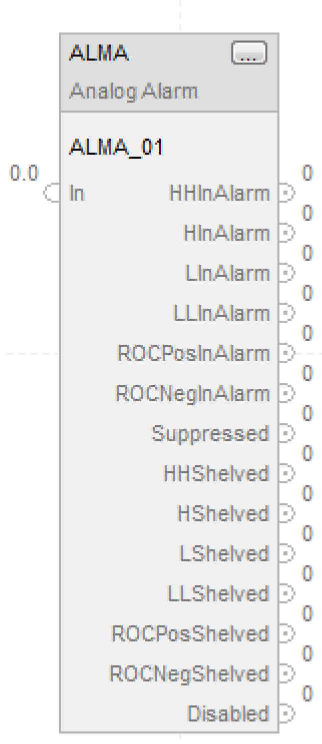
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Analog Alarm (ALMA) instruction provides level and rate-of-change alarming for any analog signal.

Ladder Diagram



Function Block



Structured Text

ALMA (ALMA,In,ProgAckAll,ProgDisable,ProgEnable)

Operands

Ladder Diagram

Operand	Type	Format	Description
ALMA	ALARM_ANALOG	Structure	ALMA structure
In	REAL DINT INT SINT	Tag Immediate	The alarm input value, which is compared with alarm limits to detect the alarm condition.
ProgAckAll	BOOL	Tag Immediate	On transition from False to True, acknowledges all alarm conditions that require acknowledgement.
ProgDisable	BOOL	Tag Immediate	When True, disables alarm (does not override Enable Commands).

ProgEnable	BOOL	Tag Immediate	When True, enables alarm (takes precedence over Disable commands).

Function Block

Operand	Type	Format	Description
ALMA tag	ALARM_ANALOG	structure	ALMA structure

Structured Text

Operand	Type	Format	Description
ALMA	ALARM_ANALOG	Structure	ALMA structure
In	REAL DINT INT SINT	Tag Immediate	The alarm input value, which is compared with alarm limits to detect the alarm condition.
ProgAckAll	BOOL	Tag Immediate	On transition from False to True, acknowledges all alarm conditions that require acknowledgement.
ProgDisable	BOOL	Tag Immediate	When True, disables alarm (does not override Enable Commands).
ProgEnable	BOOL	Tag Immediate	When True, enables alarm (takes precedence over Disable commands).

See Structured Text Syntax on page 879 for more information on the syntax of expressions within the structured text.

ALMA Structure

Input Parameters

Input Parameter	**Data Type**	**Description**
EnableIn	BOOL	Ladder Diagram: Corresponds to the rung state. If false, the instruction does not execute and outputs are not updated. Structured Text: If false, the instruction does not execute and outputs are not updated. Default is set.

Input Parameter	Data Type	Description
		Function Block: If false, the instruction does not execute and outputs are not updated. Default is set.
In	REAL	The alarm input value, which is compared with alarm limits to detect the alarm condition. Default = 0.0. Ladder Diagram: Copied from instruction operand. Structured Text: Copied from instruction operand.
InFault	BOOL	Bad health indicator for the input. The user application may set InFault to indicate the input signal has an error. When set, the instruction sets InFaulted (Status.1). When cleared to false, the instruction clears InFaulted to false (Status.1). In either case, the instruction continues to evaluate In for alarm conditions. Default is false (good health).
HHEnabled	BOOL	High High alarm condition detection. Set to true to enable detection of the High High alarm condition. Clear to false to make detection unavailable for the High High alarm condition. Default is set.
HEnabled	BOOL	High alarm condition detection. Set to true to enable detection of the High alarm condition. Clear to false to make detection unavailable for the High alarm condition. Default is set.
LEnabled	BOOL	Low alarm condition detection. Set to true to enable detection of the Low alarm condition. Clear to false to make detection unavailable for the Low alarm condition. Default is set.
LLEnabled	BOOL	Low Low alarm condition detection. Set to true to enable detection of the Low Low alarm condition. Clear to false to make

Input Parameter	Data Type	Description
		detection unavailable for the Low Low alarm condition. Default is set.
AckRequired	BOOL	Specifies whether alarm acknowledgment is required. When set to true, acknowledgment is required. When cleared to false, acknowledgment is not required and HHAcked, HAcked, LAcked, LLAcked, ROCPosAcked, and ROCNegAcked are always set to true Default is true.
ProgAckAll	BOOL	Set to true by the user program to acknowledge all alarm conditions. Takes effect only if any alarm condition is unacknowledged. Requires a false-to-true transition. Default is false. Ladder Diagram: Copied from the instruction operand. Structured Text: Copied from the instruction operand.
OperAckAll	BOOL	Set to true by the operator interface to acknowledge all alarm conditions. Takes effect only if any alarm condition is unacknowledged. The alarm instruction clears this parameter to false. Default is false.
HHProgAck	BOOL	High High program acknowledge. Set to true by the user program to acknowledge a High High condition. Takes effect only if the alarm condition is unacknowledged. Requires a false -to-true transition. Default is false.
HHOperAck	BOOL	High High operator acknowledge. Set to true by the operator interface to acknowledge a High High condition. Takes effect only if the alarm condition is unacknowledged. The alarm instruction clears this parameter to false. Default is false.
HProgAck	BOOL	High program acknowledge. Set to true by the user program to acknowledge a High condition. Takes effect only if the alarm

Input Parameter	Data Type	Description
		condition is unacknowledged. Requires a false-to-true transition. Default is false.
HOperAck	BOOL	High operator acknowledge. Set to true by the operator interface to acknowledge a High condition. Takes effect only if the alarm condition is unacknowledged. The alarm instruction clears this parameter to false. Default is false.
LProgAck	BOOL	Low program acknowledge. Set to true by the user program to acknowledge a Low condition. Takes effect only if the alarm condition is unacknowledged. Requires a false-to-true transition. Default is false.
LOperAck	BOOL	Low operator acknowledge. Set to true by the operator interface to acknowledge a Low condition. Takes effect only if the alarm condition is unacknowledged. The alarm instruction clears this parameter to false. Default is false.
LLProgAck	BOOL	Low Low program acknowledge. Set to true by the user program to acknowledge a Low Low condition. Takes effect only if the alarm condition is unacknowledged. Requires a false-to-true transition. Default is false.
LLOperAck	BOOL	Low Low operator acknowledge. Set to true by the operator interface to acknowledge a Low Low condition. Takes effect only if the alarm condition is unacknowledged. The alarm instruction clears this parameter false. Default is false.
ROCPosProgAck	BOOL	Positive rate of change program acknowledge. Set to true by the user program to acknowledge a positive rate-of-change condition. Requires a false-to-true transition while the alarm condition is unacknowledged. Default is false.

Input Parameter	Data Type	Description
ROCPosOperAck	BOOL	Positive rate of change operator acknowledge. Set to true by the operator interface to acknowledge a positive rate-of-change condition. Requires a false-to-true transition while the alarm condition is unacknowledged. The alarm instruction sets this parameter to false. Default is false.
ROCNegProgAck	BOOL	Negative rate of change program acknowledge. Set to true by the user program to acknowledge a negative rate-of-change condition. Requires a false-to-true transition while the alarm condition is unacknowledged. Default is false.
ROCNegOperAck	BOOL	Negative rate of change operator acknowledge. Set to true by the operator interface to acknowledge a negative rate-of-change condition. Requires a false-to-true transition while the alarm condition is unacknowledged. The alarm instruction clears this parameter to false. Default is false.
ProgSuppress	BOOL	Set to true by the user program to suppress the alarm. Default is cleared.
OperSuppress	BOOL	Set to true by the operator interface to suppress the alarm. The alarm instruction clears this parameter to false. Default is false.
ProgUnsuppress	BOOL	Set to true by the user program to unsuppress the alarm. Takes precedence over Suppress commands. Default is false.
OperUnsuppress	BOOL	Set to true by the operator interface to unsuppress the alarm. Takes precedence over Suppress commands. The alarm instruction sets this parameter to false. Default is false.
HHOperShelve	BOOL	High-high operator shelve. Set to true by the operator interface to shelve or reshelve a high-high condition. Requires

Input Parameter	Data Type	Description
		a false-to-true transition. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands. Shelving an alarm postpones alarm processing. It is like suppressing an alarm, except that shelving is time limited. If an alarm is acknowledged while it is shelved, it remains acknowledged even if it becomes active again. It becomes unacknowledged when the shelve duration ends.
HOperShelve	BOOL	High operator shelve. Set to true by the operator interface to shelve or reshelve a high condition. Requires a transition from false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands.
LOperShelve	BOOL	Low operator shelve. Set to true by the operator interface to shelve or reshelve a low condition. Requires a transition false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands.
LLOperShelve	BOOL	Low-low operator shelve. Set to true by the operator interface to shelve or reshelve a low-low condition. Requires a transition from false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands.
ROCPosOperShelve	BOOL	Positive rate-of-change operator shelve. Set to true by the operator

Input Parameter	Data Type	Description
		interface to shelve or reshelve a positive rate-of-change condition. Requires a transition from false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands.
ROCNegOperShelve	BOOL	Negative rate-of-change operator shelve. Set to true by the operator interface to shelve or reshelve a negative rate-of-change condition. Requires a transition from false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands.
ProgUnshelveAll	BOOL	Set to true by the user program to unshelve all conditions on this alarm. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
HHOperUnshelve	BOOL	High-high operator unshelve. Set to true by the operator interface to unshelve a high-high condition. The alarm instruction clears this parameter to false. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
HOperUnshelve	BOOL	High operator unshelve. Set to true by the operator interface to unshelve a high condition. The alarm instruction clears this parameter to false. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
LOperUnshelve	BOOL	Low operator unshelve. Set to true by the operator interface to unshelve a low condition. The alarm instruction clears

Input Parameter	Data Type	Description
		this parameter to false. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
LLOperUnshelve	BOOL	Low-low operator unshelve. Set to true by the operator interface to unshelve a low-low condition. The alarm instruction clears this parameter to false. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
ROCPosOperUnshelve	BOOL	Positive rate-of-change operator unshelve. Set to true by the operator interface to unshelve a positive rate-of-change condition. The alarm instruction clears this parameter to false. If both shelve and unshelve are set, unshelve commands take precedence over shelve commands. Default is false.
ROCNegOperUnshelve	BOOL	Negative rate-of-change operator unshelve. Set to true by the operator interface to unshelve a negative rate-of-change condition. The alarm instruction clears this parameter to false. If both shelve and unshelve are true, unshelve commands take precedence over shelve commands. Default is false.
ProgDisable	BOOL	Copied from the instruction operand.
OperDisable	BOOL	Set to true by the operator interface to disable the alarm. The alarm instruction clears this parameter to false. Default is false.
ProgEnable	BOOL	Copied from the instruction operand.
OperEnable	BOOL	Set to true by the operator interface to enable the alarm. Takes precedence over Disable command. The alarm instruction clears this parameter false. Default is false.
AlarmCountReset	BOOL	Set to true by the operator interface to reset the alarm counts for all conditions.

Input Parameter	Data Type	Description
		The alarm instruction clears this parameter to false. Default is false.
HHMinDurationEnable	BOOL	High-high minimum duration enable. Set to true to enable minimum duration timer when detecting the high-high condition. Default is true.
HMinDurationEnable	BOOL	High minimum duration enable. Set to true to enable minimum duration timer when detecting the high condition. Default is true.
LMinDurationEnable	BOOL	Low minimum duration enable. Set to true to enable minimum duration timer when detecting the low condition. Default is true.
LLMinDurationEnable	BOOL	Low-low minimum duration enable. Set to true to enable minimum duration timer when detecting the low-low condition. Default is true.
HHLimit	REAL	High High alarm limit. Valid = HLimit < HHLimit < maximum positive float. Default = 0.0.
HHSeverity	DINT	Severity of the High High alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
HLimit	REAL	High alarm limit. Valid = LLimit < HLimit < HHLimit. Default = 0.0.
HSeverity	DINT	Severity of the High alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.

Input Parameter	Data Type	Description
LLimit	REAL	Low alarm limit. Valid = LLLimit < LLimit < HLimit. Default = 0.0.
LSeverity	DINT	Severity of the Low alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
LLLimit	REAL	Low Low alarm limit. Valid = maximum negative float < LLLimit < LLimit. Default = 0.0.
LLSeverity	DINT	Severity of the Low Low alarm condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
MinDurationPRE	DINT	Minimum duration preset (milliseconds) for an alarm level condition to remain true before the condition is marked as InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected; so no data is lost while waiting to meet the minimum duration. Does not apply to rate-of-change conditions or to conditions for which minimum duration detection is disabled. MinDurationPRE only applies to the first excursion from normal in either direction. For example, once the High condition times out, the High High condition becomes active immediately, while a Low condition waits for the timeout period. Valid = 0...2147483647. Default = 0.

Input Parameter	Data Type	Description
ShelveDuration	DINT	Time duration (in minutes) for which a shelved alarm will be shelved. Minimum time is one minute. Maximum time is defined by MaxShelveDuration.
MaxShelveDuration	DINT	Maximum time duration (in minutes) for which an alarm can be shelved.
Deadband	REAL	Deadband for detecting that High High, High, Low, and Low Low alarm levels have returned to normal. A non-zero Deadband can reduce alarm condition chattering if the In value is continually changing but remaining near the level condition threshold. The Deadband value does not affect the transition to the InAlarm (active) state. Once a level condition is active, but before the condition returns to the inactive (normal) state, the In value must either: drop below the threshold minus the deadband (for High and High High conditions). OR rise above the threshold plus the deadband (for Low and Low Low conditions). The Deadband is not used to condition the Minimum Duration time measurement. Valid = 0 = Deadband < Span from first enabled Low alarm to the first enabled High alarm. Default = 0.0.
ROCPosLimit	REAL	Limit for an increasing rate-of-change in units per second. Detection is enabled for any value > 0.0 if ROCPeriod is also > 0.0. Valid = 0.0...maximum possible float. Default = 0.0.
ROCPosSeverity	DINT	Severity of the increasing rate-of-change condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber.

Input Parameter	Data Type	Description
		Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
ROCNegLimit	REAL	Limit for a decreasing rate-of-change in units per second. Detection is enabled for any value > 0.0 if ROCPeriod is also > 0.0. Valid = 0.0...maximum possible float. Default = 0.0.
ROCNegSeverity	DINT	Severity of the decreasing rate-of-change condition. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
ROCPeriod	REAL	Time period in seconds for calculation (sampling interval) of the rate of change value. Each time the sampling interval expires, a new sample of In is stored, and ROC is re-calculated. Instead of an enable bit like other conditions in the analog alarm, the rate-of-change detection is enabled by putting any non-zero value in the ROCPeriod. Valid = 0.0...32767.0 Default = 0.0.

Output Parameters

These output parameters are common to ladder logic.

Output Parameter	Data Type	Description
AnyInAlarmUnack	BOOL	Combined alarm active and acknowledged status. Set to true when any alarm condition is detected and unacknowledged. Cleared to false when all alarm conditions are inactive, acknowledged, or both.
HHInAlarm	BOOL	High High alarm condition status. Set to true when a High High condition is Active. Cleared to false when no High High condition exists.

Output Parameter	Data Type	Description
HInAlarm	BOOL	High alarm condition status. Set to true when a High condition is Active. Cleared to false when no High condition exists.
LInAlarm	BOOL	Low alarm condition status. Set to true when a Low condition is Active. Cleared to false when no Low condition exists.
LLInAlarm	BOOL	Low Low alarm condition status. Set to true when a Low Low condition is Active. Cleared to false when no Low Low condition exists.
ROCPosInAlarm	BOOL	Positive rate-of-change alarm condition status. Set to true when a positive rate-of-change condition exists. Cleared to false when no positive rate-of-change condition exists.
ROCNegInAlarm	BOOL	Negative rate-of-change alarm condition status. Set to true when a negative rate-of-change condition exists. Cleared to False when no negative rate-of-change condition exists.
ROC	REAL	Calculated rate-of-change of the In value. This value is updated when the instruction is scanned following each elapsed ROCPeriod. The ROC value is used to evaluate the ROCPosInAlarm and ROCNegInAlarm conditions. ROC = (current sample of In – previous sample of In) / ROCPeriod
HHAcked	BOOL	High High condition acknowledged status. Set to true when a High High condition is acknowledged. Always set to true when AckRequired is cleared to false. Cleared to false when a High High condition is not acknowledged.
HAcked	BOOL	High condition acknowledged status. Set to true when a High condition is acknowledged. Always set to true when AckRequired is cleared to false. Cleared to false when a High condition is not acknowledged.
LAcked	BOOL	Low condition acknowledged status. Set to true when a Low condition is acknowledged. Always set to true when

Output Parameter	Data Type	Description
		AckRequired is cleared to false. Cleared to false when a Low condition is not acknowledged.
LLAcked	BOOL	Low Low condition acknowledged status. Set to true when a Low Low condition is acknowledged. Always true when AckRequired is cleared to false. Cleared to false when a Low Low condition is not acknowledged.
ROCPosAcked	BOOL	Positive rate-of-change condition acknowledged status. Set to true when a positive rate-of-change condition is acknowledged. Always true when AckRequired is cleared to false. Cleared to false when a positive rate-of-change condition is not acknowledged.
ROCNegAcked	BOOL	Negative rate-of-change condition acknowledged status. Set to true when a negative rate-of-change condition is acknowledged. Always set to true when AckRequired is cleared to false. Cleared to false when a negative rate-of-change condition is not acknowledged.
HHInAlarmUnack	BOOL	Combined High High condition active and unacknowledged status. Set to true when the High High condition is active (HHInAlarm is true) and unacknowledged. Cleared to false when the High High condition is inactive, acknowledged, or both.
HInAlarmUnack	BOOL	Combined High condition active and unacknowledged status. Set to true when the High condition is active (HInAlarm is true) and unacknowledged. Cleared to false when the High condition is inactive, acknowledged, or both.
LInAlarmUnack	BOOL	Combined Low condition active and unacknowledged status. Set to true when the Low condition is active (LInAlarm is true) and unacknowledged. Cleared to false when the Low condition is inactive, acknowledged, or both.

Output Parameter	Data Type	Description
LLInAlarmUnack	BOOL	Combined Low Low condition active and unacknowledged status. Set to true when the Low Low condition is active (LLInAlarm is true) and unacknowledged. Cleared to false when the Low Low condition is inactive, acknowledged, or both.
ROCPosInAlarmUnack	BOOL	Combined positive rate-of-change condition active and unacknowledged status. Set to true when the positive rate-of-change condition is active (ROCPosInAlarm is true) and unacknowledged. Cleared to false when the positive rate-of-change condition is inactive, acknowledged, or both.
ROCNegInAlarmUnack	BOOL	Combined negative rate-of-change condition active and unacknowledged status. Set to true when the negative rate-of-change condition is active (ROCNegInAlarm is true) and unacknowledged. Cleared to false when the negative rate-of-change condition is inactive, acknowledged, or both.
Suppressed	BOOL	Suppressed status of the alarm. Set to true when the alarm is suppressed. Cleared to false when the alarm is not suppressed.
HHShelved	BOOL	High-high condition shelved status. Set to true when a high-high condition is shelved. Cleared to false when high-high condition is unshelved.
HShelved	BOOL	High condition shelved status. Set to true when a high condition is shelved. Cleared to false when high condition is unshelved.
LShelved	BOOL	Low condition shelved status. Set to true when a low condition is shelved. Cleared to false when low condition is unshelved.
LLShelved	BOOL	Low-low condition shelved status. Set to true when a low-low condition is shelved. Cleared to false when low-low condition is unshelved.
ROCPosShelved	BOOL	Positive rate-of-change condition shelved status. Set to true when a

Output Parameter	Data Type	Description
		positive rate-of-change condition is shelved. Cleared to false when positive rate-of-change condition is unshelved.
ROCNegShelved	BOOL	Negative rate-of-change condition shelved status. Set to true when a negative rate-of-change condition is shelved. Cleared to false when negative rate-of-change condition is unshelved.
Disabled	BOOL	Disabled status of the alarm. Set to true when the alarm is unavailable (disabled). Cleared to false when the alarm is enabled.
Commissioned	BOOL	The commissioned bit is not used.
MinDurationACC	DINT	Not Used. Value is always 0.
HHInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the High High condition limit for the most recent transition to the active state.
HHAlarmCount	DINT	The number of times the High High condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
HInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the High condition limit for the most recent transition to the active state.
HAlarmCount	DINT	The number of times the High condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
LInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the Low condition limit for the most recent transition to the active state.
LAlarmCount	DINT	The number of times the Low condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
LLInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the

Output Parameter	Data Type	Description
		Low Low condition limit for the most recent transition to the active state.
LLAlarmCount	DINT	The number of times the Low Low condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
ROCPosInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the positive-rate-of-change condition limit for the most recent transition to the active state.
ROCPosInAlarmCount	DINT	The number of times the positive rate-of-change condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
ROCNegInAlarmTime	LINT	Timestamp when the ALMA instruction detected that the In value exceeded the negative-rate-of-change condition limit for the most recent transition to the active state.
ROCNegAlarmCount	DINT	The number of times the negative rate-of-change condition has been activated. If the maximum value is reached, the counter leaves the value at the maximum count value.
AckTime	LINT	Timestamp of most recent condition acknowledgment. If the alarm does not require acknowledgment, this timestamp is equal to most recent condition alarm time.
RetToNormalTime	LINT	Timestamp of alarm returning to a normal state.
AlarmCountResetTime	LINT	Timestamp indicating when the alarm count was reset.
ShelveTime	LINT	Timestamp indicates when an alarm condition was shelved the last time. Set by controller when alarm condition is shelved. Alarm conditions can be shelved and unshelved many times. Each time

<table>
<tr><th>Output Parameter</th><th>Data Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>alarm condition is shelved the timestamp is set to current time.</td></tr>
<tr><td>UnshelveTime</td><td>LINT</td><td>Timestamp indicating when all alarm conditions are going to be unshelved. Value is set only when no alarm condition is shelved yet. The timestamp is determined as sum of the ShelveDuration time period and current time. If an alarm condition is unshelved programmatically or by an operator and no other alarm condition is shelved, then value is set to the current time.</td></tr>
<tr><td>Status</td><td>DINT</td><td>Combined status indicators:
<table>
<tr><th>Status Flag</th><th>CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers</th><th>CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers</th></tr>
<tr><td>Status.0 = InstructFault</td><td>X</td><td>X</td></tr>
<tr><td>Status.1 = InFaulted</td><td>X</td><td>X</td></tr>
<tr><td>Status.2 = SeverityInv</td><td>X</td><td>X</td></tr>
<tr><td>Status.3 = AlarmLimitsInv</td><td>X</td><td>X</td></tr>
<tr><td>Status.4 = Deadband Inv</td><td>X</td><td>X</td></tr>
<tr><td>Status.5 = ROCPosLimitInv</td><td>X</td><td>X</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Output Parameter</th><th>Data Type</th><th>Description</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>Status.6 = ROCNegLimitInv</td><td>X</td><td>X</td></tr>
<tr><td>Status.7 = ROCPeriod Inv</td><td>X</td><td>X</td></tr>
<tr><td>Status.8 = Overflow</td><td>-</td><td>X</td></tr>
</table></td></tr>
<tr><td>InstructFault (Status.0)</td><td>BOOL</td><td>Instruction error conditions exist. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.</td></tr>
<tr><td>InFaulted (Status.1)</td><td>BOOL</td><td>User program has set InFault to indicate bad quality input data. Alarm continues to evaluate In for alarm conditions.</td></tr>
<tr><td>SeverityInv (Status.2)</td><td>BOOL</td><td>Alarm severity configuration is invalid.
If severity <1, the instruction uses Severity = 1.
If severity >1000, the instruction uses Severity = 1000.</td></tr>
<tr><td>AlarmLimitsInv (Status.3)</td><td>BOOL</td><td>Alarm Limit configuration is invalid (for example, LLimit < LLLimit). If invalid, the instruction clears all level conditions active bits. Until the fault is cleared, no new level conditions can be detected.</td></tr>
<tr><td>DeadbandInv (Status.4)</td><td>BOOL</td><td>Deadband configuration is invalid. If invalid, the instruction uses
Deadband = 0.0.
Valid = 0 = Deadband < Span from first enabled low alarm to the first enabled high alarm.</td></tr>
<tr><td>ROCPosLimitInv (Status.5)</td><td>BOOL</td><td>Positive rate-of-change limit invalid.
If invalid, the instruction uses ROCPosLimit = 0.0, which makes positive rate-of-change detection unavailable.</td></tr>
<tr><td>ROCNegLimitInv (Status.6)</td><td>BOOL</td><td>Negative rate-of-change limit invalid.
If invalid, the instruction uses ROCNegLimit = 0.0, which makes negative rate-of-change detection unavailable.</td></tr>
</table>

Output Parameter	Data Type	Description
ROCPeriodInv (Status.7)	BOOL	Rate-of-change period invalid. If invalid, the instruction uses ROCPeriod = 0.0, which makes rate-of-change detection unavailable.
Overflow (Status.8)	BOOL	The Overflow bit is set to true when an overflow condition is detected. The overflow bit is cleared to false when the overflow condition has been corrected. Applies toL8 controllers for tables in topics on page only.

Connect a button to the OperShelve tag

The alarm instruction only processes the OperShelve tag on transition from cleared to set to prevent unwanted reshelving of the alarm. For example, if an operator presses a push button to shelve the alarm while the ProgUnshelve tag is set, the alarm is not shelved because the ProgUnshelve tag takes precedence. To shelve the alarm, the operator can release and press the push button again once ProgUnshelve is cleared.

Affects Math Status Flags

Controllers	Affected Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

A minor fault will occur if:	Fault Type	Fault Code
The input value is INF or NAN for CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers only.	4	4

See Math status flags on page 849.

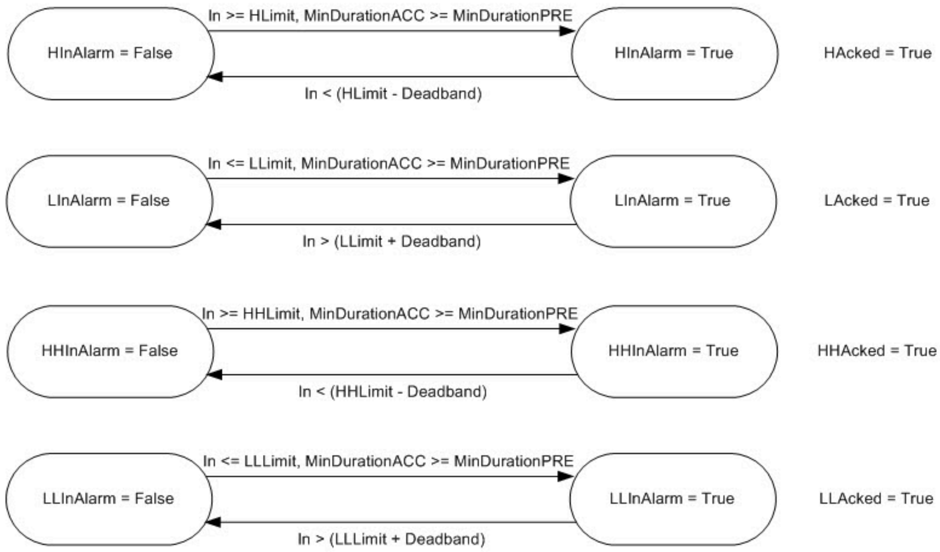
Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Analog Alarm State Diagrams

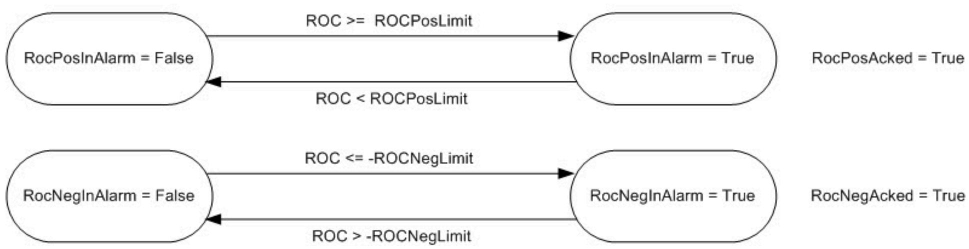
These illustrations show the manner in which an analog alarm responds to changing alarm conditions and operator commands.

AckRequired = False

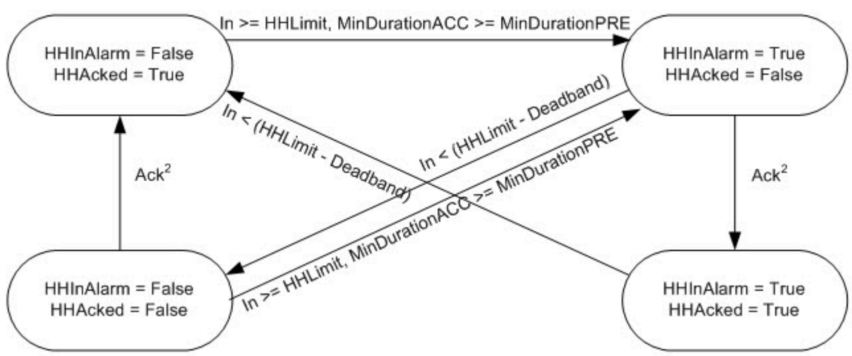


$$\mathrm{ROC} = \frac{\mathrm{In(Current\,Sample)} - \mathrm{In(Previous\,Sample)}}{\mathrm{ROCPeriod}}$$

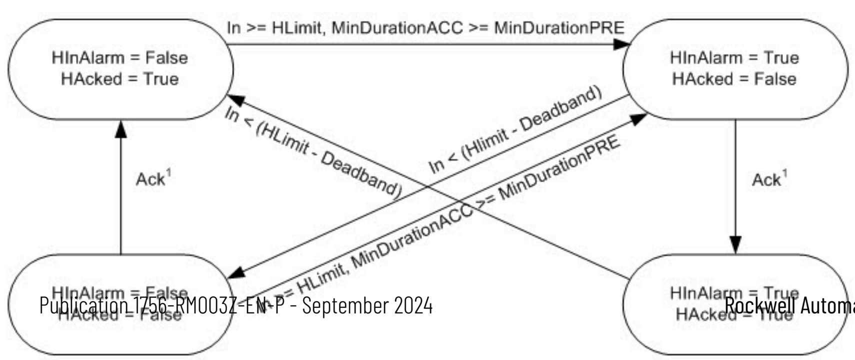
Where a new sample is collected on the next scan after the ROCPeriod has elapsed.



AckRequired = True



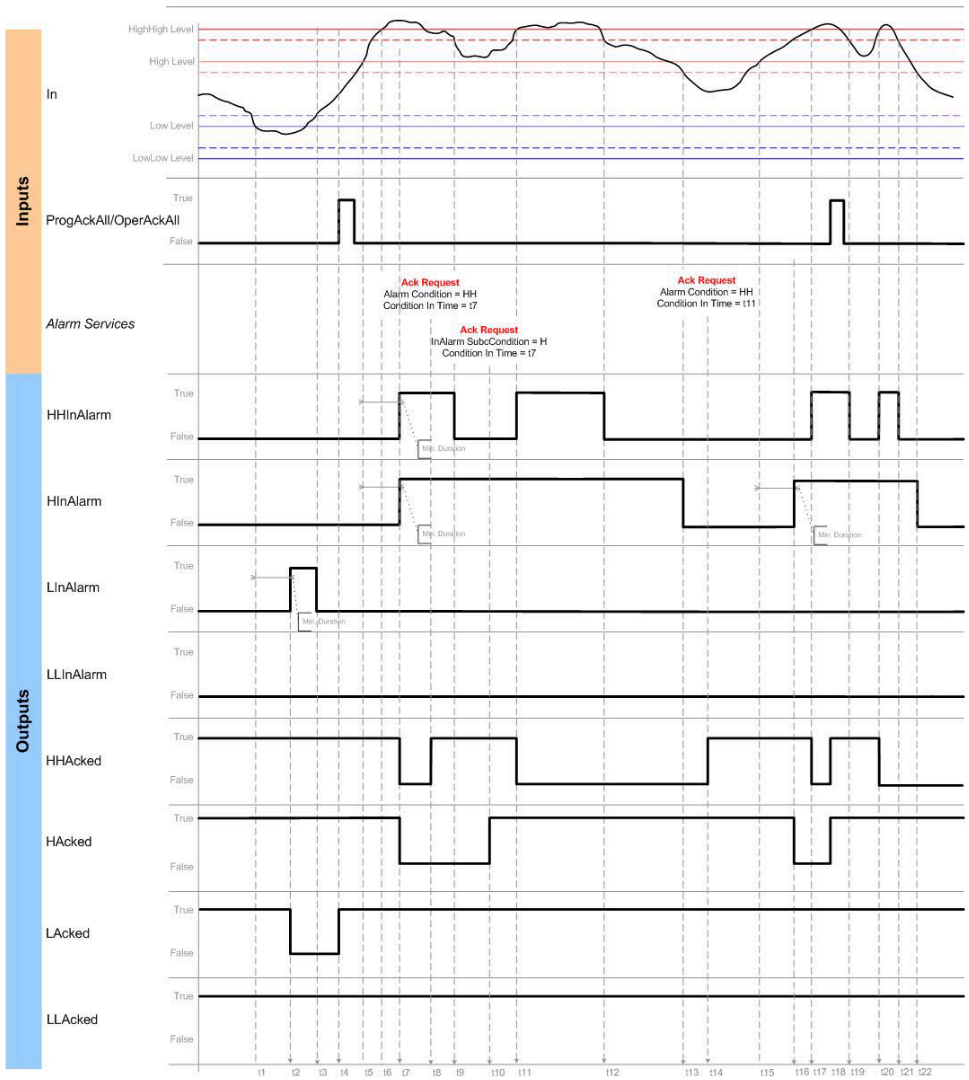
[2] HH alarm condition can be acked by several different ways: HHProgAck, HHOperAck, ProgAckAll, OperAckAll, clients (RSLogix 5000, RSview)



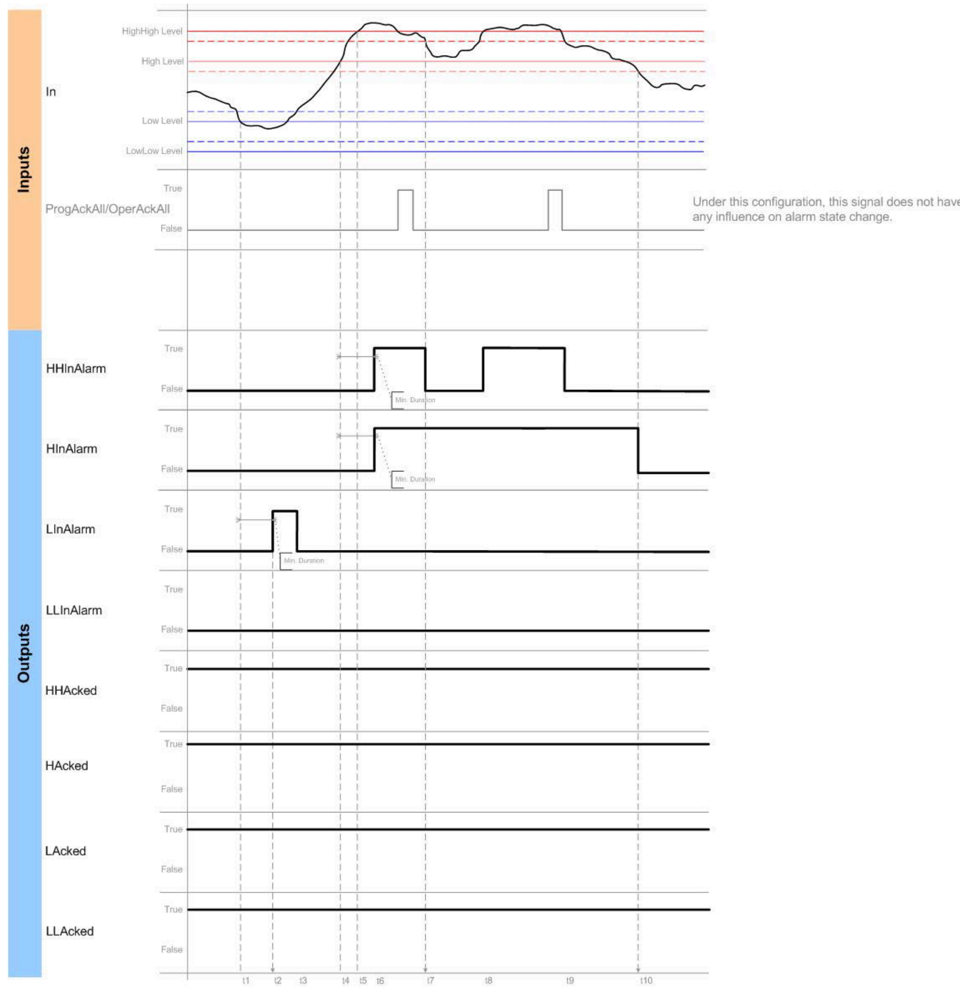
Analog Alarm Timing Diagrams

These timing diagrams show the sequence of analog alarm operations.

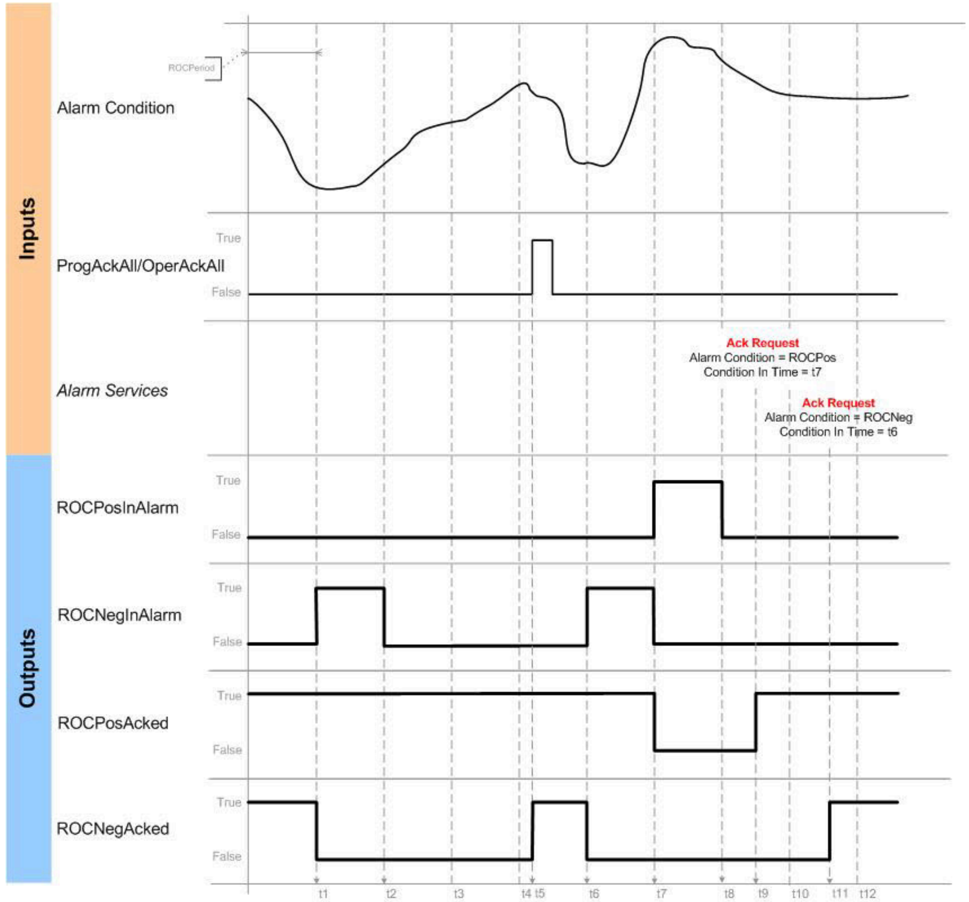
Level Conditions Behavior Acknowledge



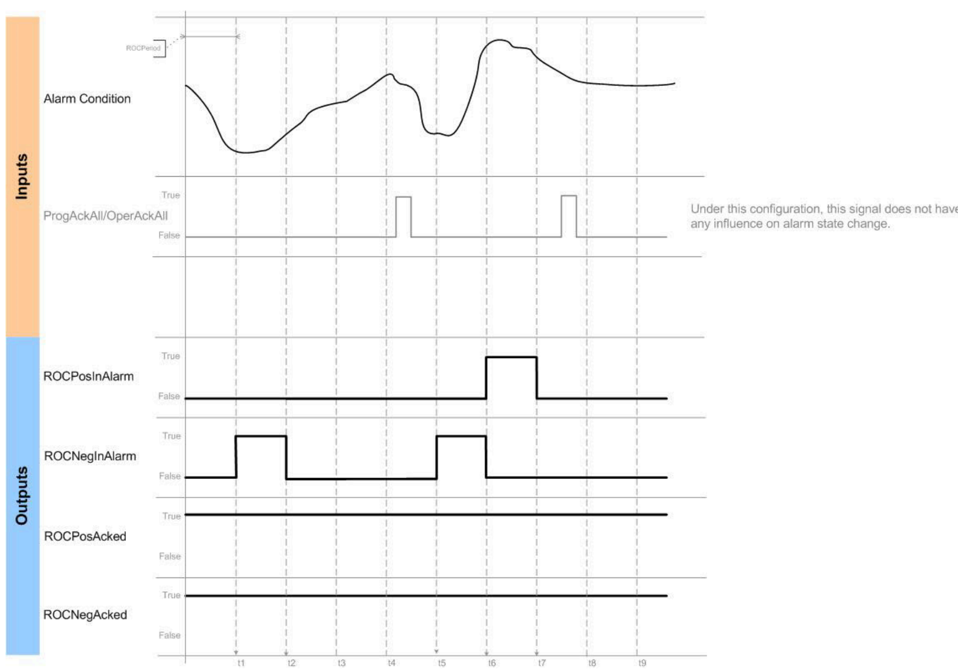
Level Conditions Behavior No Acknowledge



ROC Conditions Behavior Acknowledge



ROC Conditions Behavior No Acknowledge



Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Rung-condition-out is cleared to false. All of the ALMA structure parameters are cleared All alarm conditions are acknowledged. All operator requests are cleared All timestamps are cleared All delivery flags are cleared.
Rung-condition-in is false	Rung-condition-out is cleared to false.
Rung-condition-in is true	Rung-condition-out is set to true The instruction executes
Postscan	Rung-condition-out is cleared to false

Function Block

Condition/State	Action Taken
Prescan	Tag.EnableOut is cleared to false. All of the ALMA structure parameters are cleared All alarm conditions are acknowledged. All operator requests are cleared All timestamps are cleared All delivery flags are cleared.
Tag.EnableIn is false	Tag.EnableOut is cleared to false
Tag.EnableIn is true	The instruction executes Tag.EnableOut is set to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	Tag.EnableOut is cleared to false

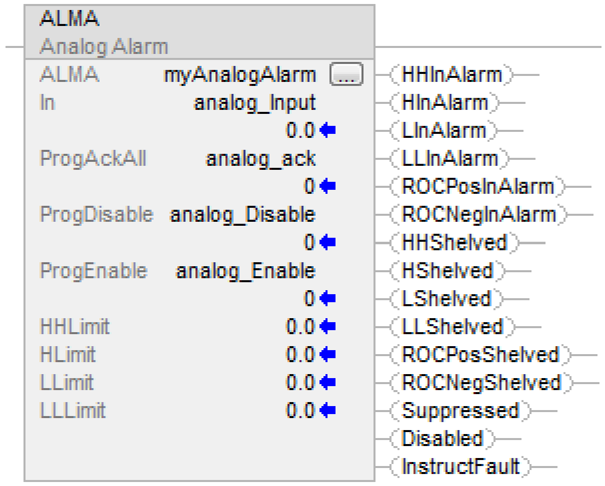
Structured Text

In Structured Text, EnableIn is always true during normal scan. Therefore, if the instruction is in the control path activated by the logic it will execute.

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal Execution	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

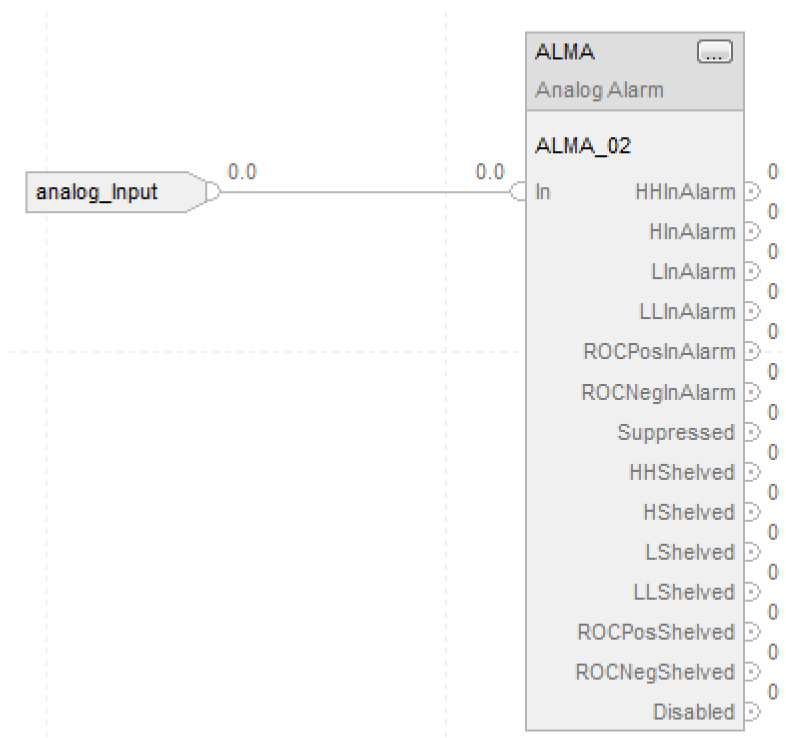
Examples

Ladder Diagram



Function Block

An example of an ALMA instruction in Function Block is shown below. In this example, the Tank 32 Level Transmitter (Tank32LT) is monitored for alarm conditions. The Tank32LevelAck tag can be used to acknowledge all conditions of this alarm.



Structured Text

In this example, the Tank 32 Level Transmitter (Tank32LT) is monitored for alarm conditions. The Tank32LevelAck tag can be used to acknowledge all conditions of this alarm.

```
ALMA(Tank32Level,Tank32LT,Tank32LevelAck,0, 0);
```

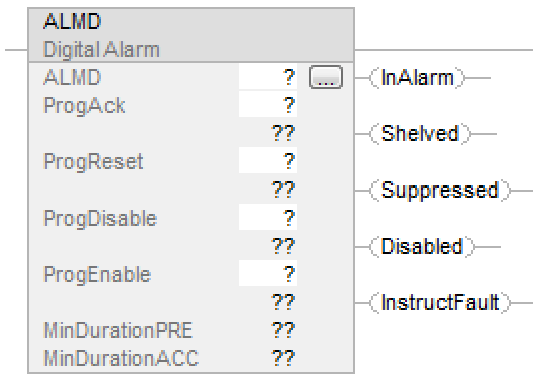
Digital Alarm (ALMD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

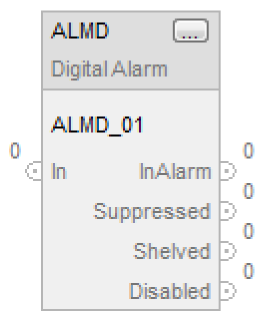
The Digital Alarm (ALMD) instruction provides alarming for any discrete Boolean value.

Available Languages

Ladder Diagram



Function Block



Structured Text

```
ALMD (ALMD,In,ProgAck,ProgReset,ProgDisable,ProgEnable)
```

Operands

Ladder Diagram

Operand	Type	Format	Description
ALMD tag	ALARM_DIGITAL	Structure	ALMD structure
ProgAck	BOOL	Tag Immediate	On transition from false to true, acknowledges alarm (if acknowledgment is required).

Operand	Type	Format	Description
ProgReset	BOOL	Tag Immediate	On transition from false to true, resets alarm (if resetting is required).
ProgDisable	BOOL	Tag Immediate	When True, disables alarm (does not override Enable Commands).
ProgEnable	BOOL	Tag Immediate	When True, enables alarm (takes precedence over Disable commands).
MinDurationPRE	DINT	Immediate	Specifies how long the alarm condition must be met before it is reported (milliseconds).
MinDurationACC	DINT	Immediate	Indicates the current accumulator value for the alarm's MinDuration timer. This value is not used in versions 29 and later of the Logix Designer application. The value is always 0.

Function Block

Operand	Type	Format	Description
ALMD tag	ALARM_DIGITAL	structure	ALMD structure

Structured Text

Operand	Type	Format	Description
ALMD tag	ALARM_DIGITAL	Structure	ALMD structure
ProgAck	BOOL	Tag Immediate	On transition from false to true, acknowledges alarm (if acknowledgment is required).
ProgReset	BOOL	Tag Immediate	On transition from false to true, resets alarm (if resetting is required).
ProgDisable	BOOL	Tag Immediate	When True, disables alarm (does not override Enable Commands).
ProgEnable	BOOL	Tag Immediate	When True, enables alarm (takes precedence over Disable commands).

Operand	Type	Format	Description
MinDurationPRE	DINT	Immediate	Specifies how long the alarm condition must be met before it is reported (milliseconds).
MinDurationACC	DINT	Immediate	Indicates the current accumulator value for the alarm's MinDuration timer. This value is not used in versions 29 and later of the Logix Designer application. The value is always 0.

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

ALMD Structure

Input Parameters

Input Parameter	Data Type	Description
EnableIn	BOOL	Ladder Diagram: Corresponds to the rung state. Does not affect processing. Function Block: If cleared to false, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is true. Structured Text: No effect. The instruction always executes.
In	BOOL	The digital signal input to the instruction. Default is false. Ladder Diagram: Follows the rung condition. Set to true if the rung condition is true. Cleared to false if the rung condition is false. Structured Text: Copied from instruction operand.
InFault	BOOL	Bad health indicator for the input. The user application may set InFault to indicate the input signal has an error. When set, the instruction sets InFaulted (Status.1). When cleared to false, the instruction clears the InFaulted (Status.1) to false. In either case, the instruction

Input Parameter	Data Type	Description
		continues to evaluate In for alarm conditions. Default is false (good health).
Condition	BOOL	Specifies how alarm is activated. When Condition is set to true, the alarm condition is activated when In is set to true. When Condition is cleared to false, the alarm condition is activated when In is Cleared to false. Default is true.
AckRequired	BOOL	Specifies whether alarm acknowledgment is required. When set to true, acknowledgment is required. When cleared to false, acknowledgment is not required and Acked is always set to true. Default is true.
Latched	BOOL	Specifies whether the alarm is latched. Latched alarms remain InAlarm when the alarm condition becomes false, until a Reset command is received. When set to true, the alarm is latched. When cleared to false, the alarm is unlatched. Default is false. A latched alarm can only be reset when the alarm condition is false.
ProgAck	BOOL	Set to true by the user program to acknowledge the alarm. Takes effect only if the alarm is unacknowledged. Requires a false-to-true transition. Default is false. Ladder Diagram: Copied from the instruction operand. Structured Text: Copied from the instruction operand.
OperAck	BOOL	Set to true by the operator interface to acknowledge the alarm. Takes effect only if the alarm is unacknowledged. The instruction clears this parameter. Default is false.
ProgReset	BOOL	Set to true by the user program to reset the latched alarm. Takes effect only if the latched alarm is InAlarm and the

Input Parameter	Data Type	Description
		alarm condition is false. Requires a false-to-true transition. Default is false. Ladder Diagram: Copied from the instruction operand. Structured Text: Copied from the instruction operand.
OperReset	BOOL	Set to true by the operator interface to reset the latched alarm. Takes effect only if the latched alarm is InAlarm and the alarm condition is false. The alarm instruction clears this parameter to false. Default is false.
ProgSuppress	BOOL	Set to true by the user program to suppress the alarm. Default is false.
OperSuppress	BOOL	Set to true by the operator interface to suppress the alarm. The alarm instruction clears this parameter to false. Default is false.
ProgUnsuppress	BOOL	Set to true by the user program to unsuppress the alarm. Takes precedence over Suppress commands. Default is false.
OperUnsuppress	BOOL	Set to true by the operator interface to unsuppress the alarm. Takes precedence over Suppress commands. The alarm instruction clears this parameter to false. Default is false.
OperShelve	BOOL	Set to true by the operator interface to shelve or reshelve the alarm. Requires a transition from false in one program scan to true in the next program scan. The alarm instruction clears this parameter to false. Default is false. Unshelve commands take precedence over Shelve commands. Shelving an alarm postpones alarm processing. It is like suppressing an alarm, except that shelving is time limited. If an alarm is acknowledged while it is shelved, it remains acknowledged

Input Parameter	Data Type	Description
		even if it becomes active again. It becomes unacknowledged when the shelve duration ends provided the alarm is still active at that moment.
ProgUnshelve	BOOL	Set to true by the user program to unshelve the alarm. Takes precedence over Shelve commands. Default is false. For more information on shelving an alarm, see the description for the OperShelve parameter.
OperUnshelve	BOOL	Set to true by the operator interface to unshelve the alarm. The alarm instruction clears this parameter to false. Takes precedence over Shelve commands. Default is cleared. For more information on shelving an alarm, see the description for the OperShelve parameter.
ProgDisable	BOOL	Set to true by the user program to disable the alarm. Default is false. Ladder Diagram: Copied from the instruction operand. Structured Text: Copied from the instruction operand.
OperDisable	BOOL	Set to true by the operator interface to disable the alarm. The alarm instruction clears this parameter to true. Default is false.
ProgEnable	BOOL	Set to true by the user program to enable the alarm. Takes precedence over a Disable command. Default is false. Ladder Diagram: Copied from the instruction operand. Structured Text: Copied from the instruction operand.
OperEnable	BOOL	Set to true by the operator interface to enable the alarm. Takes precedence over Disable command. The alarm instruction clears this parameter to false. Default is false.

Input Parameter	Data Type	Description
AlarmCountReset	BOOL	Set to true by the operator interface to reset the alarm count to zero. The alarm instruction clears this parameter to false. Default is false.
UseProgTime	BOOL	Specifies whether to use the controller's clock or the ProgTime value to timestamp alarm state change events. When set to true, the ProgTime value provides timestamp. When cleared to false, the controller's clock provides timestamp. Default is false.
ProgTime	LINT	If UseProgTime is set to true, this value is used to provide the timestamp value for all events. This lets the application apply timestamps obtained from the alarm source, such as a sequence-of-events input module.
Severity	DINT	Severity of the alarm. This does not affect processing of alarms by the controller, but can be used for sorting and filtering functions at the alarm subscriber. Valid = 1...1000 (1000 = most severe; 1 = least severe). Default = 500.
MinDurationPRE	DINT	Minimum duration preset (milliseconds) for the alarm condition to remain true before the alarm is marked as InAlarm and alarm notification is sent to clients. The controller collects alarm data as soon as the alarm condition is detected; so no data is lost while waiting to meet the minimum duration. Valid = 0...2147483647. Default = 0.
ShelveDuration	DINT	Length of time in minutes to shelve an alarm. Shelving an alarm postpones alarm processing. It is like suppressing an alarm, except that shelving is time limited. If an alarm is acknowledged while it is shelved, it remains acknowledged even if it becomes active again. It becomes unacknowledged when the

Input Parameter	Data Type	Description
		shelve duration ends (provided the alarm is still active at that time). Minimum time is one minute. Maximum time is defined by MaxShelveDuration.
MaxShelveDuration	DINT	Maximum time duration in minutes for which an alarm can be shelved. For more information on shelving an alarm, see the description for the ShelveDuration parameter.

Output Parameters

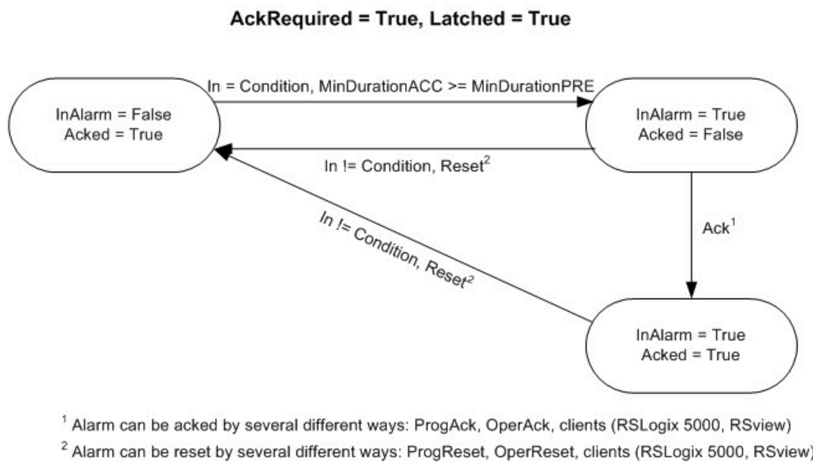
Output Parameter	Data Type	Description
EnableOut	BOOL	Enable output.
InAlarm	BOOL	Alarm active status. Set to true when the alarm is active. Cleared to false when the alarm is not active (normal status).
Acked	BOOL	Alarm acknowledged status. Set to true when the alarm is acknowledged. Cleared to false when the alarm is not acknowledged. Acked is always set to true when AckRequired is cleared to false.
InAlarmUnack	BOOL	Combined alarm active and acknowledged status. Set to true when the alarm is active (InAlarm is true) and unacknowledged (Acked is false). Cleared to false when the alarm is inactive, acknowledged, or both.
Suppressed	BOOL	Suppressed status of the alarm. Set to true when the alarm is suppressed. Cleared to falsewhen the alarm is not suppressed.
Shelved	BOOL	Shelved status of the alarm. Set to true when alarm is shelved. Cleared to false when alarm is unshelved. Shelving an alarm postpones alarm processing. It is like suppressing an alarm, except that shelving is time limited. If an alarm is acknowledged while it is shelved, it remains acknowledged even if it becomes active again. It

Output Parameter	Data Type	Description
		becomes unacknowledged when the shelve duration ends.
Disabled	BOOL	Disabled status of the alarm. Set to true when the alarm is not enabled. Cleared to false when the alarm is enabled.
Commissioned	BOOL	Commissioned status of the alarm. Set to true when the alarm is commissioned. Cleared to false when the alarm is decommissioned. Currently always set to true.
MinDurationACC	DINT	Indicates the current accumulator value for the alarm's MinDuration timer. This value is not used in versions 29 and later of the Logix Designer application. The value is always 0.
AlarmCount	DINT	Number of times the alarm has been activated (InAlarm is set). If the maximum value is reached, the counter leaves the value at the maximum count value.
InAlarmTime	LINT	Timestamp of alarm detection.
AckTime	LINT	Timestamp of alarm acknowledgment. If the alarm does not require acknowledgment, this timestamp is equal to alarm time.
RetToNormalTime	LINT	Timestamp of alarm returning to a normal state.
AlarmCountResetTime	LINT	Timestamp indicating when the alarm count was reset.
ShelveTime	LINT	Timestamp indicating when the alarm was shelved the last time. This value is set by controller when alarm is shelved. Alarm can be shelved and unshelved many times. Each time the alarm is shelved, the timestamp is set to the current time. For more information on shelving an alarm, see the description for the Shelved parameter.
UnshelveTime	LIN	Timestamp indicating when the alarm is going to be unshelved. This value is set every time the alarm is shelved (even if

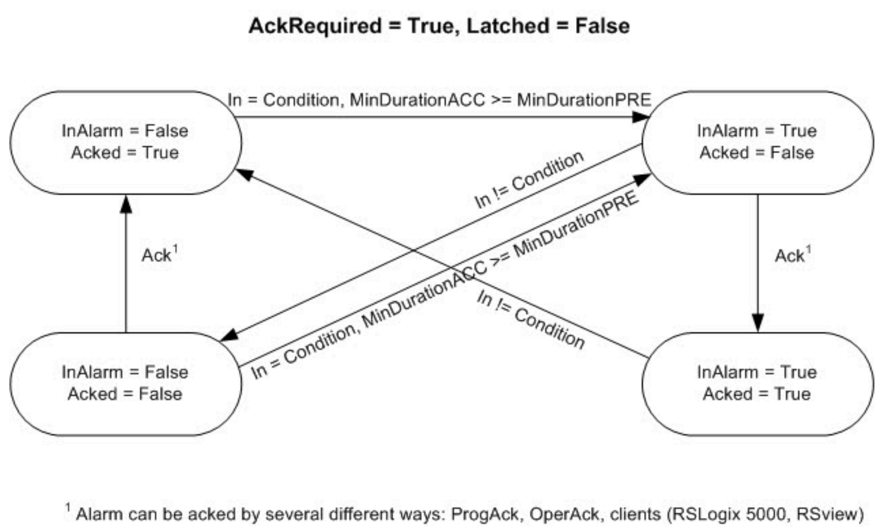
Output Parameter	Data Type	Description
		the alarm has already been shelved). The timestamp is determined by adding the ShelveDuration to the current time. If the alarm is unshelved programmatically or by an operator, then the value is set to the current time. For more information on shelving an alarm see the description for the Shelved parameter.
Status	DINT	Combined status indicators: Status.0 = InstructFault Status.1= InFaulted Status.2 = SeverityInv
InstructFault (Status.0)	BOOL	Instruction error conditions exist. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
InFaulted (Status.1)	BOOL	User program has set InFault to indicate bad quality input data. Alarm continues to evaluate In for alarm condition.
SeverityInv (Status.2)	BOOL	Alarm severity configuration. If severity <1, the instruction uses Severity = 1. If severity >1000, the instruction uses Severity = 1000.

Digital Alarms State Diagrams

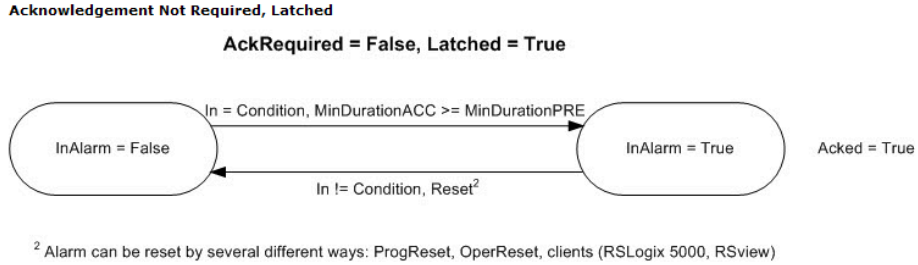
Acknowledgement Required, Latched



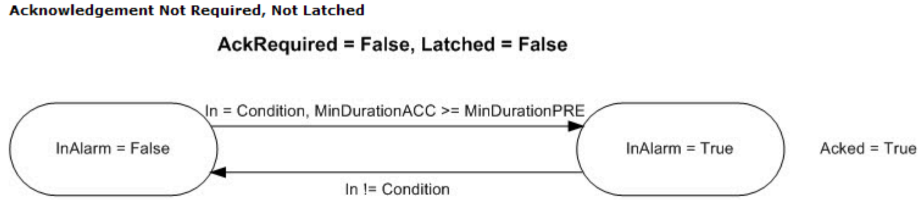
Acknowledgement Required, Not Latched



Acknowledgement Not Required, Latched

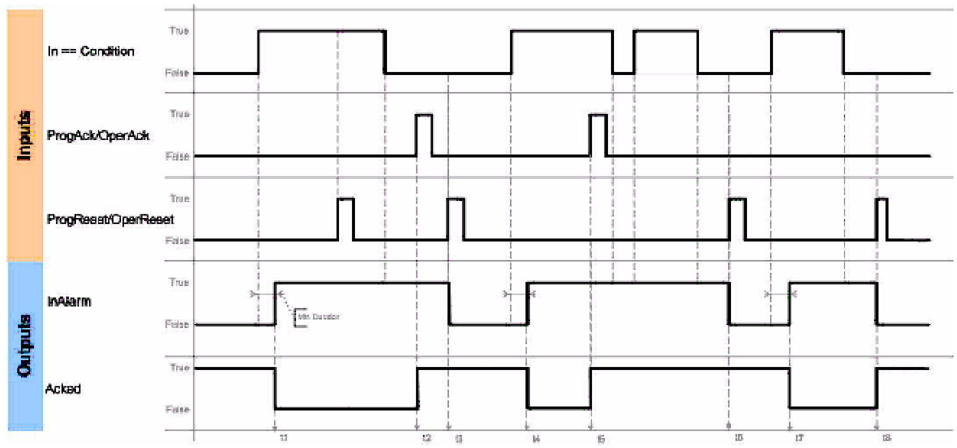


Acknowledgement Not Required, Not Latched

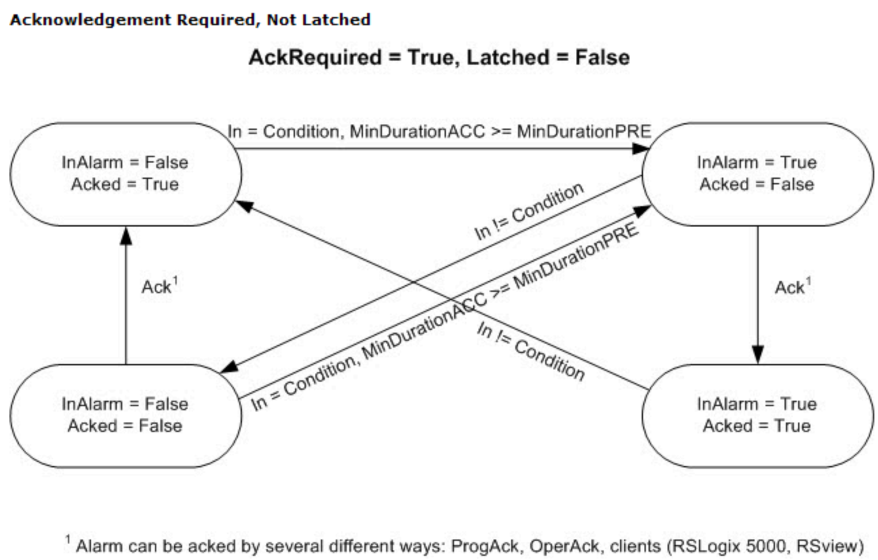


Digital Alarm Timing Diagrams

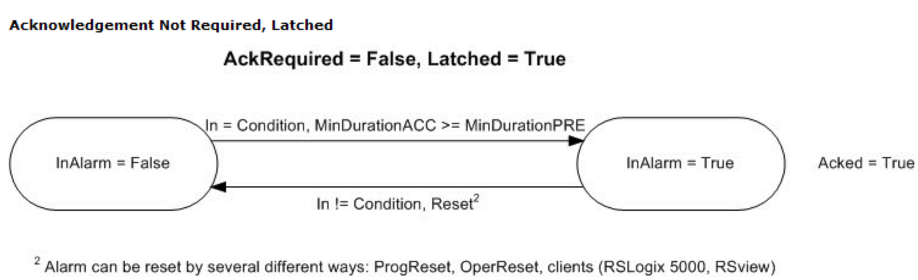
ALMD Alarm Acknowledge Required and Latched

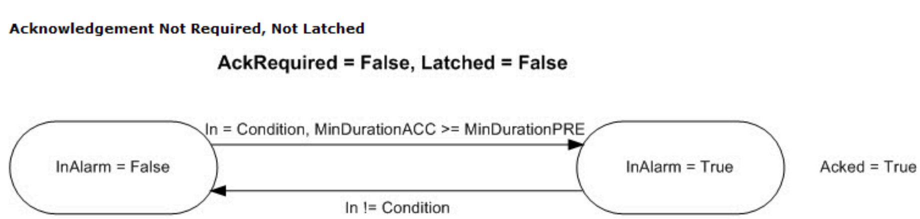


ALMD Alarm Acknowledge Required and Not Latched

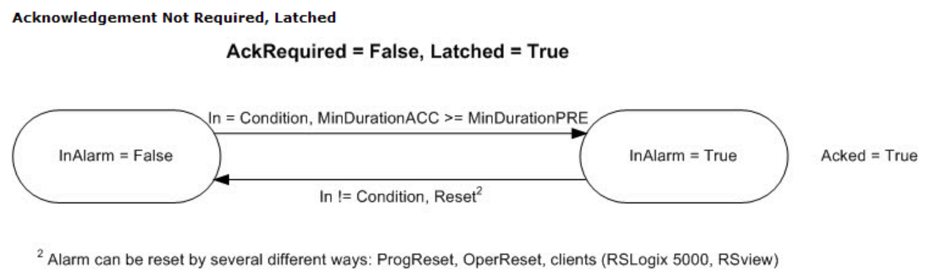


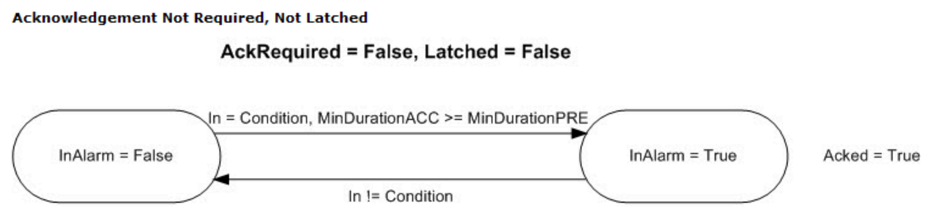
ALMD Alarm Acknowledge Not Required and Latched

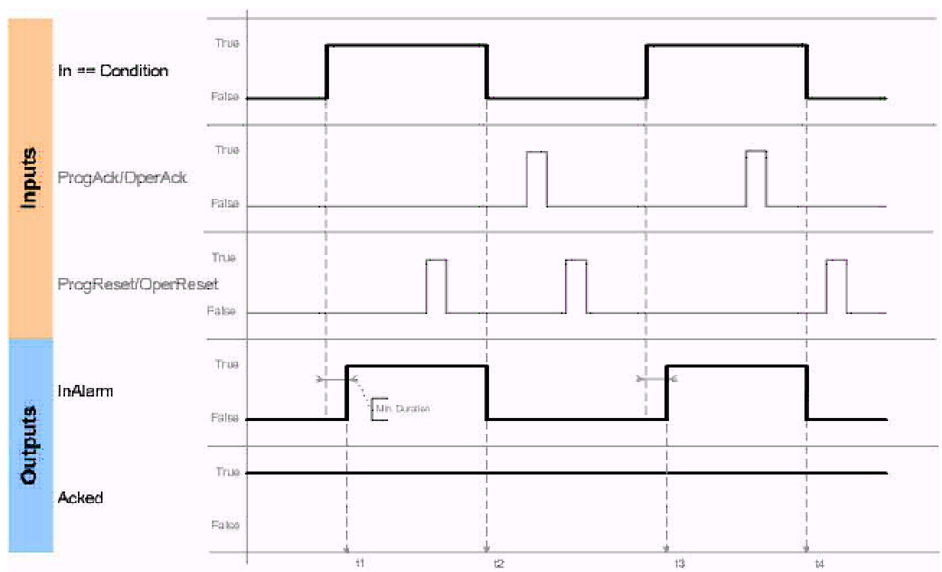




ALMD Alarm Acknowledge Not Required and Not Latched







Connect a button to the OperShelve tag

To prevent unwanted reshelving of the alarm, the alarm instruction only processes the OperShelve tag if it transitions from false to true between one program scan and the next. If an operator presses a push button to shelve the alarm while the ProgUnshelve tag is true, the alarm is not shelved because the ProgUnshelve tag takes precedence. However, because program scans complete in milliseconds, the operator may still be holding down the push button so that the OperShelve tag remains true over several program scans even though the ProgUnshelve tag has been cleared to false. This means that the alarm is not shelved.

To shelve the alarm, the operator can release and press the button again

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	EnableOut is cleared to false to false The InAlarm output is cleared to false The Shelved output is cleared to false The Acked ouput is set to true. All alarm conditions are acknowledged. All operator requests are cleared All timestamps are cleared
Rung-condition-in is false	Rung is cleared to false. The In parameter is cleared to false The instruction executes.
Rung-condition-in is true	Rung is set to true. The In parameter is set to true The instruction executes.
Postscan	Rung bit is cleared to false.

Function Block

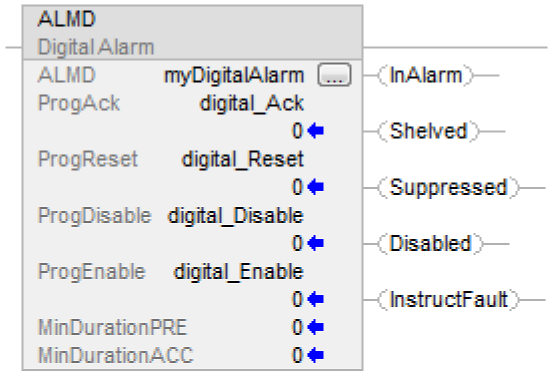
Condition/State	Action Taken
Prescan	Tag.EnableOut is cleared to false. The InAlarm output is cleared to false The Shelved output is cleared to false The Acked ouput is set to true All operator requests are cleared All timestamps are cleared
Tag.EnableIn is false	Tag.EnableOut is cleared to false
Tag.EnableIn is true	The instruction executes Tag.EnableOut is set to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	Tag.EnableOut is cleared to false.

Structured Text

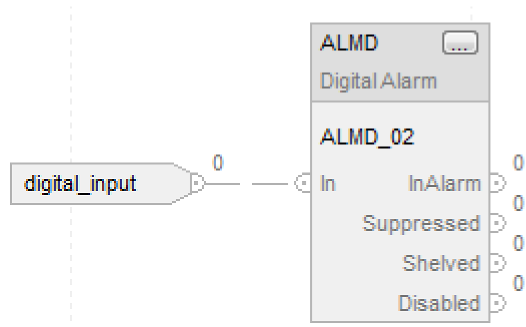
Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal Execution	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Example

Ladder Diagram



Function Block



Structured Text

An example of an ALMD instruction in Structured Text is shown below. In this example, two motor failure signals are combined such that if either one occurs, a motor fault alarm is activated. The Motor101Ack tag can be used to acknowledge the alarm.

Motor101FaultConditions := Motor101Overtemp OR Motor101FailToStart;

```
ALMD(Motor101Fault,Motor101FaultConditions,Motor101Ack,0,0,0 );
```

Bit Instructions

Use the bit (relay-type) instructions to monitor and control the status of bits, such as input bits or timer-control word bits.

Available Instructions

Ladder Diagram

If you want to:	Use this instruction:
Enable outputs when a bit is set	XIC on page 80
Enable outputs when a bit is cleared	XIO on page 84
set a bit	OTE on page 76
set a bit (retentive)	OTL on page 77
clear bit (retentive)	Output Latch (OTL) on page 77
Enable outputs for one scan each time a rung goes true	ONS on page 65
set a bit for one scan each time a rung goes true	OSR on page 72
set a bit for one scan each time the rung goes false	OSF on page 67
set a bit for one scan each time the input is set in function block	One Shot Rising with Input (OSRI) on page 74
set a bit for one scan each time the input is cleared in funciton block	One Shot Falling with Input (OSFI) on page 69

One Shot (ONS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The One Shot (ONS) instruction makes the remainder of the rung true each time rung-condition-in transitions from false to true.

Available Languages

Ladder Diagram

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Storage bit	BOOL	tag	Internal storage bit. Retains the rung-condition-in from the last time the instruction was executed. There are various operand addressing modes possible for the storage bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

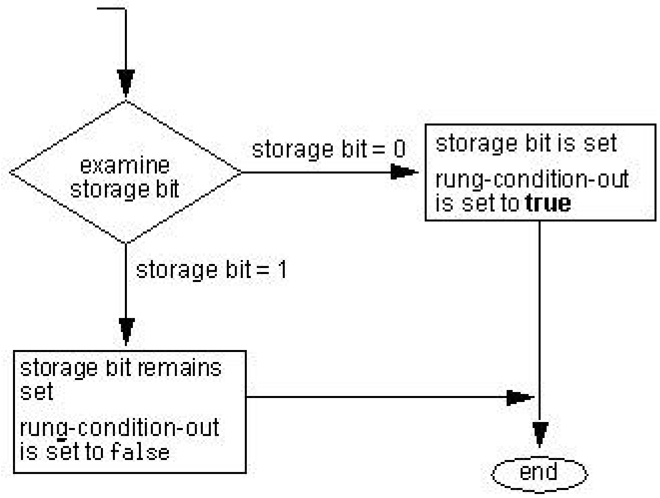
None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

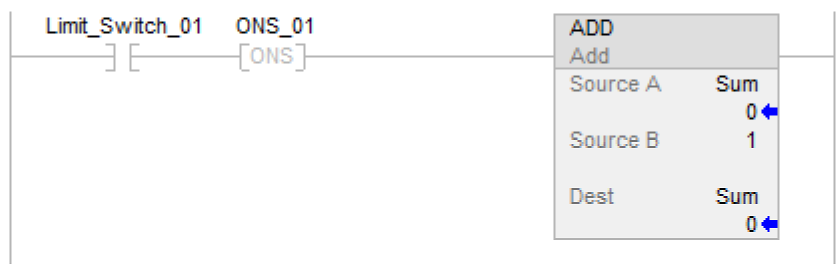
Condition/State	Action Taken
Prescan	The storage bit is set to true to prevent an invalid trigger during the first scan.
Rung-condition-in is false	The storage bit is cleared to false, rung-condition-out is cleared to false.
Rung-condition-in is true	See ONS Flow Chart (True).
Postscan	N/A

ONS Flow Chart (True)



Example

Ladder Diagram



In this example, the sum increments each time limit_switch_1 goes from false to true.

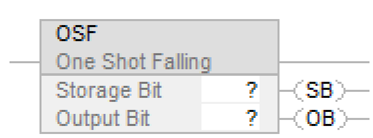
One Shot Falling (OSF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The One Shot Falling (OSF) instruction sets the output bit for one scan when rung-condition-in transitions from true to false.

Available Languages

Ladder Diagram



Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Storage Bit	BOOL	tag	Stores the rung-condition-in from when the instruction was last executed. There are various operand addressing modes possible for the storage bit, see Bit Addressing on page 864 for examples.
Output Bit	BOOL	tag	Bit to be modified.

Affects Math Status Flags

No

Major/Minor Faults

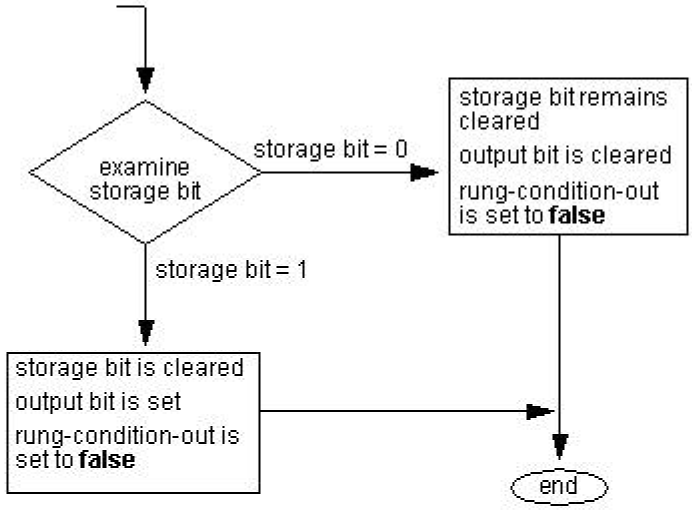
None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The storage bit is cleared to false to prevent an invalid trigger during the first program scan. The output bit is cleared to false.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in. See OSF Flow Chart (False).
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The storage bit is set to true. The output bit is cleared to false.
Postscan	N/A

OSF Flow Chart (False)



Example

Ladder Diagram

This example shows how an OSF can be used to make one or more instructions edge-triggered. Each time Limit_Switch_01 transitions from true to false the OSF will set Output_bit_02 to true. Any instruction conditioned by Output_bit_02 will be enabled and, since Output_bit_02 is only true for one scan, will execute once per transition.

One Shot Falling with Input (OSFI)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

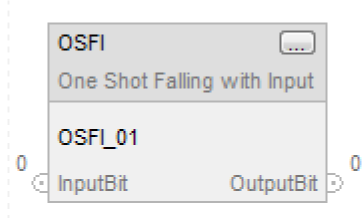
The One Shot Falling with Input (OSFI) instruction sets the OutputBit for one execution cycle when the InputBit toggles from false to true.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

OSFI(OSFI_tag)

Operands

Structured Text

Operand	Type	Format	Description
OSFI tag	FBD_ONESHOT	Structure	OSFI structure

See Structured Text Syntax on page 879 for operand-related faults

Function Block

Operand	Type	Format	Description
OSFI tag	FBD_ONESHOT	Structure	OSFI structure

FBD_ONESHOT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
InputBit	BOOL	Input bit.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
OutputBit	BOOL	Output bit

Description

If InputBit is false, and it was true the last time the instruction was scanned then OutputBit will be set, otherwise OutputBit will be cleared.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes for General Instructions* on page 849 Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Function Block

Condition / State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes
Instruction first run	N/A
Instruction first scan	Previous InputBit history is cleared to require a True to False transition of InputBit.
Postscan	EnableIn and EnableOut bits are cleared to false.

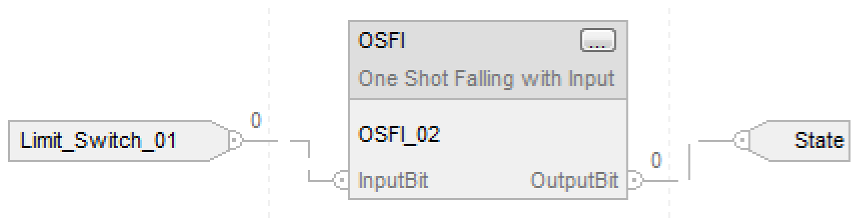
Structured Text

Condition / State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

When limit_switch1 goes from set to cleared, the OSFI instruction sets OutputBit for one scan.

Function Block



Structured Text

OSFI_01.InputBit := limit_switch1;

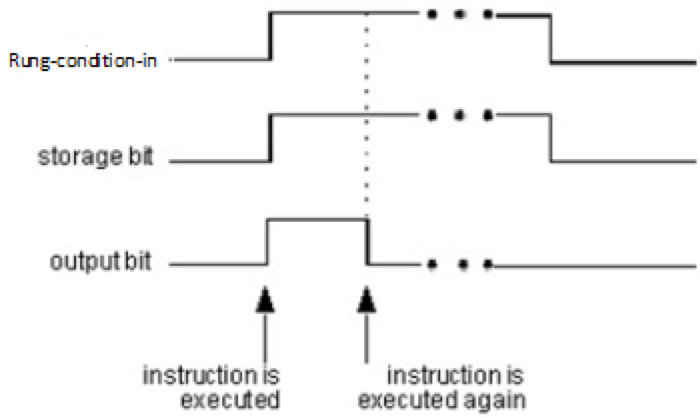
OSFI(OSFI_01);

Output_state := OSFI_01.OutputBit;

One Shot Rising (OSR)

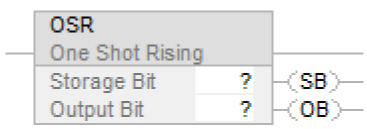
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The One Shot Rising (OSR) instruction sets the output bit for one scan when rung-condition-in transitions from false to true.



Available Languages

Ladder Diagram



Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Storage Bit	BOOL	tag	Stores the rung-condition-in from when the instruction was last executed. There are various operand addressing modes possible for the storage bit, see Bit

Operand	Data Type	Format	Description
			Addressing on page 864 for examples.
Output Bit	BOOL	tag	Bit to be modified.

Affects Math Status Flags

No

Major/Minor Faults

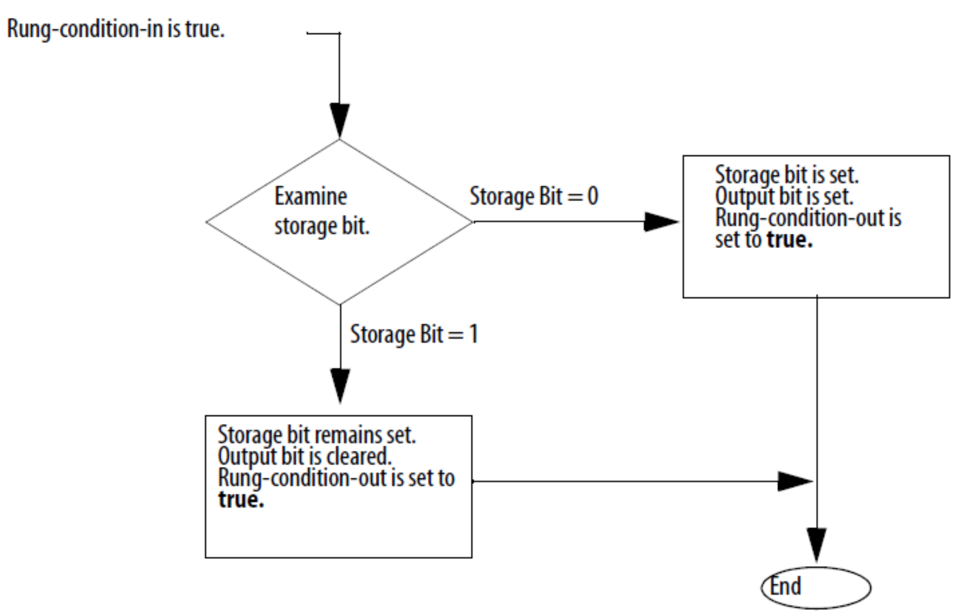
None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The storage bit is set to true to prevent an invalid trigger during the first program scan. The output bit is cleared to false.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in The storage bit is cleared to false. The output bit is cleared to false.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in See OSR Flow Chart (True).
Postscan	N/A

OSR Flow Chart (True)



Example

Ladder Diagram

This example shows how an OSR can be used to make one or more instructions edge-triggered. Each time Limit_Switch_01 transitions from false to true the OSR will set Output_bit_02 to true. Any instruction conditioned by Output_bit_02 will be enabled and, since Output_bit_02 is only true for one scan, will execute once per transition.

One Shot Rising with Input (OSRI)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

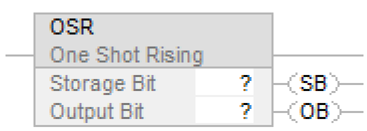
The One Shot Rising with Input (OSRI) instruction sets the output bit for one execution cycle when the input bit toggles from cleared to set.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

OSRI(OSRI_tag);

Operands

Structured Text

Operand	Type	Format	Description
OSRI tag	FBD_ONESHOT	Structure	OSRI structure

Function Block

Operand	Type	Format	Description
OSRI tag	FBD_ONESHOT	Structure	OSRI structure

FBD_ONESHOT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
InputBit	BOOL	Input bit. Default is cleared.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
OutputBit	BOOL	Output bit

Description

If InputBit is true, and it was false the last time the instruction was scanned then OutputBit will be set, otherwise OutputBit will be cleared.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Function Block

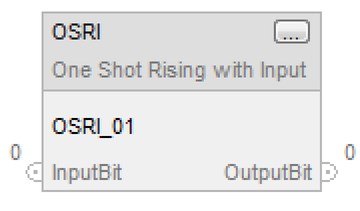
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.Enable-in is false	EnableIn and EnableOut bits are cleared to false.
Tag.Enable-in is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	Previous InputBit history is set to require a False to True transition of InputBit.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table

Examples

Function Block



When limit_switch1 goes from cleared to set, the OSRI instruction sets OutputBit for one scan.

Structured Text

OSRI_01.InputBit := limit_switch1;

OSRI(OSRI_01);

State := OSRI_01.OutputBit;

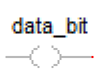
Output Energize (OTE)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Output Energize (OTE) instruction sets or clears the data bit based on rung condition.

Available Languages

Ladder Diagram



Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be modified. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The data bit is cleared to false
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in. The data bit is cleared to false
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The data bit is set to true.
Postscan	The data bit is cleared to false.

Example

Ladder Diagram

When switch is true, the OTE instruction sets Light_01 to true. When switch is false, the OTE instruction clears Light_01 to false.

Output Latch (OTL)

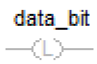
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Output Latch (OTL) instruction sets (latches) the data bit.

When the rung condition is true, the OTL instruction sets the data bit to true. The data bit remains true until it is cleared, typically by an OTU instruction. When the rung condition is changed to false, the OTL instruction does not change the status of the data bit.

Available Languages

Ladder Diagram



Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be modified. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

If the operand is an indirect array reference and the subscript is out of range, then the controller does not generate a major fault when the OTL instruction is false.

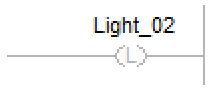
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The data bit is set to true.
Postscan	N/A

Example

Ladder Diagram



When enabled, the OTL instruction turns the light on.

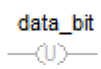
Output Unlatch (OTU)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Output Unlatch (OTU) instruction clears (unlatches) the data bit.

Available Languages

Ladder Diagram



Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten
- Members of a structure operand are overwritten
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be modified. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

If the operand is an indirect array reference and the subscript is out of range, then the controller does not generate a major fault when the OTL instruction is false.

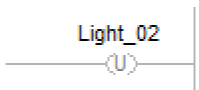
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The data bit is set to true.
Postscan	N/A

Example

Ladder Diagram



When enabled, the OTL instruction clears Light_02.

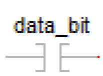
Examine if Closed (XIC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Examine if Closed (XIC) instruction examines the data bit to set or clear the rung condition.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be tested. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

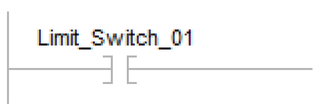
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	If DataBit is true, rung-condition-out is set to true. If DataBit is false, rung-condition-out is cleared to false.
Postscan	N/A

Example 1

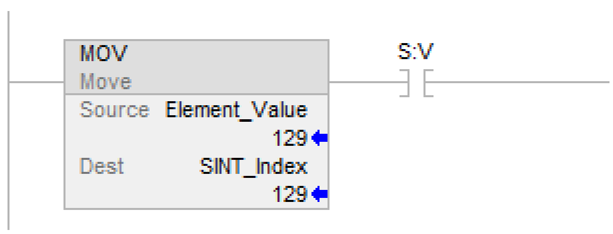
Ladder Diagram



If Limit_Switch_1 is true, the next instruction is enabled.

Example 2

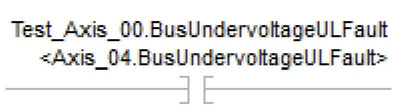
Ladder Diagram



If S:V is true (generated by MOV), the next instruction is enabled.

Example 3

Ladder Diagram



XIC Access LINT Number

Axis_04 is an AXIS_CIP_DRIVE tag.

Test_Axis_00 is an Alias for Axis_04.

The AXIS_CIP_DRIVE type has a LINT member called CIPAxisFaults.

BusUndervoltageULFault is a bit member of CIPAxisFaults.

Test_Axis_00.BusUndervoltageULFault is bit 34 of CIPAxisFaults. The bit 34 value is 0x400000000.

If Test_Axis_00.BusUndervoltageULFault is true, this enables the next instruction.

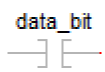
Examine if Closed (XIC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Examine if Closed (XIC) instruction examines the data bit to set or clear the rung condition.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be tested. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864 for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

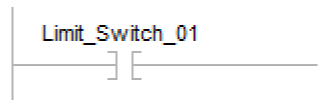
Ladder Diagram

Condition/State	Action Taken
Prescan	N/A

Condition/State	Action Taken
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	If DataBit is true, rung-condition-out is set to true. If DataBit is false, rung-condition-out is cleared to false.
Postscan	N/A

Example 1

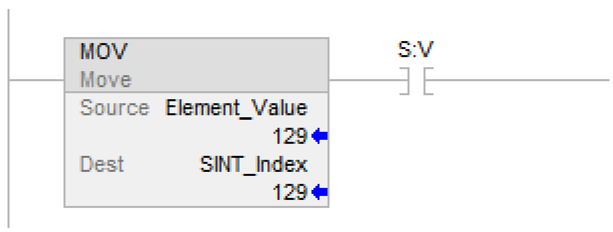
Ladder Diagram



If Limit_Switch_1 is true, the next instruction is enabled.

Example 2

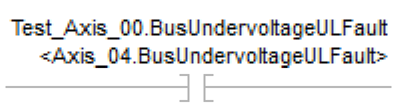
Ladder Diagram



If S:V is true (generated by MOV), the next instruction is enabled.

Example 3

Ladder Diagram



XIC Access LINT Number

Axis_04 is an AXIS_CIP_DRIVE tag.

Test_Axis_00 is an Alias for Axis_04.

The AXIS_CIP_DRIVE type has a LINT member called CIPAxisFaults.

BusUndervoltageULFault is a bit member of CIPAxisFaults.

Test_Axis_00.BusUndervoltageULFault is bit 34 of CIPAxisFaults. The bit 34 value is 0x400000000.

If Test_Axis_00.BusUndervoltageULFault is true, this enables the next instruction.

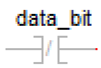
Examine If Open (XIO)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Examine If Open (XIO) instruction examines the data bit to set or clear the rung condition.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

Operand	Data Type	Format	Description
Data bit	BOOL	tag	Bit to be tested. There are various operand addressing modes possible for the data bit, see Bit Addressing on page 864for examples.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

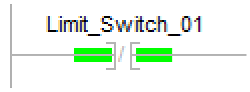
Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	If Data Bit is true, rung-condition-out is cleared to false. If Data Bit is false, rung-condition-out is set to true.
Postscan	N/A

Examples

Example 1

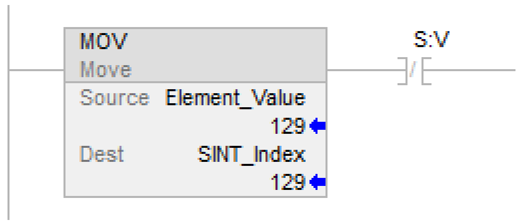
Ladder Diagram



If Limit_Switch_01 is false, the next instruction is enabled.

Example 2

Ladder Diagram



If S:V is false, this enables the next instruction.

Timer and Counter Instructions

Timers and counters control operations based on time or the number of events.

Available Instructions

Ladder Diagram

TON on page 119	TOF on page 111	RTO on page 103	CTU on page 92	CTD on page 87	RES on page 101

Function Block and Structured Text

TONR on page 123	TOFR on page 115	RTOR on page 106	CTUD on page 97

If you want to	Use this instruction
time how long a timer is enabled	TON
time how long a timer is disabled	TOF
accumulate time	RTO
time how long a timer is enabled with built-in reset in function block	TONR
time how long a timer is disabled with built-in reset in function block	TOFR
accumulate time with built-in reset in function block	RTOR
count up	CTU
count down	CTD
count up and count down in function block	CTUD
reset a timer or counter	RES

The time base is 1 msec for all timers. For example, a 2 second timer's .PRE value should be 2000.

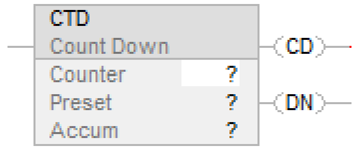
Count Down (CTD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The CTD instruction counts downward each time the rung-condition-in transitions from false to true.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Counter	COUNTER	tag	Counter structure
Preset	DINT	immediate	Value of Counter.PRE.
Accum	DINT	immediate	Value of Counter.ACC.

Preset and Accum (corresponding to .PRE and .ACC in the counter tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

COUNTER Structure

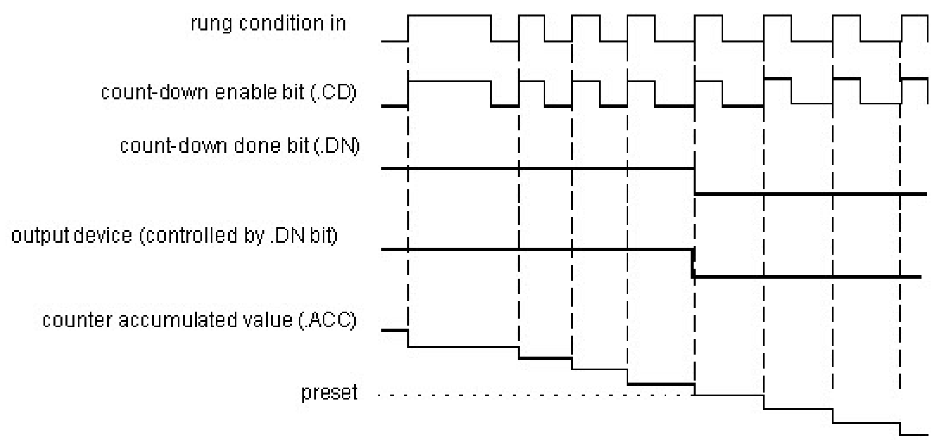
Mnemonic	Data Type	Description
.CD	BOOL	The countdown enable bit contains rung-condition-in when the instruction was last executed.
.DN	BOOL	The done bit when clear indicates the counting operation is complete.
.OV	BOOL	The overflow bit when set indicates the counter incremented past the upper limit of 2,147,483,647.
.UN	BOOL	The underflow when set indicates the counter decremented past the lower limit of -2,147,483,648.

Mnemonic	Data Type	Description
.PRE	DINT	The preset value specifies the value which the accumulated value must reach before the instruction indicates it is done.
.ACC	DINT	The accumulated value specifies the number of transitions the instruction has counted.

Description

The CTD instruction is typically used with a CTU instruction that references the same counter structure.

When rung-condition-in is set to true and .CD is false, .ACC will be decremented by one. When rung-condition-in is false, .CD will be cleared to false.



Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

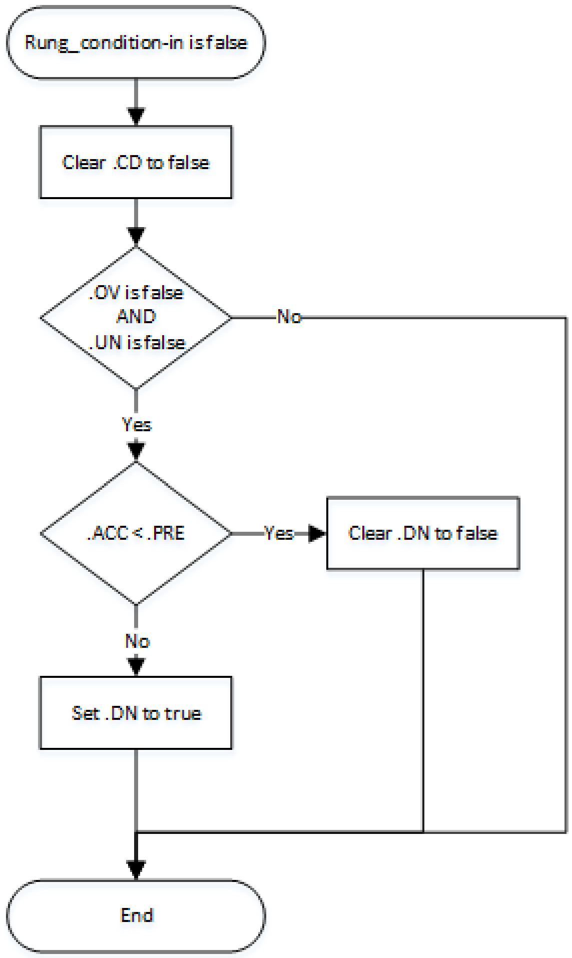
Execution

Ladder Diagram

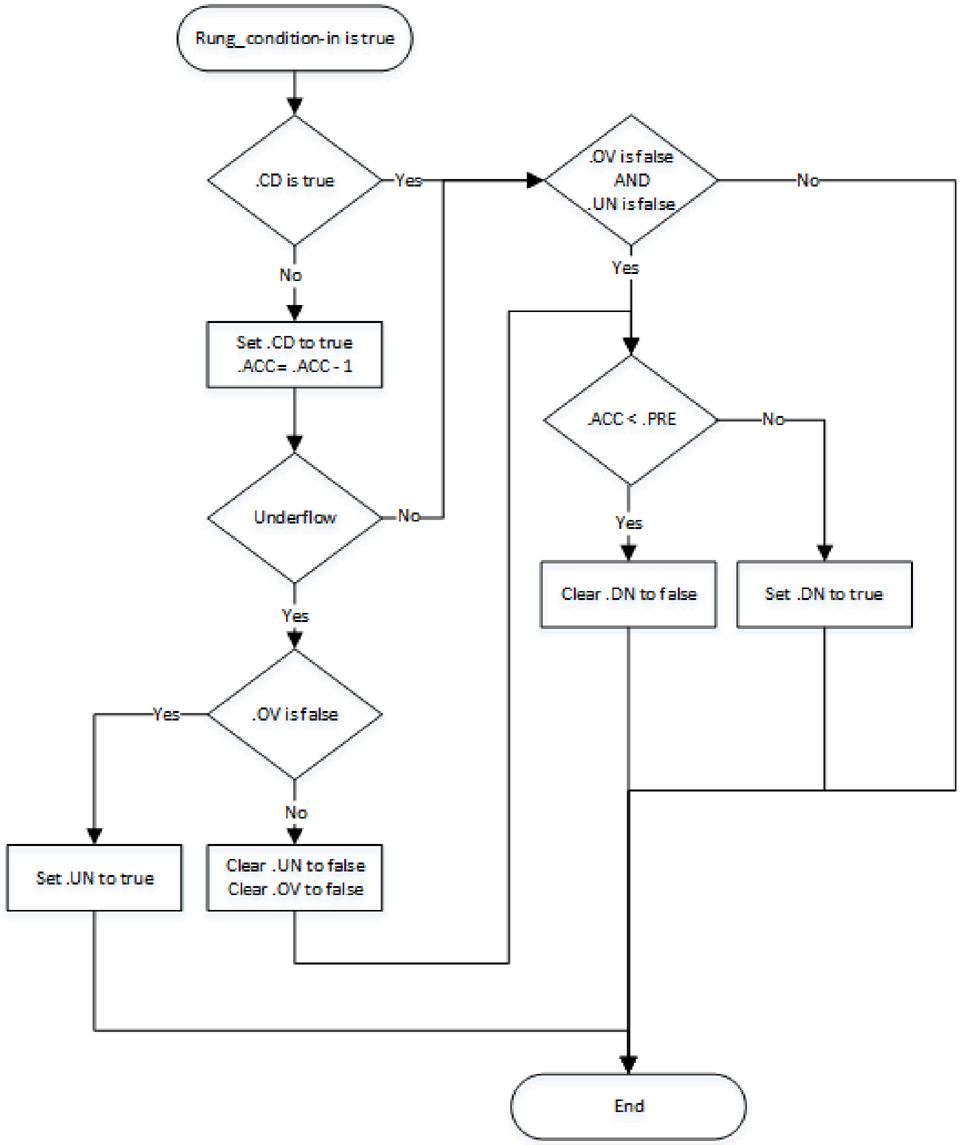
Condition/State	Action Taken
Prescan	The .CD bit is set to true to prevent invalid decrements during the first program scan.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in See CTD Flow Chart (False)
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in See CTD Flow Chart (True)

Condition/State	Action Taken
Postscan	N/A

CTD Flow Chart (False)

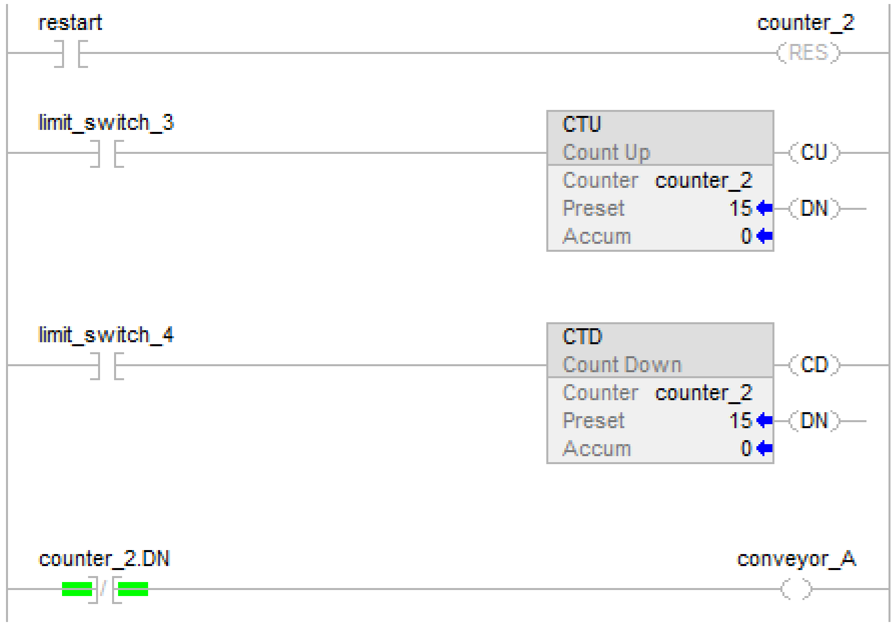


CTD Flow Chart (True)



Example

Ladder Diagram



A conveyor brings parts into a buffer zone. Each time a part enters, limit_switch_3 is enabled and counter_2 increments by 1. Each time a part leaves, limit_switch_4 is enabled and counter_2 decrements by 1. If there are 100 parts in the buffer zone (counter_2.dn is true), conveyor_A turns on and stops the conveyor from bringing in any more parts until the buffer has room for more parts.

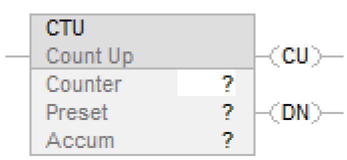
Count Up (CTU)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The CTU instruction counts upward each time the rung-condition-in transitions from false to true.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Counter	COUNTER	tag	Counter structure
Preset	DINT	immediate	Value of Counter.PRE.
Accum	DINT	immediate	Value of Counter.ACC.

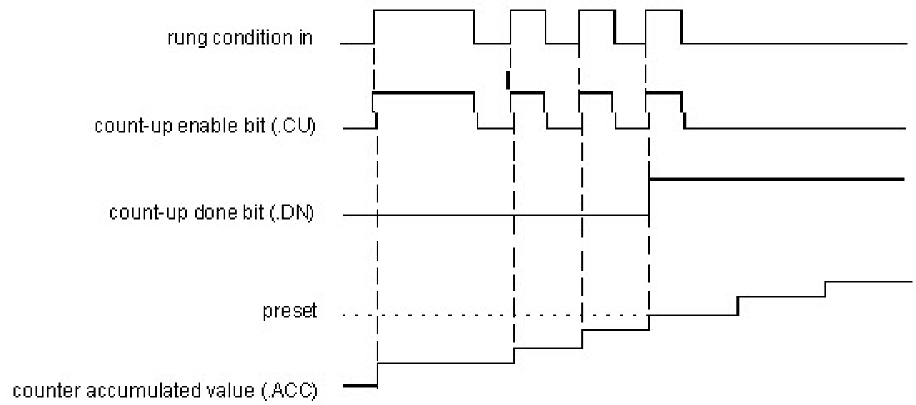
Length (corresponding to .LEN in the control tag) is a pseudo-operand. For details, see Pseudo-operand initialization on page 856.

COUNTER Structure

Mnemonic	Data Type	Description
.CU	BOOL	The count up enable contains rung-condition-in when the instruction was last executed.
.DN	BOOL	The done bit when set indicates the counting operation is complete.
.OV	BOOL	The overflow bit when set indicates the counter incremented past the upper limit of 2,147,483,647.
.UN	BOOL	The underflow when set indicates the counter decremented past the lower limit of -2,147,483,648.
.PRE	DINT	The preset value specifies the value which the accumulated value must reach before the instruction indicates it is done.
.ACC	DINT	The accumulated value specifies the number of transitions the instruction has counted.

Description

When rung-condition-in is set to true and .CU is false, ACC will be incremented by one. When rung-condition-in is false, .CU will be cleared to false.



Affects Math Status Flags

No

Major/Minor Faults

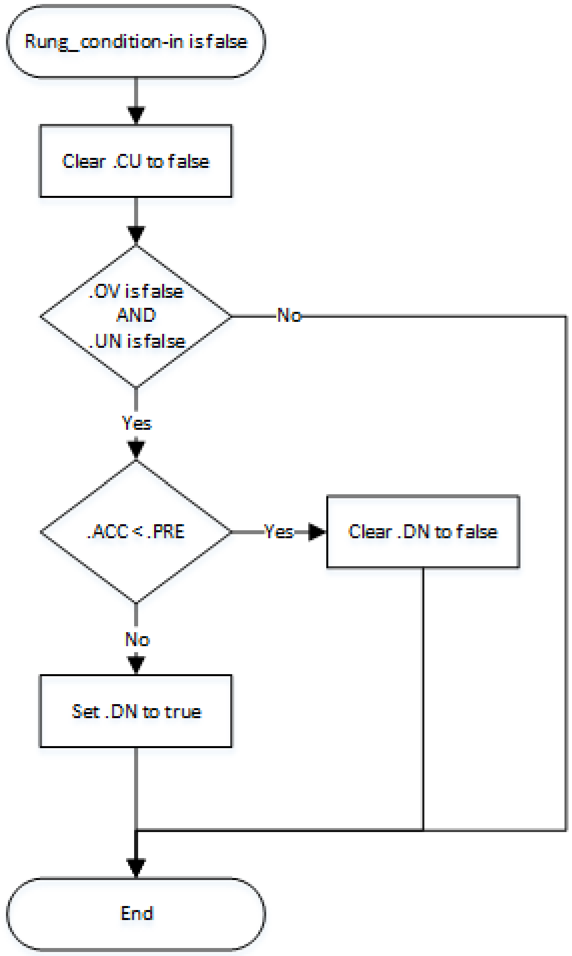
None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

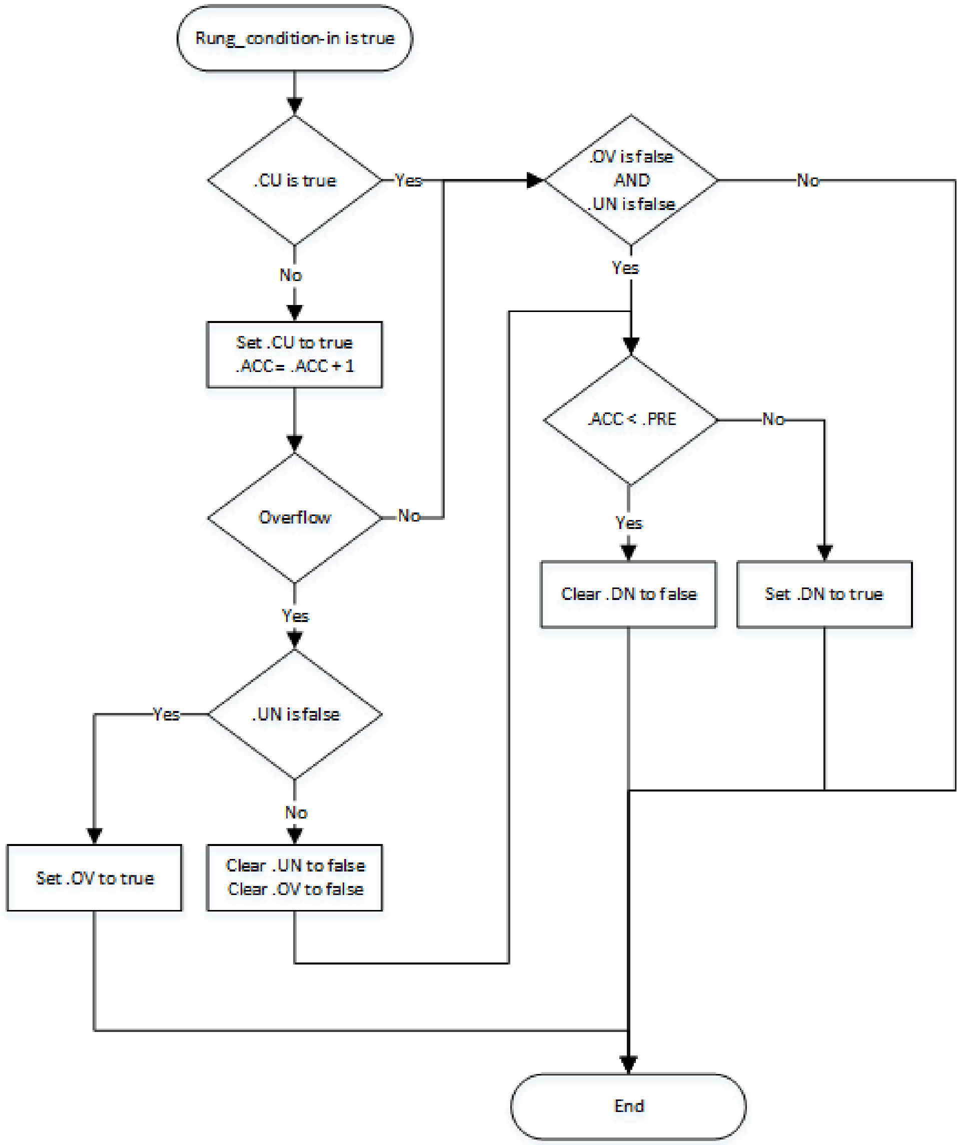
Ladder Diagram

Condition/State	Action Taken
Prescan	The .CU bit is set to true to prevent invalid increments during the first program scan.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in See CTU Flow Chart (False)
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in See CTU Flow Chart (True)
Postscan	N/A

CTU Flow Chart (False)

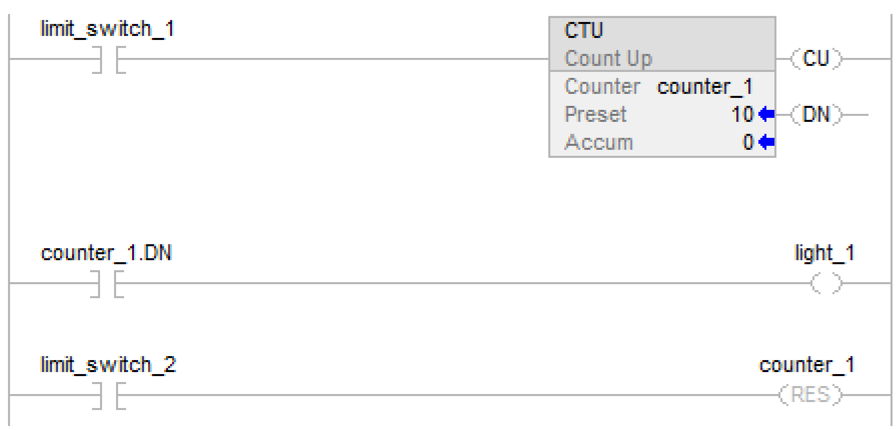


CTU Flow Chart (True)



Example

Ladder Diagram



After limit_switch_1 goes from disabled to enabled 10 times, the .DN bit is set to true and light_1 turns on. If limit_switch_1 continues to go from disabled to enabled, counter_1 continues to increment its count and the .DN bit remains set. When limit_switch_2 is enabled, the RES instruction resets counter_1 (clears the status bits and the .ACC value) and light_1 turns off.

Count Up/Down (CTUD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

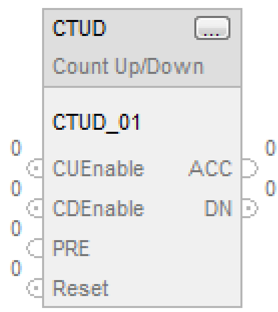
The CTUD instruction counts up by one when CUEnable transitions from clear to set. The instruction counts down by one when CDEnable transitions from clear to set.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

CTUD(CTUD_tag)

Operands

Structured Text

Variable	Type	Format	Description
CTUD tag	FBD_COUNTER	Structure	CTUD structure

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
CTUD tag	FBD_COUNTER	Structure	CTUD structure

FBD_COUNTER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
CUEnable	BOOL	Enable up count. When input toggles from clear to set, accumulator counts up by one. Default is cleared
CDEnable	BOOL	Enable down count. When input toggles from clear to set, accumulator counts down by one. Default is cleared
PRE	DINT	Counter preset value. This is the value the accumulated value must reach before DN is set. Valid = any integer Default is 0
Reset	BOOL	Request to reset the timer. When set, the counter resets. Default is cleared

Output Parameter	Data Type	Description
EnableOut	BOOL	The instruction produced a valid result.
ACC	DINT	Accumulated value.
CU	BOOL	Count up enabled.
CD	BOOL	Count down enabled.
DN	BOOL	Counting done. Set when accumulated value is greater than or equal to preset.
OV	BOOL	Counter overflow. Indicates the counter exceeded the upper limit of 2,147,483,647. The counter then rolls over to -2,147,483,648 and begins counting down again.
UN	BOOL	Counter underflow. Indicates the counter exceeded the lower limit of -2,147,483,648. The counter then rolls

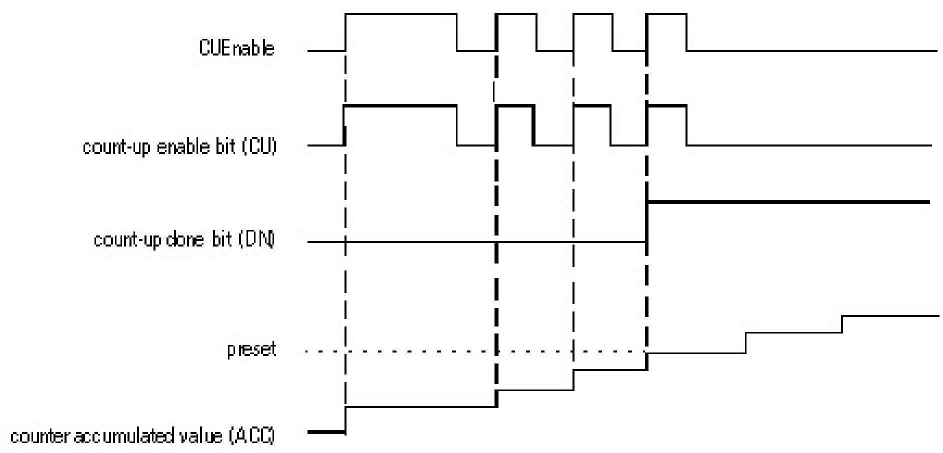
Output Parameter	Data Type	Description
		over to 2,147,483,647 and begins counting down again.

Description

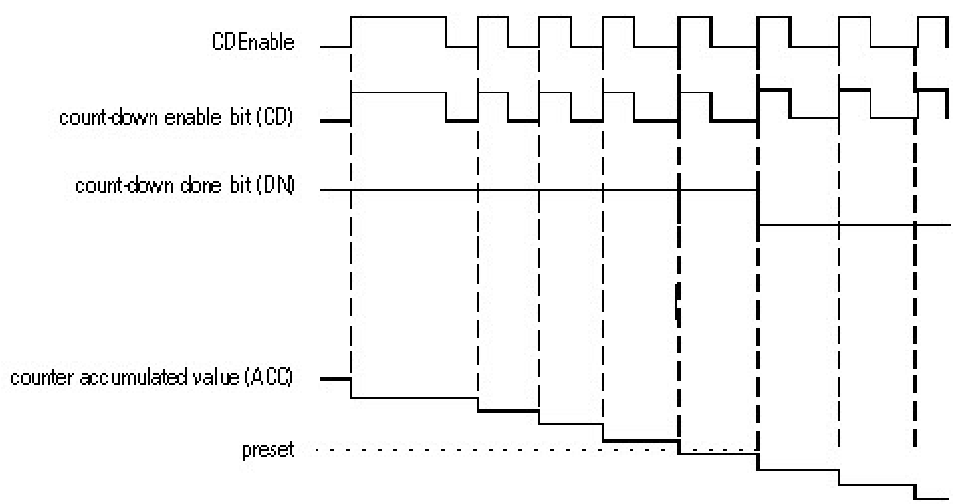
When true and CUEnable is true, the CTUD instructions increments the counter by one. When true and CDEnable is true, the CTUD instruction decrements the counter by one.

Both the CUEnable and CDEnable input parameters can be toggled during the same scan. The instruction executes the count up prior to the count down.

Count Up



Count Down



When disabled, the CTUD instruction retains its accumulated value. Set the Reset input parameter to reset the instruction.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes for General Instructions* on page 849 for operand-related faults.

Execution

Function Block

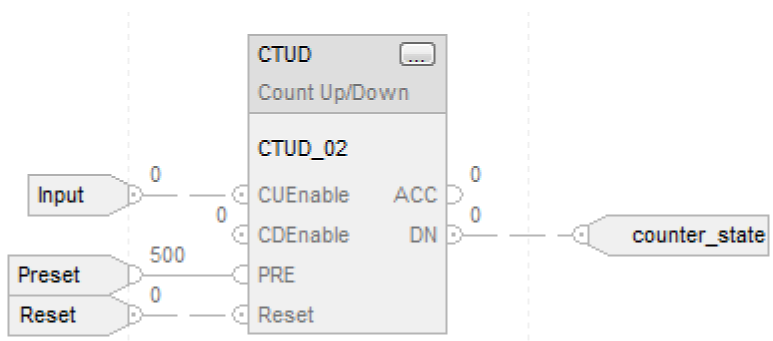
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false. Initialize data to require a "zero to one" transition of CuEnable or CdEnable to effect ACC.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	Initialize data to require a "zero to one" transition of CuEnable or CdEnable to effect ACC.
Instruction first scan	Initialize data to require a "zero to one" transition of CuEnable or CdEnable to effect ACC.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Function Block



Structured Text

CTUD_01.PRE := 500;

```
CTUD_01.Reset := Reset;

CTUD_01.CUEnable := Input;

CTUD(CTUD_01);

counter_state := CTUD_01.DN;
```

Reset (RES)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The RES instruction resets a TIMER, COUNTER, or CONTROL structure.

Available Languages

Ladder Diagram

-(RES)-

Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Structure	TIMER CONTROL COUNTER	Tag	Structure to reset

Description

When true, the RES instruction clears these elements:

When using a RES instruction for a:	The instruction clears:

TIMER	.ACC value to 0 control status bits to false
COUNTER	.ACC value to 0 control status bits to false
CONTROL	.POS value to 0 control status bits to false

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

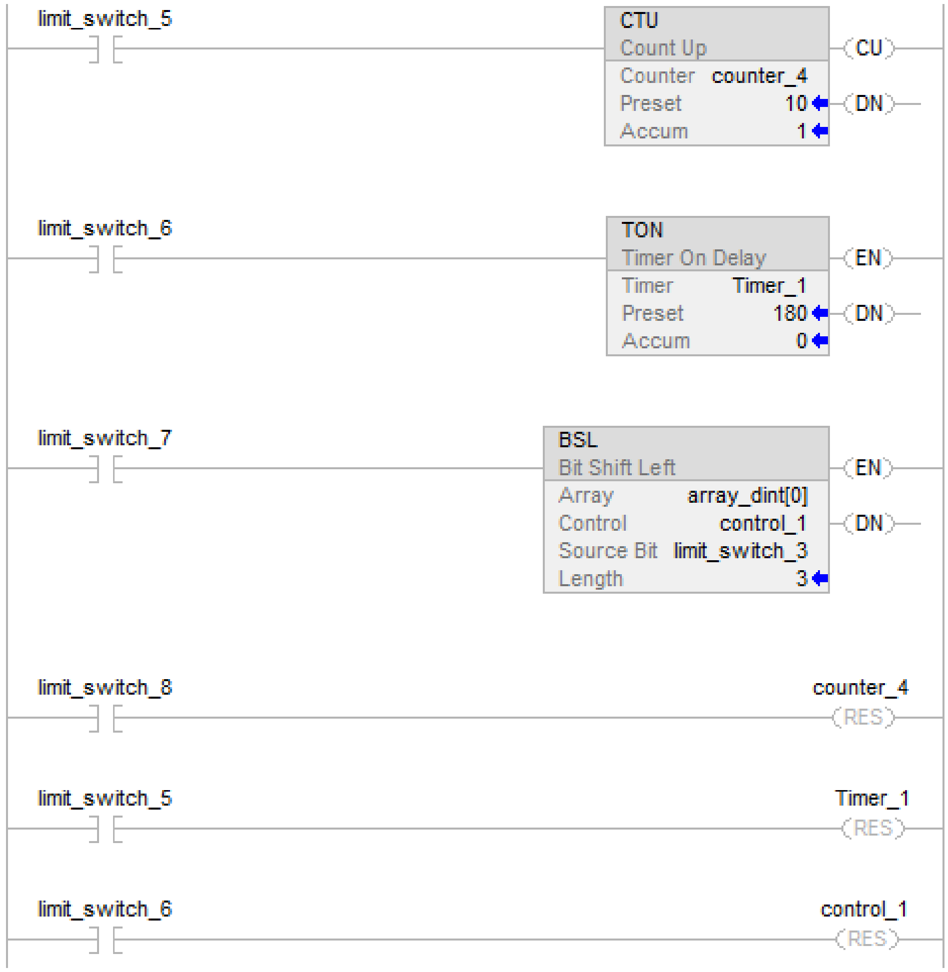
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Reset the specified structure.
Postscan	N/A

Example

Ladder Diagram



Reset Example

In the preceding example:

when limit_switch_8 is enabled, reset counter_4

when limit_switch_5 is enabled, reset Timer_1

when limit_switch_6 is enabled, reset control_1

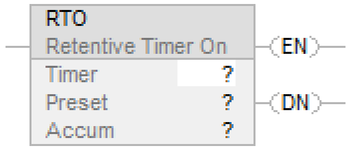
Retentive Timer On (RTO)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The RTO instruction is a retentive timer that accumulates time when the instruction is enabled.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Timer	TIMER	tag	Timer structure
Preset	DINT	immediate	Value of Timer.PRE.
Accum	DINT	immediate	Value of Timer.ACC.

Preset and Accum (corresponding to .PRE and .ACC in the timer tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

TIMER Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit contains rung-condition-in when the instruction was last executed.
.TT	BOOL	The timing bit when set indicates the timing operation is in process.
.DN	BOOL	The done bit when set indicates the timing operation is complete (or paused).
.PRE	DINT	The preset value specifies the value (1 millisecond units) which the accumulated value must reach before the instruction indicates it is done.

.ACC	DINT	The accumulated value specifies the number of milliseconds that have elapsed since the RTO instruction was enabled.

Description

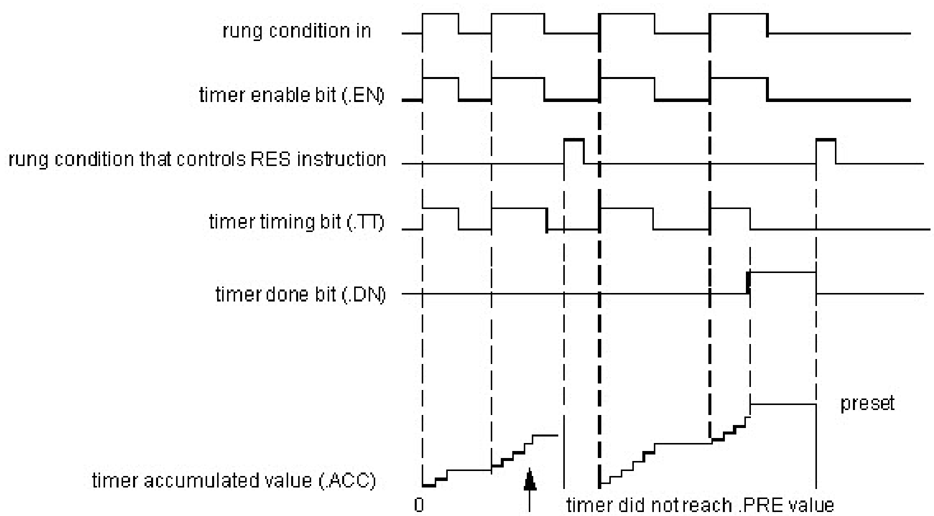
The RTO instruction accumulates time until:

- The timer is disabled.
- The timer completes.

The time base is always 1 millisecond. For example, for a 2 second timer, enter 2000 for the .PRE value.

The timer will set the .DN bit to true when the timer completes.

When enabled, timing can be paused by setting the .DN bit to true and resumed by clearing the .DN bit to false.



How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

ACC = ACC + (current_time - last_time_scanned)

After it updates the ACC, the timer sets last_time_scanned = current_time. This gets the timer ready for the next scan.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.PRE < 0	4	34
.ACC < 0	4	34

See Index through arrays on page 863 for array-indexing faults.

Execution

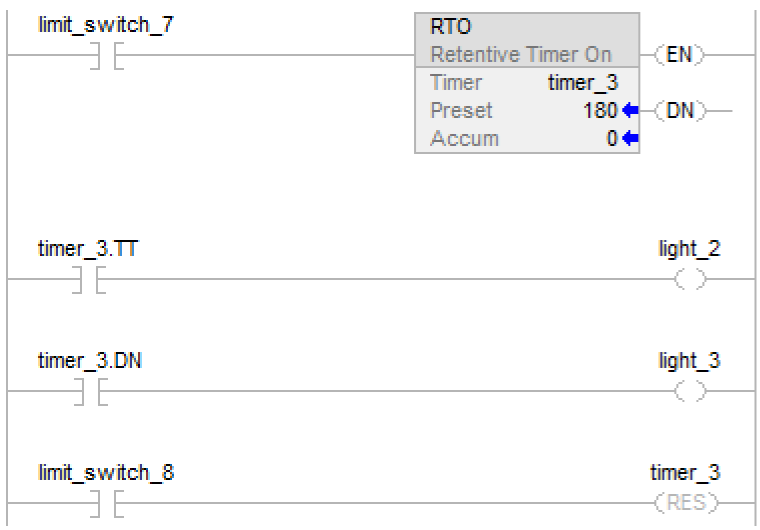
Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN bit is cleared to false. The .TT bit is cleared to false.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in The .EN bit is cleared to false. The .TT bit is cleared to false.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in See RTO Flow Chart (True).
Postscan	N/A

RTO Flow Chart (True)

Example

Ladder Diagram



When limit_switch_7 is set, light_2 is on for 180 milliseconds (timer_3 is timing). When timer_3.acc reaches 180, light_2 goes off and light_3 goes on. Light_3 remains on until timer_3 is reset. If limit_switch_7 is cleared while timer_3 is timing, light_2 goes off. When limit_switch_8 is set, the RES instruction resets timer_3 (clears status bits and .ACC value).

Retentive Timer On with Reset (RTOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

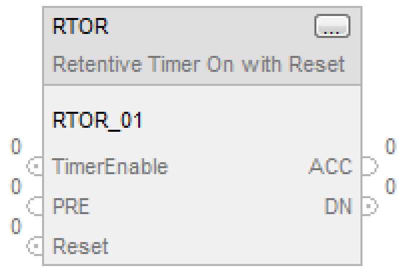
The RTOR instruction is a retentive timer that accumulates time when TimerEnable is set.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

RTOR(RTOR_tag)

Operands

Structured Text

Variable	Type	Format	Description
RTOR tag	FBD_TIMER	Structure	RTOR structure

See *Structured Text Syntax* on page 879 for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
RTOR tag	FBD_TIMER	Structure	RTOR structure

FBD_TIMER Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
TimerEnable	BOOL	If set, this enables the timer to run and accumulate time. Default is cleared.
PRE	DINT	Timer preset value. This is the value in 1 msec units that ACC must reach before timing is finished. If invalid, the

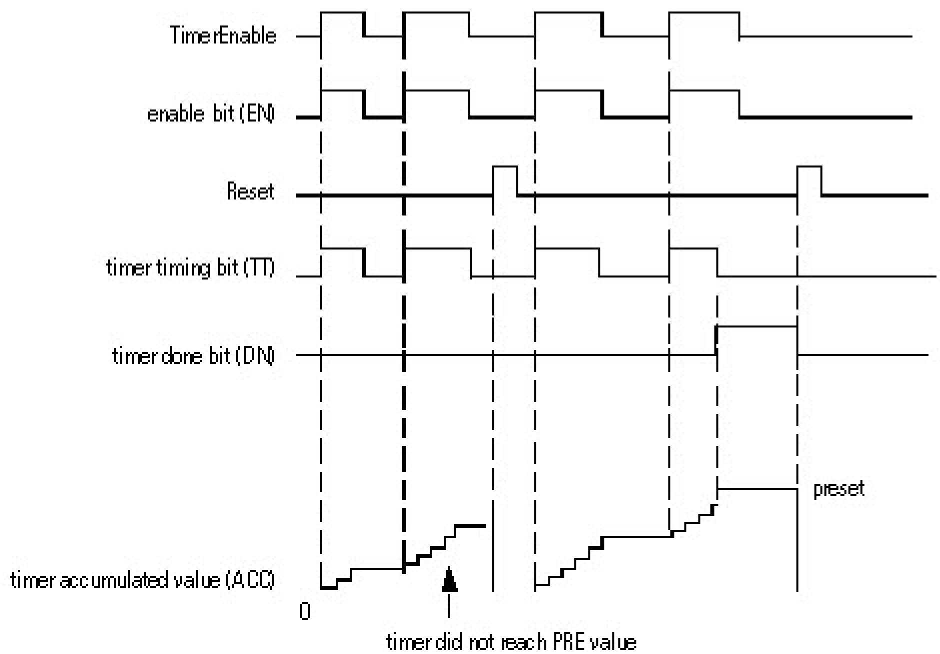
Input Parameter	Data Type	Description
		instruction sets the appropriate bit in Status and the timer does not execute. Valid = 0 to maximum positive integer
Reset	BOOL	Request to reset the timer. When set, the timer resets. When the Reset input parameter is set, the instruction clears EN, TT and DN and sets ACC = 0.

Output Parameter	Data Type	Description
EnableOut	BOOL	The instruction produced a valid result.
ACC	DINT	Accumulated time in milliseconds. This value is retained even while the TimerEnable input is cleared.
EN	BOOL	Timer enabled output. Indicates the timer instruction is enabled.
TT	BOOL	Timer timing output. When set, a timing operation is in progress.
DN	BOOL	Timing done output. Indicates when accumulated time is greater than or equal to preset.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
PresetInv (Status.1)	BOOL	The preset value is invalid.

Description

The RTOR instruction accumulates time until it is false. When the RTOR instruction is false, it retains its ACC value. You must clear the .ACC value using the Reset input.

The time base is always 1 msec. For example, for a 2-second timer, enter 2000 for the PRE value.



Set the Reset input parameter to reset the instruction. If TimerEnable is set when Reset is set, the RTOR instruction begins timing again when Reset is cleared.

How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

- ACC = ACC + (current_time - last_time_scanned)
- After it updates the ACC, the timer sets last_time_scanned= current_time. This gets the timer ready for the next scan.

IMPORTANT: Be sure to scan the timer at least every 69 minutes while it runs. Otherwise, the ACC value won't be correct.

The last_time_scanned value has a range of up to 69 minutes. The timer's calculation rolls over if you don't scan the timer within 69 minutes. The ACC value won't be correct if this happens.

While a timer runs, scan it within 69 minutes if you put it in a:

- Subroutine
- Section of code that is between JMP and LBL instructions
- Sequential function chart (SFC)
- Event or periodic task
- State routine of a phase

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes for General Instructions* on page 849 for operand-related faults.

Execution

Function Block

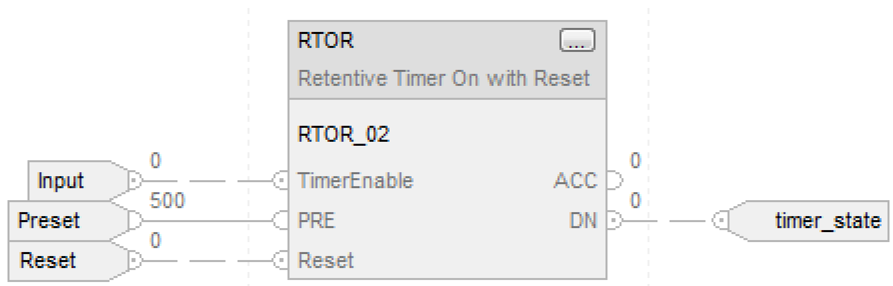
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes. When the Reset input parameter is set, the instruction clears EN, TT and DN and sets ACC = 0.
Instruction first run	EN, TT and DN are cleared to false. The instruction executes.
Instruction first scan	N/A
Postscan	EnableIn and EnableOut are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Function Block



Structured Text

RTOR_01.PRE := 500;

RTOR_01.Reset := Reset;

RTOR_01.TimerEnable := Input;

RTOR(RTOR_01);

timer_state := RTOR_01.DN;

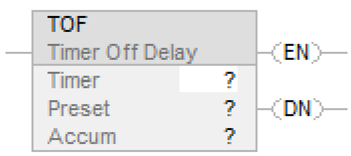
Timer Off Delay (TOF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The TOF instruction is a non-retentive timer that accumulates time when the instruction is enabled (rung-condition-in is false).

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Timer	TIMER	tag	Timer structure
Preset	DINT	immediate	Value of Timer.PRE.
Accum	DINT	immediate	Value of Timer.ACC.

Preset and Accum (corresponding to .PRE and .ACC in the timer tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

TIMER Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit contains rung-condition-in when the instruction was last executed.
.TT	BOOL	The timing bit when set indicates the timing operation is in process.
.DN	BOOL	The done bit when cleared indicates the timing operation is complete (or paused).
.PRE	DINT	The preset value specifies the value (1 millisecond units) which the accumulated value must reach before the instruction indicates it is done.
.ACC	DINT	The accumulated value specifies the number of milliseconds that have elapsed since the TOF instruction was enabled.

Description

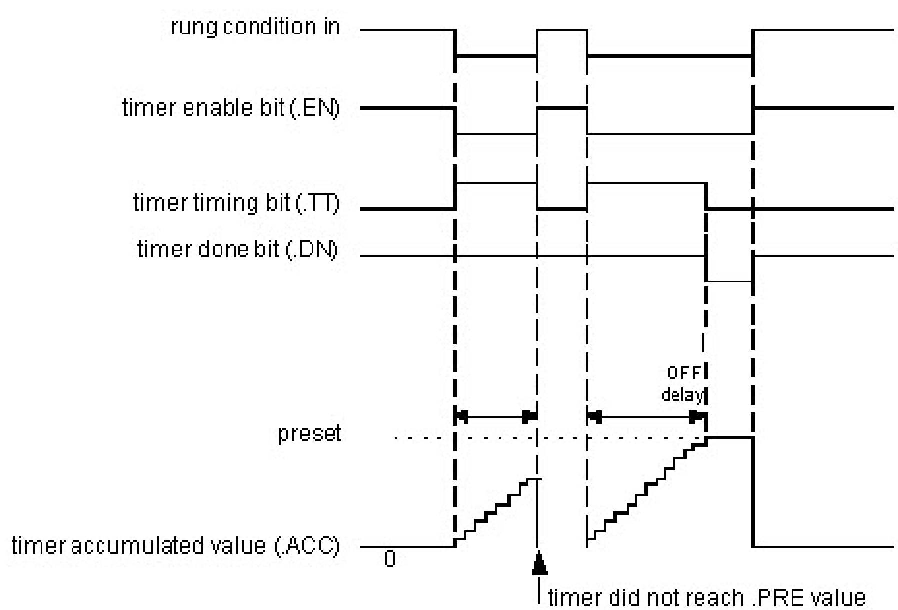
The TOF instruction accumulates time until:

- The timer is disabled
- The timer completes

The time base is always 1 millisecond. For example, for a 2 second timer, enter 2000 for the .PRE value.

The timer will clear the .DN bit to false when the timer completes.

When enabled, timing can be paused by clearing the .DN bit to false and resumed by setting the .DN bit to true.



How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

ACC = ACC + (current_time - last_time_scanned)

After it updates the ACC, the timer sets last_time_scanned = current_time. This gets the timer ready for the next scan.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.PRE < 0	4	34
.ACC < 0	4	34

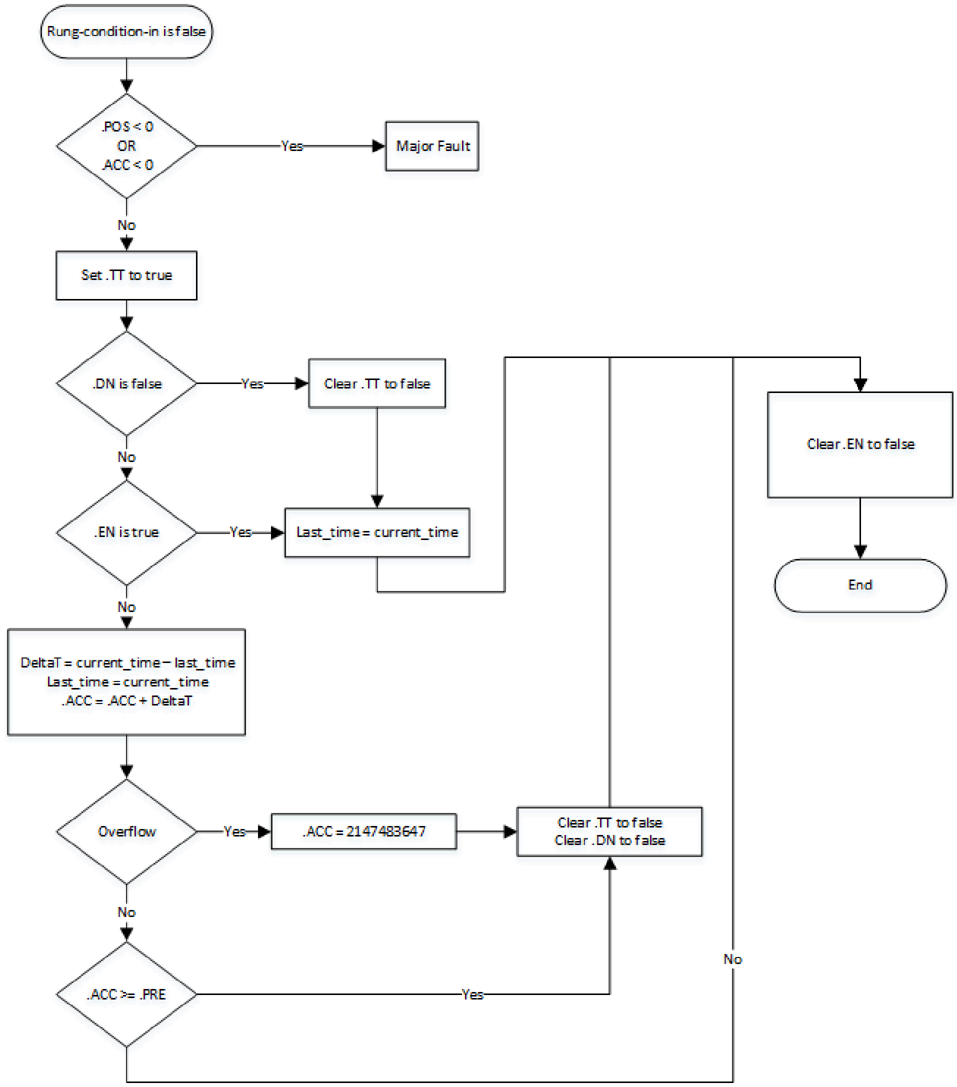
See *Index through arrays* on page 863 for array-indexing faults.

Execution

Ladder Diagram

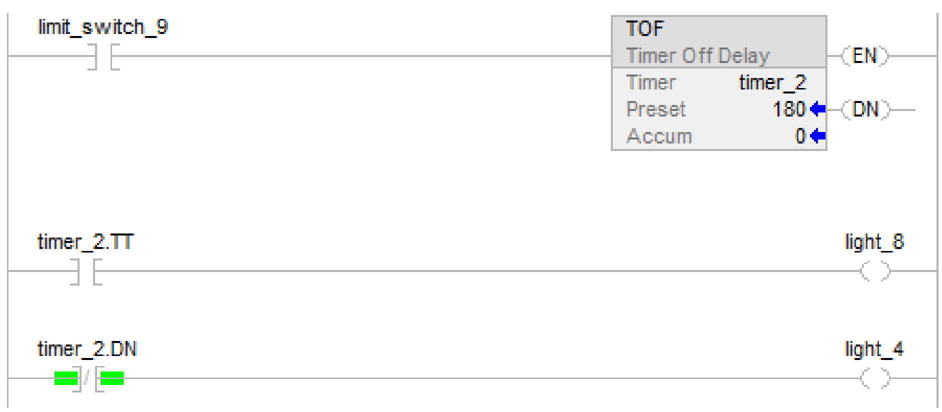
Condition/State	Action Taken
Prescan	The .EN bit is cleared to false. The .TT bit is cleared to false. The .DN bit is cleared to false. The .ACC value is set to equal the .PRE value.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in See TOF Flow Chart (False).
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in The .EN bit is set to true. The .TT bit is cleared to false. The .DN bit is set to true. The .ACC value is cleared to zero.
Postscan	The .EN bit is cleared to false. The .TT bit is cleared to false. The .DN bit is cleared to false. The .ACC value is set to equal the .PRE value.

TOF Flow Chart (False)



Example

Ladder Diagram



When limit_switch_9 is cleared, light_8 is on for 180 milliseconds (timer_2 is timing). When timer_2.acc reaches 180, light_8 goes off and light_4 goes on. Light_4 remains on until the TOF instruction is enabled. If limit_switch_9 is true while timer_2 is timing, light_8 goes off.

Timer Off Delay with Reset (TOFR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

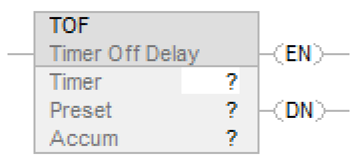
The TOFR instruction is a non-retentive timer that accumulates time when TimerEnable is cleared.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

TOFR(TOFR_tag)

Operands

Structured Text

Variable	Type	Format	Description
TOFR tag	FBD_TIMER	Structure	TOFR structure

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
TOFR tag	FBD_TIMER	Structure	TOFR structure

FBD_TIMER Structure

Input Parameters	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes.

Input Parameters	Data Type	Description
		Default is set.
TimerEnable	BOOL	If cleared, this enables the timer to run and accumulate time. Default is cleared.
PRE	DINT	Timer preset value. This is the value in 1 msec units that ACC must reach before timing is finished. If invalid, the instruction sets the appropriate bit in Status and the timer does not execute. Valid = 0 to maximum positive integer
Reset	BOOL	Request to reset the timer. When set, the timer resets. Default is cleared. When the Reset input parameter is set, the instruction clears EN, TT and DN and sets ACC = PRE. Note that this is different than using a RES instruction on a TOF instruction.

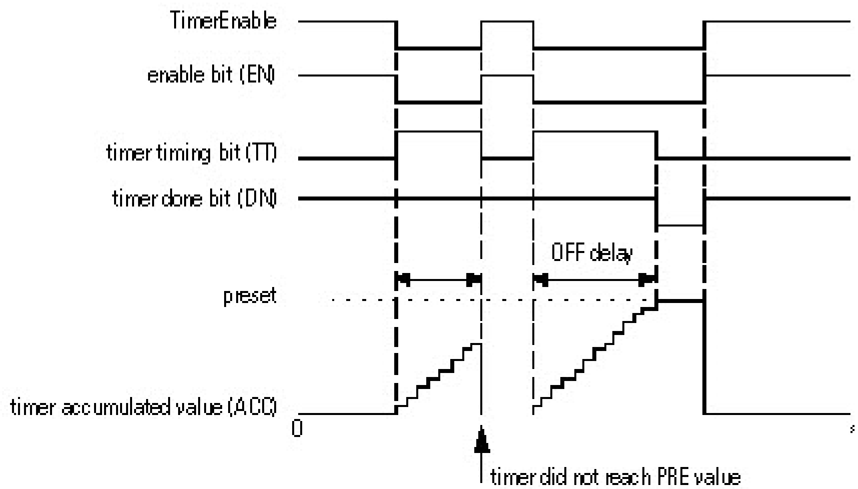
Output Parameters	Data Type	Description
EnableOut	BOOL	The instruction produced a valid result.
ACC	BOOL	Accumulated time in milliseconds.
EN	BOOL	Timer enabled output. Indicates the timer instruction is enabled.
TT	BOOL	Timer timing output. When set, a timing operation is in progress.
DN	BOOL	Timing done output. Indicates when accumulated time is greater than or equal to preset.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
PresetInv (Status.1)	BOOL	The preset value is invalid.

Description

When true, the TOFR instruction accumulates time until the:

- TOFR instruction is disabled
    - ACC is greater than or equal to PRE

The time base is always 1 msec. For example, for a 2-second timer, enter 2000 for the PRE value.



Set the Reset input parameter to reset the instruction. If TimerEnable is false when Reset is true, the TOFR instruction does not begin timing again when Reset is false.

How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

ACC = ACC + (current_time - last_time_scanned)

After it updates the ACC, the timer sets last_time_scanned= current_time. This gets the timer ready for the next scan.

IMPORTANT: Be sure to scan the timer at least every 69 minutes while it runs. Otherwise, the ACC value won't be correct.

The last_time_scanned value has a range of up to 69 minutes. The timer's calculation rolls over if you don't scan the timer within 69 minutes. The ACC value won't be correct if this happens.

While a timer runs, scan it within 69 minutes if you put it in a:

- Subroutine
- Section of code that is between JMP and LBL instructions
- Sequential function chart (SFC)
- Event or periodic task
- State routine of a phase

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes for General Instructions* on page 849 for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag. EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag. EnableIn is true	EnableIn and EnableOut bits are set to true. The main algorithm of the instruction will be executed and outputs will be updated.
Instruction first run	N/A
Instruction first scan	EN, TT and DN are cleared ACC value is not modified.
Postscan	EnableIn and EnableOut bits are cleared to false.

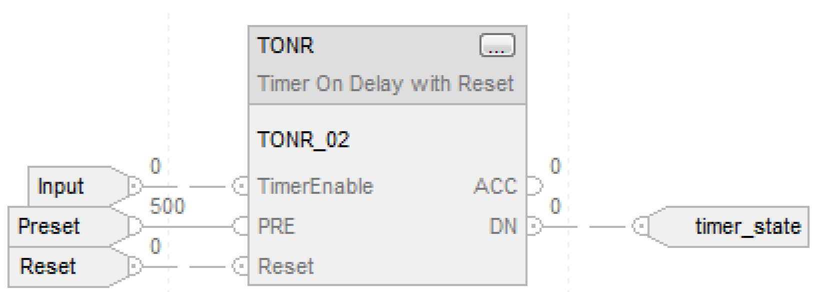
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Each scan after limit_switch1 is cleared, the TOFR instruction increments the ACC value by elapsed time until the ACC value reaches the PRE value. When ACC ≥ PRE, the DN parameter is cleared, and timer_state2 is set.

Function Block



Structured Text

TOFR_01.PRE := 500;

TOFR_01.Reset := Reset;

TOFR_01.TimerEnable := Input;

```
TOFR(TOFR_01);

timer_state := TOFR_01.DN;
```

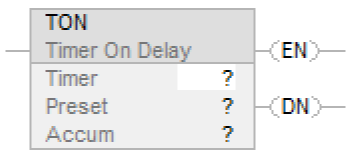
Timer On Delay (TON)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The TON instruction is a non-retentive timer that accumulates time when the instruction is enabled.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type	Format	Description
Timer	TIMER	tag	Timer structure
Preset	DINT	immediate	Value of Timer.PRE.
Accum	DINT	immediate	Value of Timer.ACC.

Preset and Accum (corresponding to .PRE and .ACC in the timer tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

TIMER Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit contains rung-condition-in when the instruction was last executed.
.TT	BOOL	The timing bit when set indicates the timing operation is in process.
.DN	BOOL	The done bit when set indicates the timing operation is complete (or paused).
.PRE	DINT	The preset value specifies the value (1 millisecond units) which the accumulated value must reach before the instruction indicates it is done.
.ACC	DINT	The accumulated value specifies the number of milliseconds that have elapsed since the TON instruction was enabled.

Description

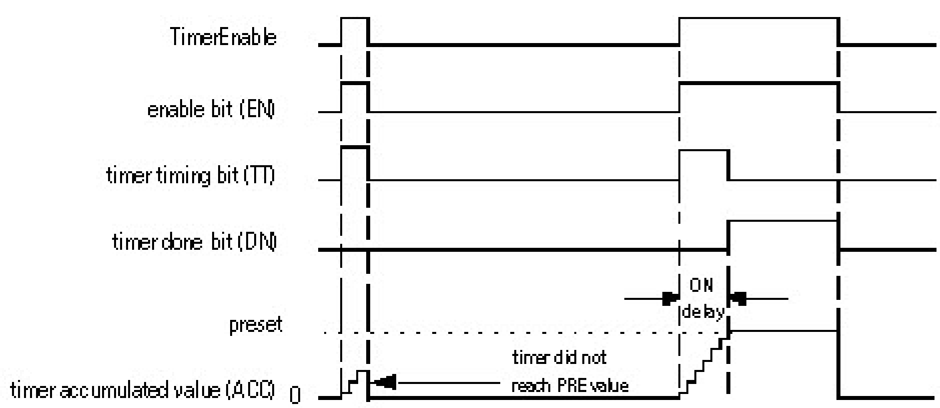
The TON instruction accumulates time from the time it is enabled until:

- The timer is disabled
- The timer completes

The time base is always 1 millisecond. For example, for a 2 second timer, enter 2000 for the .PRE value.

The timer will set the .DN bit to true when the timer completes.

When enabled, timing can be paused by setting the .DN bit to true and resumed by clearing the .DN bit to false.



How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

ACC = ACC + (current_time - last_time_scanned)

After it updates the ACC, the timer sets last_time_scanned = current_time. This gets the timer ready for the next scan.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.PRE < 0	4	34
.ACC < 0	4	34

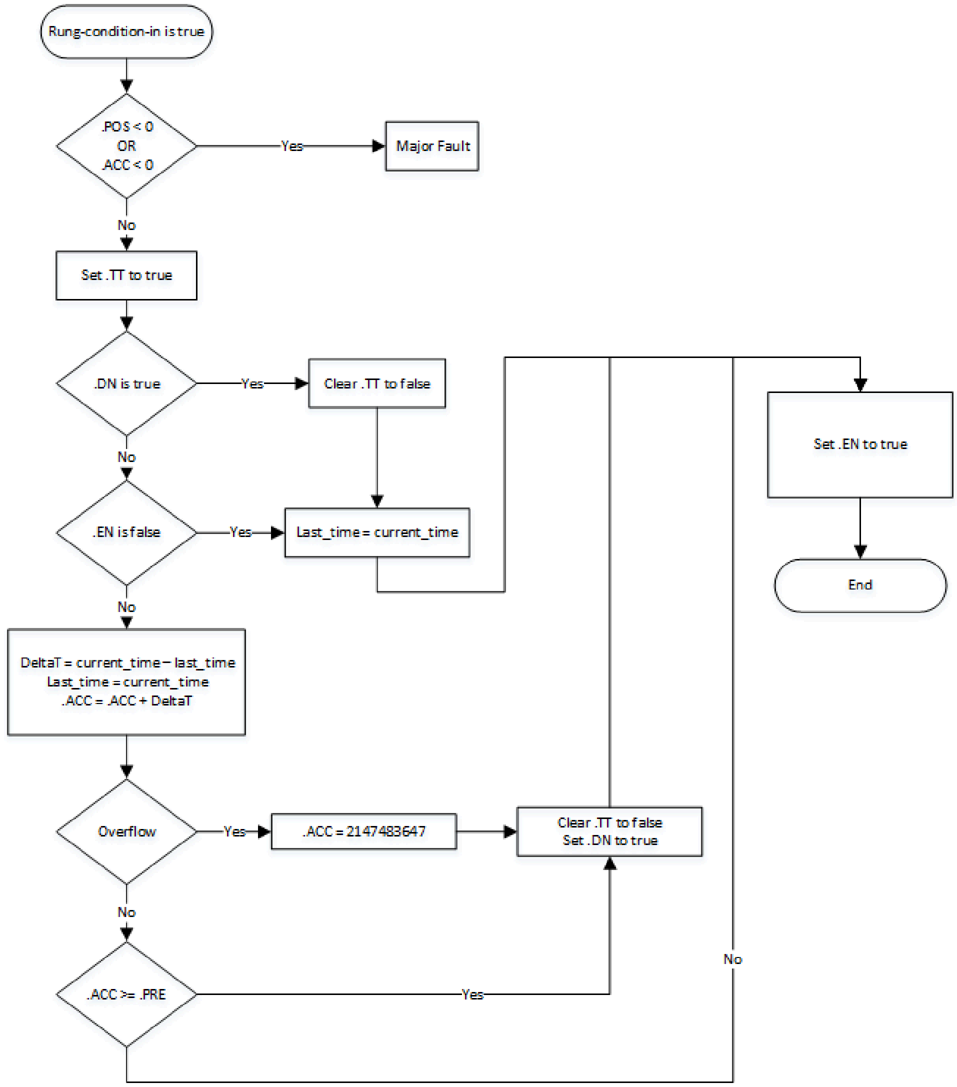
See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

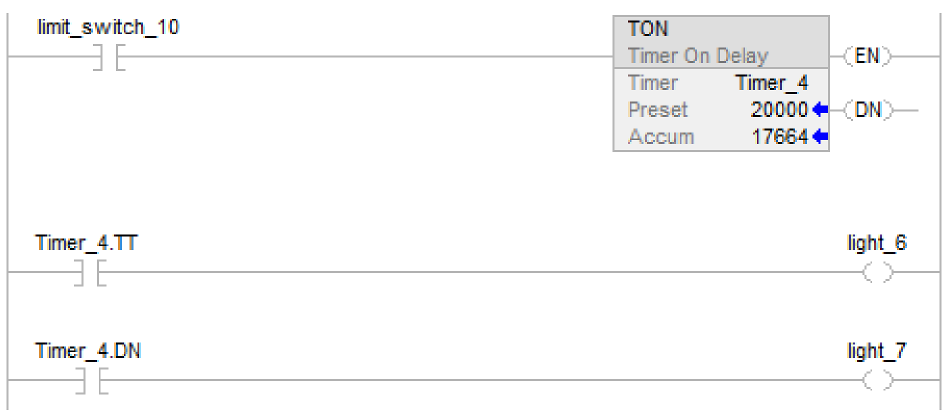
Condition/State	Action Taken
Prescan	The .EN bit is cleared to false. The .TT bit is cleared to false. The .DN bit is cleared to false. The .ACC value is cleared to zero.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in The .EN bit is cleared to false. The .TT bit is cleared to false. The .DN bit is cleared to false. The .ACC value is cleared to zero.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in See TON Flow Chart (True)
Postscan	The .EN bit is cleared to false. The .TT bit is cleared to false. The .DN bit is cleared to false. The .ACC value is cleared to zero.

TON Flow Chart (True)



Example

Ladder Diagram



When limit_switch_10 is set to true, light_6 is on for 20000 milliseconds (Timer_4 is timing). When Timer_4.acc reaches 20000, light_6 goes off and light_7 goes on. If limit_switch_10 is cleared to false while Timer_4 is timing, light_6 goes off. When limit_switch_10 is cleared to false, Timer_4 status bits and .ACC value are reset.

Timer On Delay with Reset (TONR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

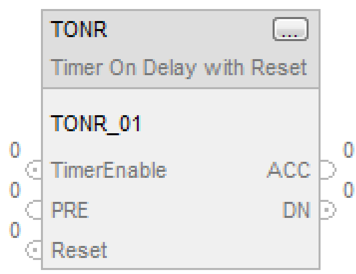
The TONR instruction is a non-retentive timer that accumulates time when TimerEnable is set.

Available Languages

Ladder Diagram

This instruction is not available in ladder diagram.

Function Block



Structured Text

TONR(TONR_tag);

Operands

Structured Text

Operand	Type	Format	Description
TONR tag	FBD_TIMER	Structure	TONR structure

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
TONR tag	FBD_TIMER	Structure	TONR structure

FBD_TIMER Structure

Input Parameter	Data Type	Description

EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
TimerEnable	BOOL	If set, this enables the timer to run and accumulate time. Default is cleared.
PRE	DINT	Timer preset value. This is the value in 1 msec units that ACC must reach before timing is finished. If invalid, the instruction sets the appropriate bit in Status and the timer does not execute. Valid = 0 to maximum positive integer
Reset	BOOL	Request to reset the timer. When set, the timer resets. Default is cleared. When the Reset input parameter is set, the instruction clears EN, TT and DN and sets ACC = 0.

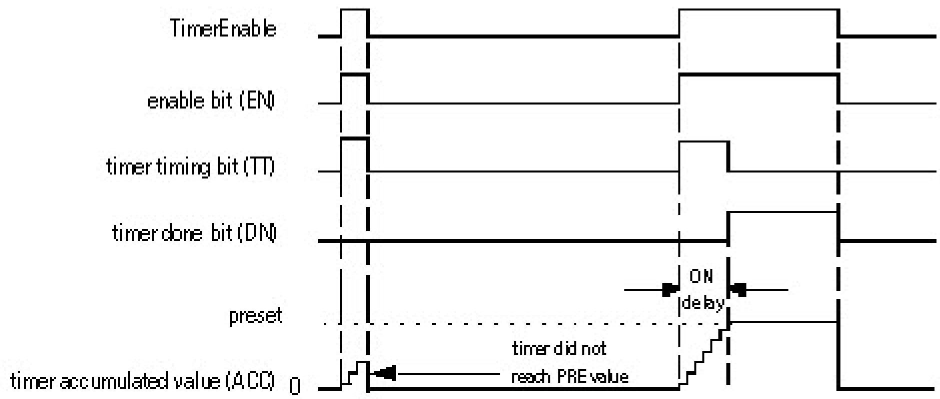
Output Parameter	Data Type	Description
EnableOut	BOOL	The instruction produced a valid result.
ACC	BOOL	Accumulated time in milliseconds.
ENF	BOOL	Timer enabled output. Indicates the timer instruction is enabled.
TT	BOOL	Timer timing output. When set, a timing operation is in progress.
DN	BOOL	Timing done output. Indicates when accumulated time is greater than or equal to preset.
Status	DINT	Status of the function block.
InstructFault (Status.0)	BOOL	The instruction detected one of the following execution errors. This is not a minor or major controller error. Check the remaining status bits to determine what occurred.
PresetInv (Status.1)	BOOL	The preset value is invalid.

Description

When true, the TONR instruction accumulates time until the:

- TONR instruction is disabled
- ACC ≥ PRE

The time base is always 1 msec. For example, for a 2-second timer, enter 2000 for the PRE value.



Set the Reset input parameter to reset the instruction. If TimerEnable is set when Reset is true, the TONR instruction begins timing again when Reset is false.

How a Timer Runs

A timer runs by subtracting the time of its last scan from the current time:

- ACC = ACC + (current_time - last_time_scanned)

After it updates the ACC, the timer sets last_time_scanned= current_time. This gets the timer ready for the next scan.

IMPORTANT: Be sure to scan the timer at least every 69 minutes while it runs. Otherwise, the ACC value will not be correct.

The last_time_scanned value has a range of up to 69 minutes. The timer's calculation rolls over if you don't scan the timer within 69 minutes. The ACC value won't be correct if this happens.

While a timer runs, scan it within 69 minutes if you put it in a:

- Subroutine
- Section of code that is between JMP and LBL instructions
- Sequential function chart (SFC)
- Event or periodic task
- State routine of a phase

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes for General Instructions* on page 849 for operand-related faults.

Execution

Function Block

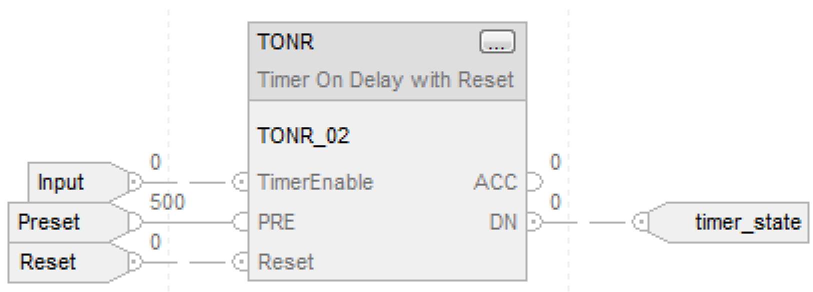
Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The main algorithm of the instruction is executed and outputs are updated.
Instruction first run	N/A
Instruction first scan	EN, TT and DN are cleared ACC value is set to 0.
Postscan	EnableIn and EnableOut bits are cleared to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

Function Block



Structured Text

```
TONR_01.PRE := 500;

TONR_01.Reset := Reset;

TONR_01.TimerEnable := Input;

TONR(TONR_01);

timer_state := TONR_01.DN;
```

Input/Output Instructions

The input/output instructions read or write data to or from the controller or a block of data to or from another module on another network.

Available Instructions

Ladder Diagram and Structured Text

MSG on page 127	GSV on page 162	SSV on page 162	IOT on page 189

Function Block

Not available

If you want to:	Use this instruction:
Send data to or from another module	MSG
Get controller status information	GSV
Set controller status information	SSV
Send output values to an I/O module or consuming controller at a specific point in your logic Trigger an event task in another controller	IOT

Message (MSG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The MSG instruction asynchronously reads or writes a block of data to another module on a network.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

MSG(MessageControl);

Operands

Ladder Diagram

Operand	Type	Format	Description
Message	MSG	tag	Message structure

Structured Text

Operand	Type	Format	Description
Message	MSG	tag	Message structure

See *Structured Text Syntax* on page 879 for more information on the syntax of expressions within structured text.

MESSAGE Structure

IMPORTANT: If you check the status bits more than once:

Use a copy of the bits if you check them in more than one place in your logic. Otherwise, the bits may change during the scan and your logic won't work as you expect it.

One way to make a copy is to use the FLAGS word. Copy the FLAGS word to another tag and check the bits in the copy.

IMPORTANT: Do not change the following bits of a MSG instruction:

- DN
- EN
- ER
- EW
- ST

Do not change these bits either by themselves or as part of the FLAGS word. If you do, the controller may have a non-recoverable fault. The controller clears the project from its memory when it has a non-recoverable fault.

Mnemonic	Data Type	Description	
.FLAGS	INT	The .FLAGS member provides access to the status members (bits) in one, 16-bit word.	
		This bit 2	Is this member .EW
		4	.ER
		5	.DN

Mnemonic	Data Type	Description	
		6	.ST
		7	.EN
		8	.TO
		9	.EN_CC
		Important: Do not change the EW, ER, DN, or ST bits of the FLAGS member. For example, do not clear the entire FLAGS word. The controller ignores the change and uses the internally-stored values of the bits.	
.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. **Important:** Do not change the EW bit. The controller ignores the change and uses the internally-stored value of the bit.	
.ER	BOOL	The error bit is set when the controller detects that a transfer failed. The .ER bit is reset the next time the EnableIn goes from false to true. **Important:** Do not change the ER bit. The controller ignores the change and uses the internally-stored value of the bit.	
.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 EnableIn goes from false to true. **Important:** Do not change the DN bit. The controller ignores the change and uses the internally-stored value of the bit.	
.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. **Important:** Do not change the ST bit. The controller ignores the change and uses the internally-stored value of the bit.	
.EN	BOOL	The enable bit is set when the EnableIn goes true and remains set until either the .DN bit or the .ER bit is set and the EnableIn is false. If the EnableIn goes false, but the .DN bit and the .ER bit are cleared, the .EN bit remains set. **Important:** Do not change the EN bit. The controller ignores the change and uses the internally-stored value of the bit.	

<table>
<tr><th>Mnemonic</th><th>Data Type</th><th colspan="2">Description</th></tr>
<tr><td>.TO</td><td>BOOL</td><td colspan="2">If you manually set the .TO bit, the controller stops processing the message and sets the .ER bit.</td></tr>
<tr><td>.EN_CC</td><td>BOOL</td><td colspan="2">The enable cache bit determines how to manage the MSG connection. If you want the controller to maintain the connection (such as when you repeat the same MSG instruction many times), set the .EN_CC bit. If you rarely execute the MSG instruction and have other needs for a controller connection, clear the .EN_CC bit.
Connections for MSG instructions going out the serial port are not cached, even if the .EN_CC bit is set.</td></tr>
<tr><td>.ERR_SRC</td><td>SINT</td><td colspan="2">Shows the error path in the Message Configuration dialog.</td></tr>
<tr><td>.DestinationLink</td><td>INT</td><td colspan="2">To change the Destination Link of a DH+ or CIP with Source ID message, set this member to the required value.</td></tr>
<tr><td>.DestinationNode</td><td>INT</td><td colspan="2">To change the Destination Node of a DH+ or CIP with Source ID message, set this member to the required value.</td></tr>
<tr><td>.SourceLink</td><td>INT</td><td colspan="2">To change the Source Link of a DH+ or CIP with Source ID message, set this member to the required value.</td></tr>
<tr><td>.Class</td><td>INT</td><td colspan="2">To change the Class parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td>.Attribute</td><td>INT</td><td colspan="2">To change the Attribute parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td>.Instance</td><td>DINT</td><td colspan="2">To change the Instance parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td rowspan="4">.LocalIndex</td><td rowspan="4">DINT</td><td colspan="2">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.</td></tr>
<tr><td>If the message:</td><td>Then the local array is the:</td></tr>
<tr><td>Reads data</td><td>Destination element</td></tr>
<tr><td>Writes data</td><td>Source element</td></tr>
<tr><td>.Channel</td><td>SINT</td><td colspan="2">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.</td></tr>
<tr><td>.Rack</td><td>SINT</td><td colspan="2">To change the rack number for a block transfer message, set this member to the required rack number (octal).</td></tr>
<tr><td>.Group</td><td>SINT</td><td colspan="2">To change the group number for a block transfer message, set this member to the required group number (octal).</td></tr>
<tr><td rowspan="2">.Slot</td><td rowspan="2">SINT</td><td colspan="2">To change the slot number for a block transfer message, set this member to the required slot number.</td></tr>
<tr><td>If the message goes over this network:</td><td>Then specify the slot number in:</td></tr>
</table>

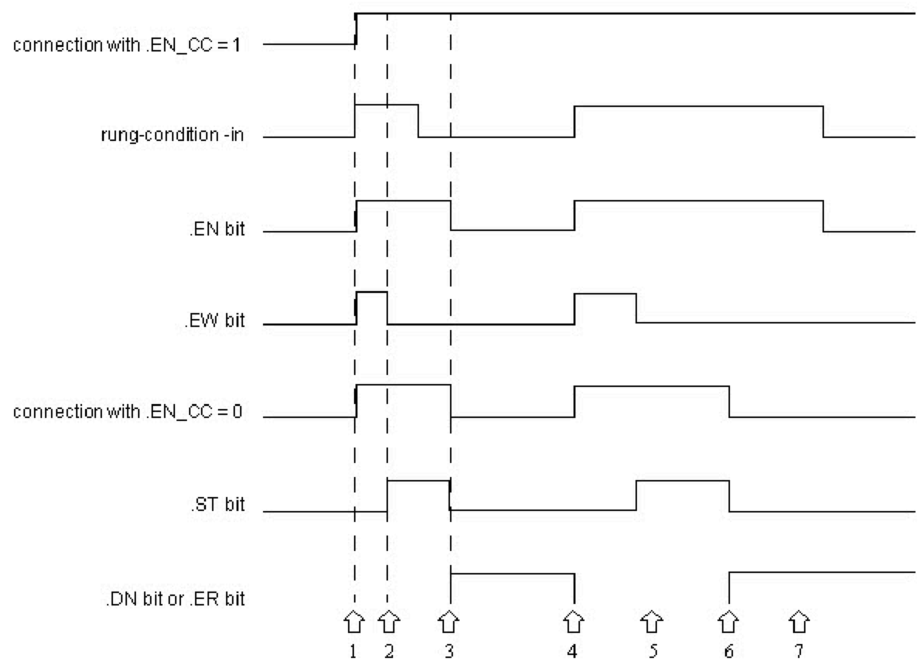
Mnemonic	Data Type	Description	
		Universal remote I/O	octal
		ControlNet	decimal (0-15)
.Path	STRING	To send the message to a different controller, set this member to the new path. Enter the path as hexadecimal values. Omit commas [,] For example, for a path of 1, 0, 2, 42, 1, 3, enter $01$00$02$2A$01$03. To browse to a device and automatically create a portion or all of the new string, right-click a string tag and choose Go to Message Path Editor.	
.RemoteIndex	DINT	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.	
		If the message	Then the remote array is the
		Reads data	Source element
		Writes data	Destination element
.RemoteElement	STRING	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.	
		If the message	Then the remote array is the
		Reads data	Source element
		Writes data	Destination element
.UnconnectedTimeout	DINT	The time out for an unconnected message or for making a connection. The default value is 30 seconds. If the message is unconnected, the ER bit turns on if the controller doesn't get a response within the UnconnectedTimeout time. If the message is connected, the ER bit turns on if the controller doesn't get a response for making the connection within the UnconnectedTimeout time.	
.ConnectionRate	DINT	Time out for a connected message once it has a connection. This time out is for the response from the other device. This time out applies only after the connection is made. The time out = ConnectionRate x TimeoutMultiplier 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).	
.TimeoutMultiplier	SINT		

Mnemonic	Data Type	Description
		To change the time out, change the ConnectionRate and leave the TimeoutMultiplier at the default value.

Description

The MSG instruction transfers elements of data. This is a transitional instruction:

- In ladder diagram, toggle the EnableIn from cleared to set each time the instruction executes.
- The size of each element depends on the data types you specify and the type of message command you use.



Where	Description
1	EnableIn is true .EN is set .EW is set connection is opened
2	message is sent .ST is set .EW is cleared
3	message is done or errored EnableIn is false .DN or .ER is set .ST is cleared connection is closed (if .EN_CC = 0) .EN is cleared (because the EnableIn is false)
4	EnableIn is true and .DN or .ER was previously set .EN is set .EW is set connection is opened

Where	Description
	.DN or .ER is cleared
5	message is sent .ST is set .EW is cleared
6	message is done or errored and EnableIn is still true .DN or .ER is set .ST is cleared connection is closed (if .EN_CC = 0)
7	EnableIn goes false and .DN or .ER is set .EN is cleared

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

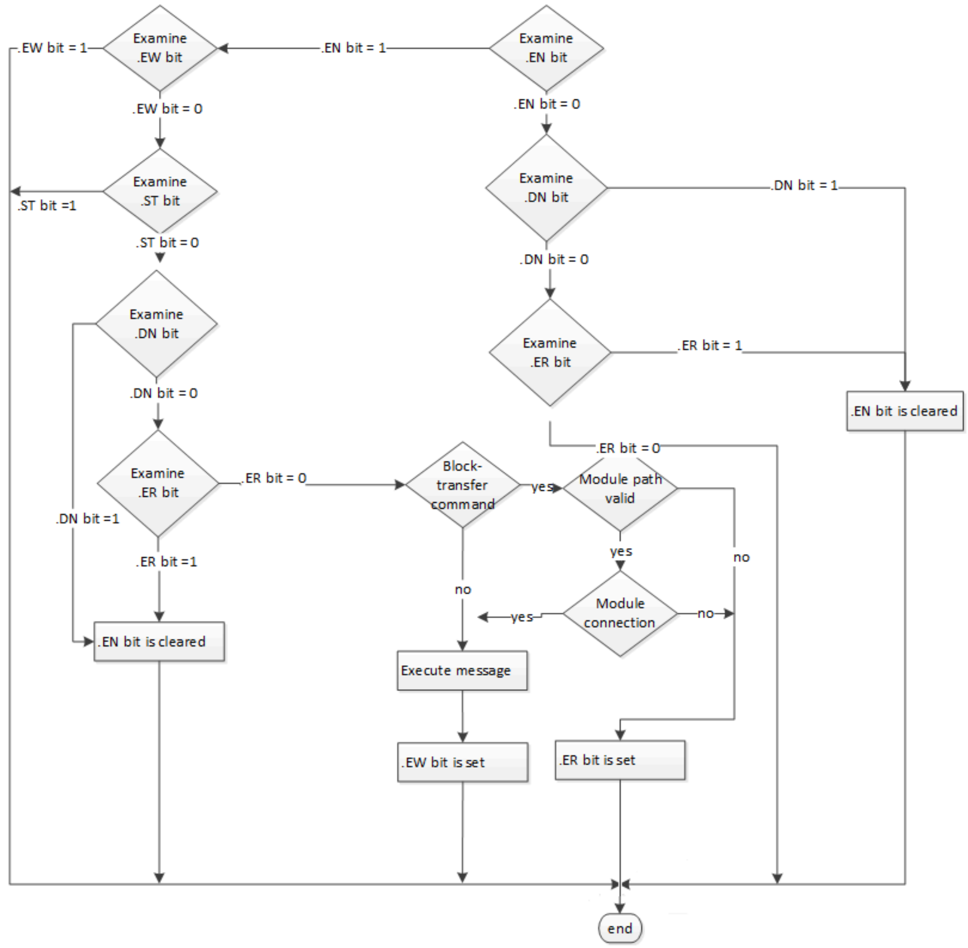
Ladder Diagram

Condition/State	Action Taken
Prescan	The .EWS, .ST, .DN, and .ER bits are cleared.
Rung-condition-in is false	See MSG Flow Chart (False)
Rung-condition-in is true	See MSG Flow Chart (True)
Postscan	N/A

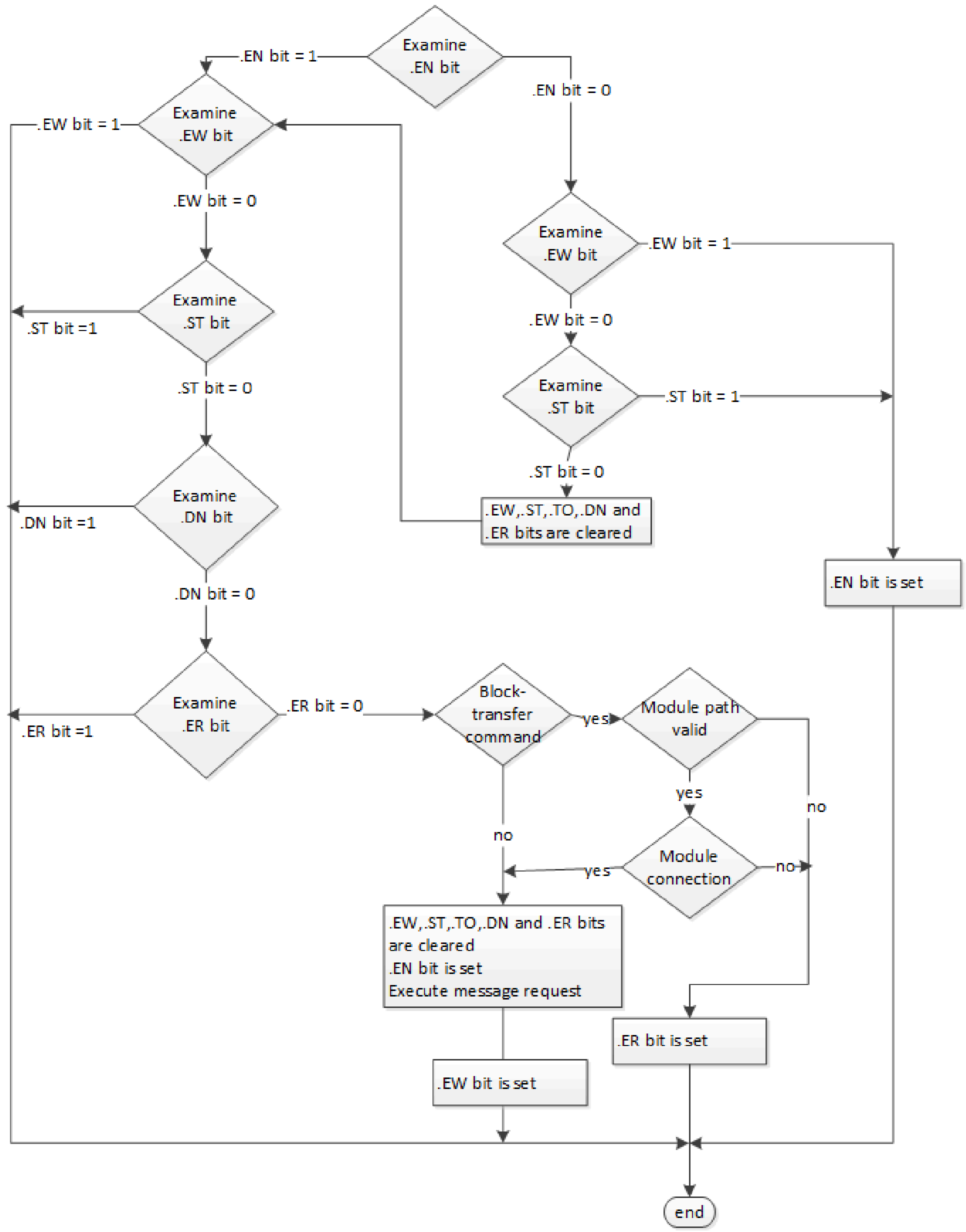
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See MSG Flow Chart (True)
Postscan	See Postscan in the Ladder Diagram table

MSG Flow Chart (False)

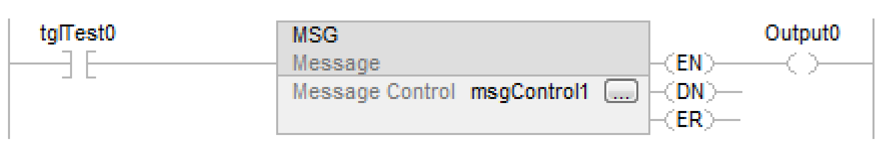


MSG Flow Chart (True)



Example

Ladder Diagram



Structured Text

MSG (MessageControl);

Message (MSG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

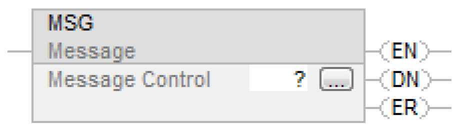
The MSG instruction asynchronously reads or writes a block of data to another module on a network.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MSG(MessageControl);

Operands

Ladder Diagram

Operand	Type	Format	Description
Message	MSG	tag	Message structure

Structured Text

Operand	Type	Format	Description
Message	MSG	tag	Message structure

See *Structured Text Syntax* on page 879 for more information on the syntax of expressions within structured text.

MESSAGE Structure

IMPORTANT: If you check the status bits more than once:

Use a copy of the bits if you check them in more than one place in your logic. Otherwise, the bits may change during the scan and your logic won't work as you expect it.

One way to make a copy is to use the FLAGS word. Copy the FLAGS word to another tag and check the bits in the copy.

IMPORTANT: Do not change the following bits of a MSG instruction:

- DN
- EN
- ER
- EW
- ST

Do not change these bits either by themselves or as part of the FLAGS word. If you do, the controller may have a non-recoverable fault. The controller clears the project from its memory when it has a non-recoverable fault.

Mnemonic	Data Type	Description	
.FLAGS	INT	The .FLAGS member provides access to the status members (bits) in one, 16-bit word.	
		This bit 2	Is this member .EW
		4	.ER
		5	.DN
		6	.ST
		7	.EN
		8	.TO
		9	.EN_CC
		Important: Do not change the EW, ER, DN, or ST bits of the FLAGS member. For example, do not clear the entire FLAGS word. The controller ignores the change and uses the internally-stored values of the bits.	
.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.	

Mnemonic	Data Type	Description
.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. **Important:** Do not change the EW bit. The controller ignores the change and uses the internally-stored value of the bit.
.ER	BOOL	The error bit is set when the controller detects that a transfer failed. The .ER bit is reset the next time the EnableIn goes from false to true. **Important:** Do not change the ER bit. The controller ignores the change and uses the internally-stored value of the bit.
.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 EnableIn goes from false to true. **Important:** Do not change the DN bit. The controller ignores the change and uses the internally-stored value of the bit.
.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. **Important:** Do not change the ST bit. The controller ignores the change and uses the internally-stored value of the bit.
.EN	BOOL	The enable bit is set when the EnableIn goes true and remains set until either the .DN bit or the .ER bit is set and the EnableIn is false. If the EnableIn goes false, but the .DN bit and the .ER bit are cleared, the .EN bit remains set. **Important:** Do not change the EN bit. The controller ignores the change and uses the internally-stored value of the bit.
.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. If you want the controller to maintain the connection (such as when you repeat the same MSG instruction many times), set the .EN_CC bit. If you rarely execute the MSG instruction and have other needs for a controller connection, clear the .EN_CC bit. Connections for MSG instructions going out the serial port are not cached, even if the .EN_CC bit is set.
.ERR_SRC	SINT	Shows the error path in the Message Configuration dialog.
.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.

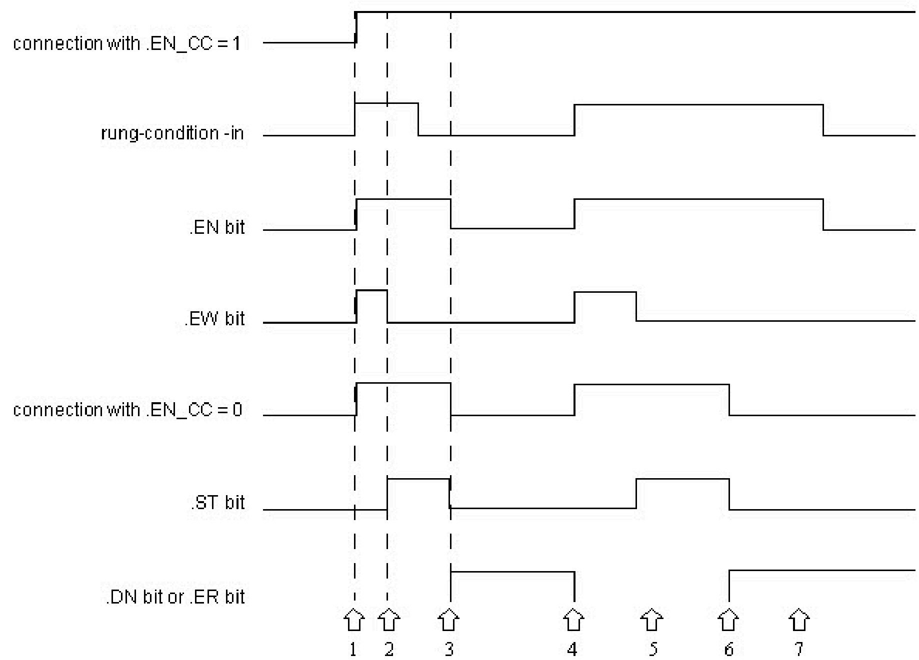
<table>
<tr><th>Mnemonic</th><th>Data Type</th><th>Description</th></tr>
<tr><td>.SourceLink</td><td>INT</td><td>To change the Source Link of a DH+ or CIP with Source ID message, set this member to the required value.</td></tr>
<tr><td>.Class</td><td>INT</td><td>To change the Class parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td>.Attribute</td><td>INT</td><td>To change the Attribute parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td>.Instance</td><td>DINT</td><td>To change the Instance parameter of a CIP Generic message, set this member to the required value.</td></tr>
<tr><td>.LocalIndex</td><td>DINT</td><td>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.
<table>
<tr><th>If the message:</th><th>Then the local array is the:</th></tr>
<tr><td>Reads data</td><td>Destination element</td></tr>
<tr><td>Writes data</td><td>Source element</td></tr>
</table></td></tr>
<tr><td>.Channel</td><td>SINT</td><td>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.</td></tr>
<tr><td>.Rack</td><td>SINT</td><td>To change the rack number for a block transfer message, set this member to the required rack number (octal).</td></tr>
<tr><td>.Group</td><td>SINT</td><td>To change the group number for a block transfer message, set this member to the required group number (octal).</td></tr>
<tr><td>.Slot</td><td>SINT</td><td>To change the slot number for a block transfer message, set this member to the required slot number.
<table>
<tr><th>If the message goes over this network:</th><th>Then specify the slot number in:</th></tr>
<tr><td>Universal remote I/O</td><td>octal</td></tr>
<tr><td>ControlNet</td><td>decimal (0-15)</td></tr>
</table></td></tr>
<tr><td>.Path</td><td>STRING</td><td>To send the message to a different controller, set this member to the new path.
Enter the path as hexadecimal values.
Omit commas [,]
For example, for a path of 1, 0, 2, 42, 1, 3, enter $01$00$02$2A$01$03.
To browse to a device and automatically create a portion or all of the new string, right-click a string tag and choose Go to Message Path Editor.</td></tr>
<tr><td>.RemoteIndex</td><td>DINT</td><td>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>
</table>

Mnemonic	Data Type	Description	
		If the message	Then the remote array is the
		Reads data	Source element
		Writes data	Destination element
.RemoteElement	STRING	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.	
		If the message	Then the remote array is the
		Reads data	Source element
		Writes data	Destination element
.UnconnectedTimeout	DINT	The time out for an unconnected message or for making a connection. The default value is 30 seconds. If the message is unconnected, the ER bit turns on if the controller doesn't get a response within the UnconnectedTimeout time. If the message is connected, the ER bit turns on if the controller doesn't get a response for making the connection within the UnconnectedTimeout time.	
.ConnectionRate	DINT	Time out for a connected message once it has a connection. This time out is for the response from the other device. This time out applies only after the connection is made. The time out = ConnectionRate x TimeoutMultiplier 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.	
.TimeoutMultiplier	SINT		

Description

The MSG instruction transfers elements of data. This is a transitional instruction:

- In ladder diagram, toggle the EnableIn from cleared to set each time the instruction executes.
- The size of each element depends on the data types you specify and the type of message command you use.



Where	Description
1	EnableIn is true .EN is set .EW is set connection is opened
2	message is sent .ST is set .EW is cleared
3	message is done or errored EnableIn is false .DN or .ER is set .ST is cleared connection is closed (if .EN_CC = 0) .EN is cleared (because the EnableIn is false)
4	EnableIn is true and .DN or .ER was previously set .EN is set .EW is set connection is opened .DN or .ER is cleared
5	message is sent .ST is set .EW is cleared
6	message is done or errored and EnableIn is still true .DN or .ER is set .ST is cleared connection is closed (if .EN_CC = 0)
7	EnableIn goes false and .DN or .ER is set

Where	Description
	.EN is cleared

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

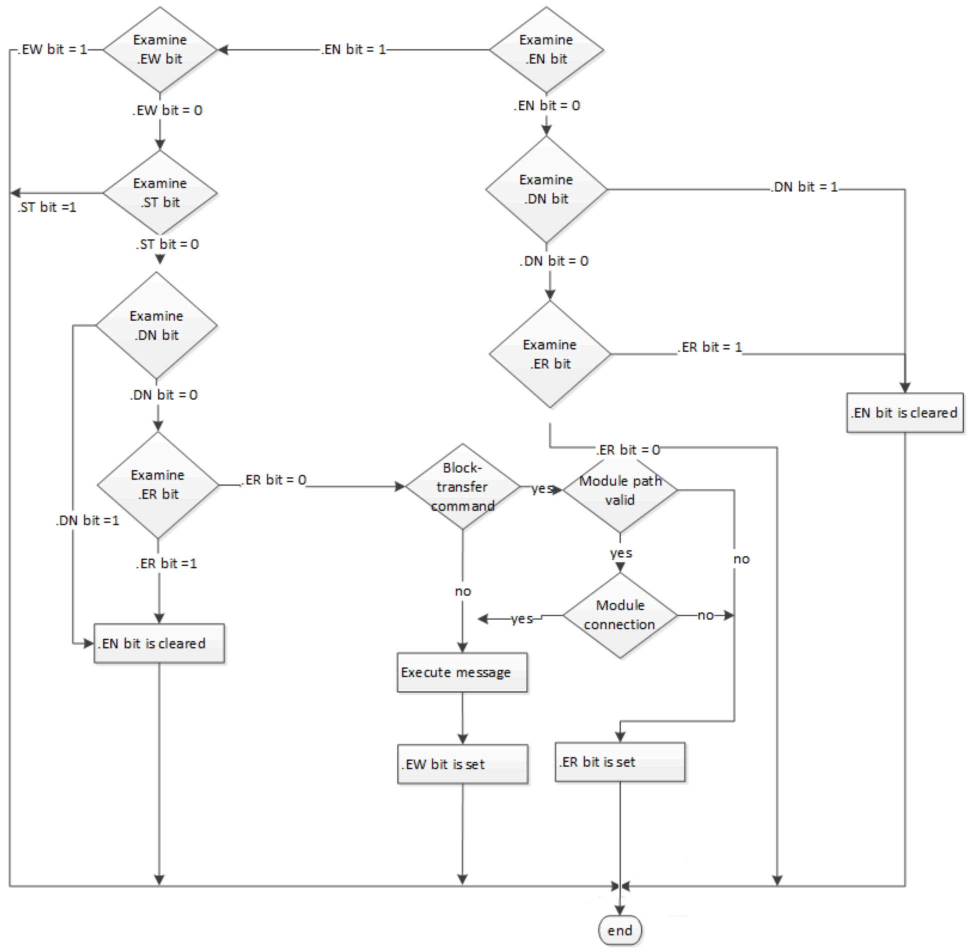
Ladder Diagram

Condition/State	Action Taken
Prescan	The .EWS, .ST, .DN, and .ER bits are cleared.
Rung-condition-in is false	See MSG Flow Chart (False)
Rung-condition-in is true	See MSG Flow Chart (True)
Postscan	N/A

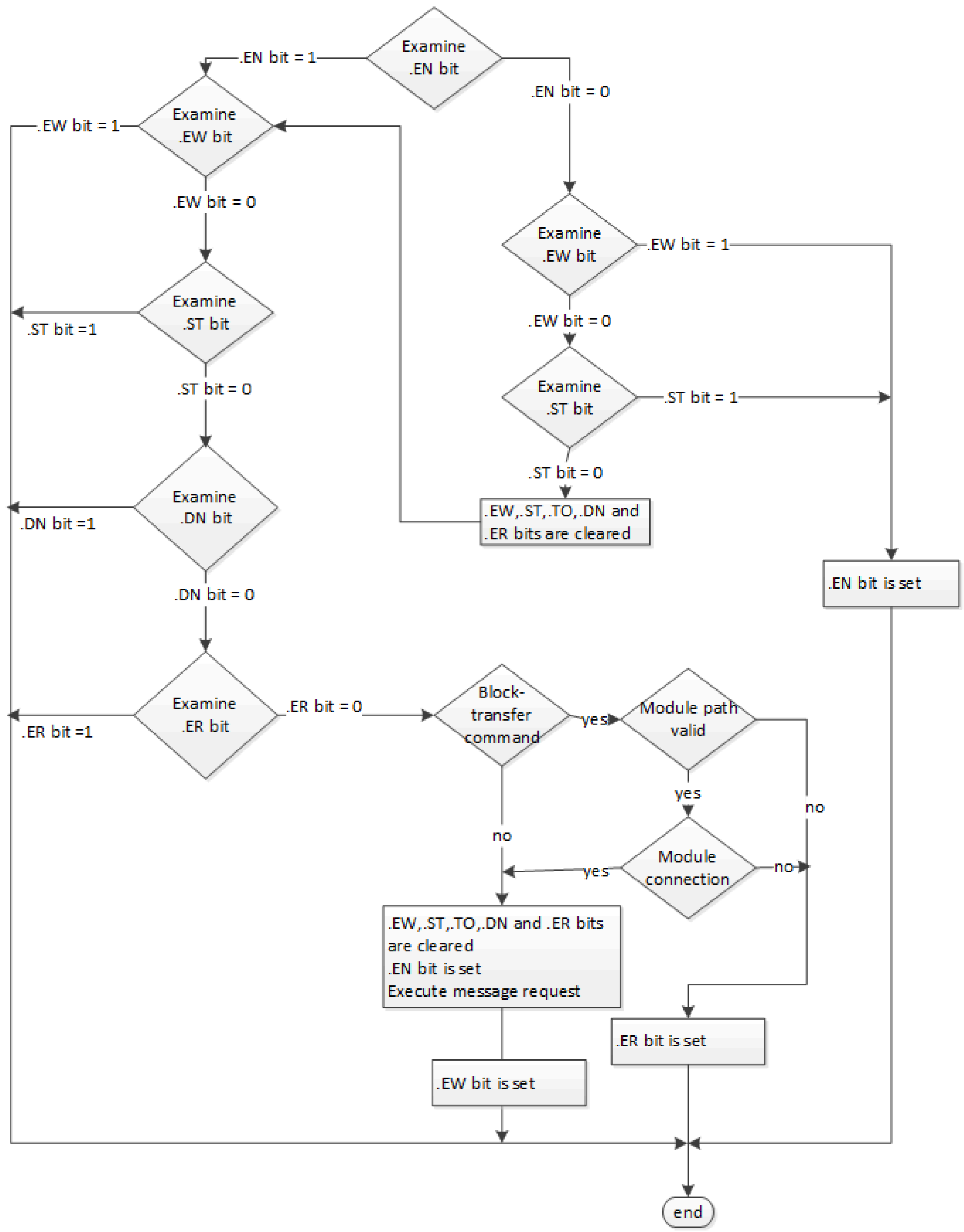
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See MSG Flow Chart (True)
Postscan	See Postscan in the Ladder Diagram table

MSG Flow Chart (False)

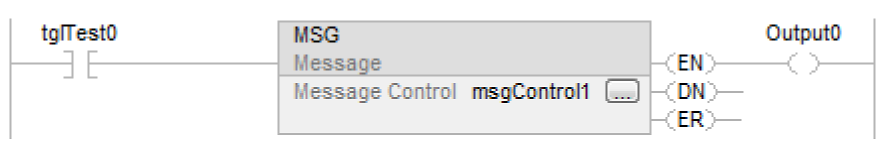


MSG Flow Chart (True)



Example

Ladder Diagram



Structured Text

MSG (MessageControl);

MSG Configuration Examples

The following examples show source and destination tags and elements for different controller combinations.

The table explains the path for MSG instructions originating from a Logix 5000 controller and being written to another controller.

Message Path	Example Source and Destination	
Logix 5000 -> Logix 5000	Source tag	array_1[0]
	Destination tag	array_2[0]
	You can use an alias tag for the source tag in the originating Logix 5000 controller. You cannot use an alias for the destination tag. The destination must be a base tag.	
Logix 5000 -> PLC-5 Logix 5000 -> SLC	Source tag	array_1[0]
	Destination element	N7:10
	You can use an alias tag for the source tag, in the originating Logix 5000 controller.	
Logix 5000 -> PLC-2	Source tag	array_1[0]
	Destination element	010

The table explains the path for MSG instructions originating from a Logix 5000 controller and reading from another controller.

Message Path	Example Source and Destination	
Logix 5000 -> Logix 5000	Source tag	array_1[0]
	Destination tag	array_2[0]
	You cannot use an alias tag for the source tag. The source must be a base tag. You can use an alias tag for the destination tag, in the originating Logix 5000 controller.	
Logix 5000 -> PLC-5 Logix 5000 -> SLC	Source element	N7:10
	Destination tag	array_1[0]
	You can use an alias tag for the destination tag, in the originating Logix 5000 controller.	
Logix 5000 -> PLC-2	Source element	010
	Destination tag	array_1[0]

Major fault types and codes

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

Minor fault types and codes

The following are the minor fault types and codes.

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

Message Error Codes

Error codes depend on the type of MSG instruction.

Error Codes

The Logix Designer application does not always display the full description.

Error Code (Hex)	Description	Display In Software
0001	Connection failure (extended error codes)	Same as description
0002	Insufficient resource	
0003	Invalid value	
0004	IOI syntax error (see extended error codes)	
0005	Destination unknown, class unsupported, instance undefined or structure element undefined (see extended error codes)	
0006	Insufficient packet space	
0007	Connection lost	
0008	Service unsupported	
0009	Error in data segment or invalid attribute value	
000A	Attribute list error	
000B	State already exists	
000C	Object model conflict	
000D	Object already exists	
000E	Attribute cannot be set	
000F	Permission denied	
0010	Device state conflict	
0011	Reply will not fit	
0012	Fragment primitive	
0013	Insufficient command data	
0014	Attribute not supported	
0015	Too much data	
001A	Bridge request too large	
001B	Bridge response too large	

Error Code (Hex)	Description	Display In Software
001C	Attribute list shortage	
001D	Invalid attribute list	Same as description
001E	Embedded service error	
001F	Connection related failure (see extended error codes)	
0022	Invalid reply received	
0025	Key segment error	
0026	Invalid IOI error	
0027	Unexpected attribute in list	
0028	DeviceNet error - invalid member ID	
0029	DeviceNet error - member not settable	
00D1	Module not in run state	Unknown error
00FB	Message port not supported	
00FC	Message unsupported data type	
00FD	Message uninitialized	
00FE	Message timeout	
00FF	General error (see extended error codes)	

Extended Error Codes

The Logix Designer application does not display any text for the extended error codes.

The table lists extended error codes for error code 0001.

Extended Error Code (Hex)	Description
0100	Connection in use
0103	Transport not supported
0106	Ownership conflict
0107	Connection not found
0108	Invalid connection type
0109	Invalid connection size
0110	Module not configured
0111	EPR not supported
0113	MSG write failed
0114	Wrong module
0115	Wrong device type
0116	Wrong revision
0118	Invalid configuration format
011A	Application out of connections

Extended Error Code (Hex)	Description
0203	Connection timeout
0204	Unconnected message timeout
0205	Unconnected send parameter error
0206	Message too large
0301	No buffer memory
0302	Bandwidth not available
0303	No screens available
0305	Signature mismatch
0311	Port not available
0312	Link address not available
0315	Invalid segment type
0317	Connection not scheduled

The table lists the extended error codes for error code 001F.

Extended Error Code (Hex)	Description
0203	Connection timeout

The table lists the extended error codes for error code 0004 and 0005.

Extended Error Code (Hex)	Description
0000	extended status out of memory
0001	extended status out of instances

The table lists the extended error codes for error code 00FF.

Extended Error Code (Hex)	Description
2001	Excessive IOI
2002	Bad parameter value
2018	Semaphore reject
201B	Size too small
201C	Invalid size
2100	Privilege failure
2101	Invalid keyswitch position
2102	Password invalid
2103	No password issued
2104	Address out of range
2105	Address and how many out of range
2106	Data in use
2107	Type is invalid or not supported

Extended Error Code (Hex)	Description
2108	Controller in upload or download mode
2109	Attempt to change number of array dimensions
210A	Invalid symbol name
210B	Symbol does not exist
210E	Search failed
210F	Task cannot start
2110	Unable to write
2111	Unable to read
2112	Shared routine not editable
2113	Controller in faulted mode
2114	Run mode inhibited

PLC and SLC Error Codes (.ERR)

Logix firmware revision 10.x and later provides new error codes for errors that are associated with PLC and SLC™ message types (PCCC messages).

This change lets RSLogix 5000 software display a more meaningful description for many of the errors. Previously the software did not give a description for any of the errors associated with the 00F0 error code.

The change also makes the error codes more consistent with errors returned by other controllers, such as PLC-5® controllers.

The table shows the change in the error codes from R9.x and earlier to R10.x and later. As a result of the change, the .ERR member returns a unique value for each PCCC error. The .EXERR is no longer required for these errors.

PLC and SLC Error Codes (hex)

R9.x And Earlier		R10.x And Later		Description
.ERR	.EXERR	.ERR	.EXERR	
0010		1000		Illegal command or format from local processor
0020		2000		Communication module not working
0030		3000		Remote node is missing, disconnected, or shut down
0040		4000		Processor connected but faulted (hardware)
0050		5000		Wrong station number

R9.x And Earlier		R10.x And Later	Description
0060		6000	Requested function is not available
0070		7000	Processor is in Program mode
0080		8000	Compatibility file of processor does not exist
0090		9000	Remote node cannot buffer command
00B0		B000	Processor is downloading so it is not accessible
00F0	0001	F001	Processor incorrectly converted the address
00F0	0002	F002	Incomplete address
00F0	0003	F003	Incorrect address
00F0	0004	F004	Illegal address format - symbol not found
00F0	0005	F005	Illegal address format - symbol has 0 or greater than the maximum number of characters supported by the device
00F0	0006	F006	Address file does not exist in target processor
00F0	0007	F007	Destination file is too small for the number of words requested
00F0	0008	F008	Cannot complete request Situation changed during multipacket operation
00F0	0009	F009	Data or file is too large Memory unavailable
00F0	000A	F00A	Target processor cannot put requested information in packets
00F0	000B	F00B	Privilege error; access denied
00F0	000C	F00C	Requested function is not available

R9.x And Earlier		R10.x And Later	Description
00F0	000D	F00D	Request is redundant
00F0	000E	F00E	Command cannot be executed
00F0	000F	F00F	Overflow; histogram overflow
00F0	0010	F010	No access
00F0	0011	F011	Data type requested does not match data ava ilable
00F0	0012	F012	Incorrect command parameters
00F0	0013	F013	Address reference exists to deleted area
00F0	0014	F014	Command execution failure for unknown reason PLC-3® histogram overflow
00F0	0015	F015	Data conversion error
00F0	0016	F016	The scanner is not available to communicate with a 1771 rack adapter
00F0	0017	F017	The adapter is no available to communicate with the module
00F0	0018	F018	The 1771 module response was not valid
00F0	0019	F019	Duplicate label
00F0	001A	F01A	File owner active - the file is being used
00F0	001B	F01B	Program owner active - someone is downloading or editing online
00F0	001C	F01C	Disk file is write protected or otherwise not accessible (offline only)
00F0	001D	F01D	Disk file is being used by another application

R9.x And Earlier	R10.x And Later	Description
		Update not performed (offline only)

Block Transfer Error Codes

These are the Logix5000 block-transfer specific error codes.

Error Code (Hex)	Description	Display In Software
00D0	The scanner did not receive a block-transfer response from the block-transfer module within 3.5 seconds of the request.	Unknown error
00D1	The checksum from the read response did not match the checksum of the data stream.	
00D2	The scanner requested either a read or write but the block-transfer module responded with the opposite.	
00D3	The scanner requested a length and the block-transfer module responded with a different length.	
00D6	The scanner received a response from the block-transfer module indicating the write request failed.	
00EA	The scanner was not configured to communicate with the rack that would contain this block-transfer module.	
00EB	The logical slot specified is not available for the given rack size.	
00EC	There is currently a block-transfer request in progress and a response is required before another request can begin.	
00ED	The size of the block-transfer request is not consistent with valid block-transfer size requests.	
00EE	The type of block-transfer request is not consistent with the expected BT_READ or BT_WRITE.	
00EF	The scanner was unable to find an available slot in the block-transfer table	

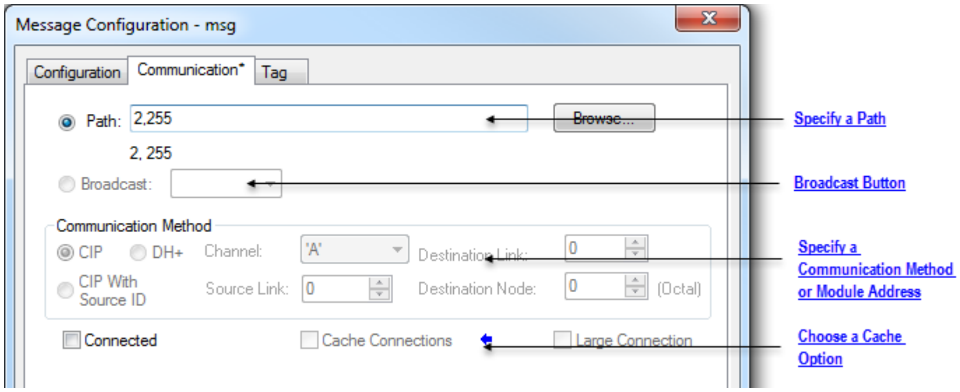
Error Code (Hex)	Description	Display In Software
	to accommodate the block-transfer request.	
00F0	The scanner received a request to reset the remote I/O channels while there were outstanding block-transfers.	
00F3	Queues for remote block-transfers are full.	
00F5	No communication channels are configured for the requested rack or slot.	
00F6	No communication channels are configured for remote I/O.	
00F7	The block-transfer timeout, set in the instruction, timed out before completion.	
00F8	Error in block-transfer protocol - unsolicited block-transfer.	
00F9	Block-transfer data was lost due to a bad communication channel.	
00FA	The block-transfer module requested a different length than the associated block-transfer instruction.	
00FB	The checksum of the block-transfer read data was wrong.	
00FC	There was an invalid transfer of block-transfer write data between the adapter and the block-transfer module.	
00FD	The size of the block-transfer plus the size of the index in the block-transfer data table was greater than the size of the block-transfer data table file.	

Specify the Communication Details

Set up a broadcast in ladder logic or structured text programs. In ladder logic, add a rung and click on the **MSG** property to access the **Message Configuration** dialog box and set up a new message. In structured text, type **MSG** (aMsg) and then right-click the aMsg to open the **Message Configuration** dialog box and configure the message.

NOTE: Logix Designer versions 37 and later do not support controllers with serial ports. The **Broadcast** button appears only in Logix Designer versions 36 and earlier for controllers with a serial port.

To configure a MSG instruction, specify the following on the **Communication** tab:



Specify a Path

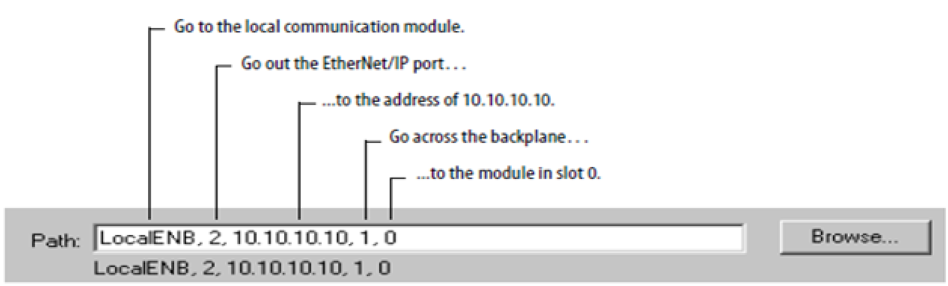
The path shows the route that the message takes to get to the destination. It uses names from the I/O configuration of the controller, numbers that you type, or both. You can default the path by using the broadcast button, which must be enabled with the system protocol and message type.

If	Then
The I/O configuration of the controller has the module that gets the message.	Browse to select the module.
The I/O configuration of the controller has only the local communication module.	Browse to select the local communication module and type the rest of the path.
The I/O configuration of the controller does not have any of the modules required for the message.	Type the path.

Tip: Also supported is THIS, which indicates a path to self. THIS is used to send an unconnected message to the controller.

Examples

The I/O configuration of the controller has only the local communication module:



To type a path, use the format:

port, next_address, port, next_address,

Where	Is	
	For this network	Type
Port	Backplane	1
	DF1 (serial, serial channel 0)	2
	ControlNet	
	EtherNet/IP	
	DH+ channel A	
	DH+ channel B	3
	DF1 channel 1 (serial channel 1)	
Next_address	Backplane	Slot number of the module
	DF1 (serial)	Station address (0-254)
	ControlNet	Node number (1-99 decimal)
	DH+	8# followed by the node number (1-77 octal) For example, to specify the octal node address of 37, type 8#37.
	EtherNet/IP	Specify a module on an EtherNet/IP network by using any of these formats: • IP address. For example, 10.10.10.10 • IP address:Port. For example, 10.10.10.10:24 • DNS name. For example, tanks • DNS name:Port. For example, tanks:24

Broadcast Button

The **Broadcast** button is used with the serial port.

- This functionality for RSLogix 5000 software, beginning with version 18, enhances the ability to define the route and message type that are required to send a message to its destination.

The **Broadcast** button, when enabled, allows you to default the path by selecting an available channel(s) in a combo box. The number of channels listed in the combo box depends on the current controller.

By default, the **Path** button on the **Communication** tab is active.

Perform these steps to enable the **Broadcast** button and select a channel to default a path for the message.

1. On the **Controller Organizer**, right-click **Controller**, and select **Properties**. The **Controller Properties** dialog box appears.
2. Click the **System Protocol** tab.
3. Select **DF1 Master** in the **Protocol** box. The Polling mode defaults 'Message Based' (slave can initiate messages).
4. Click **OK**.

5. In ladder logic, click the box inside the MSG tag. The **Message Configuration** dialog box appears with the **Configuration** tab open.
6. In the **Message Type** box, select **CIP Data Table Write**.
7. Click **OK**.You have enabled the **Broadcast** button on the **Communication** tab.
8. Click the **Communication** tab.
9. Next to the **Broadcast** button, select a channel in the combo box. The number of channels in the combo box depends on the controller.

   When you select channel 0 or 1, the corresponding message path on the **Message Configuration** dialog box defaults to 2,255 (channel 0) or 3,255 (channel 1). The Path grays out to not allow you to manually enter a path value.
10. Click **OK**.

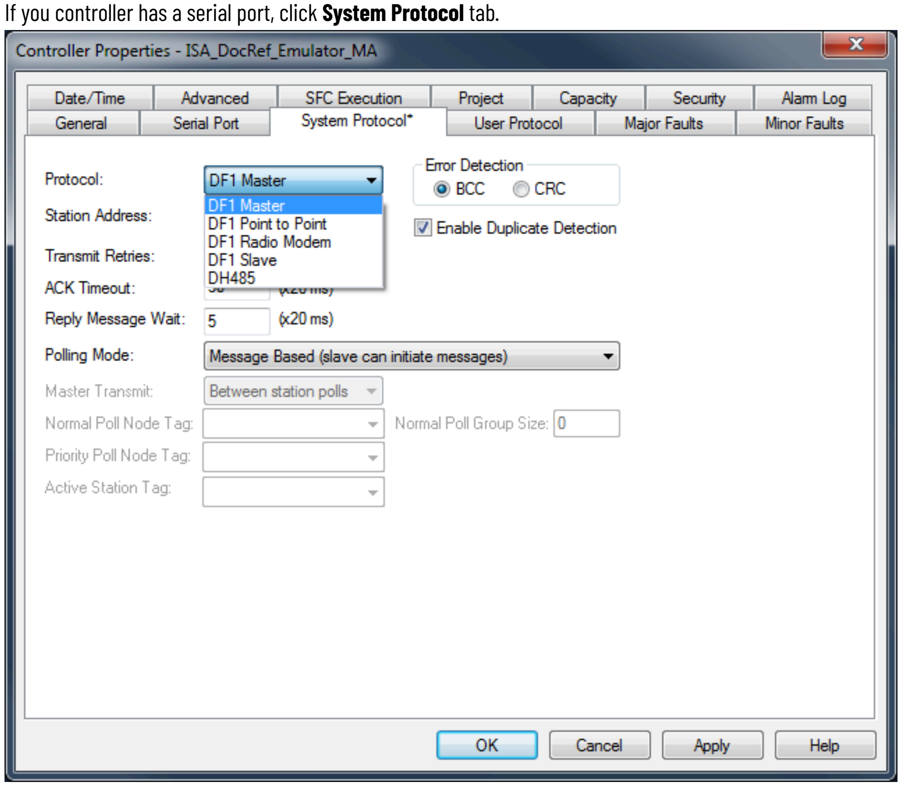
System Protocol Tab Configuration

NOTE: Logix Designer versions 37 and later do not support controllers with serial ports. The **System Protocol** tab appears only in Logix Designer versions 36 and earlier for controllers with a serial port.

To run broadcast in ControlLogix controllers in the Logix Designer application, you must configure the **System Protocol** tab in the **Controller Properties** dialog box. The protocol must be compatible with the message type of 'write' on the **Message Configuration** dialog box.

Follow these steps to set up the system protocol to be compatible with the broadcast feature.

1. Create or open an existing controller in the application.
2. On the **Controller Organizer**, right-click the controller name, and select **Properties**.The **Controller Properties** dialog box appears.
3. If you controller has a serial port, click **System Protocol** tab.



4. In the Protocol box, select a protocol.

IMPORTANT: The **Message Type** box on the **Message Configuration Tab** dialog box must be write-typed to be compatible with the system protocol. Otherwise, the **Broadcast** button is disabled.

5. Enter the information on the **System Protocol** tab for each protocol outlined in the following tables.

Topic	Description
Protocol	DF-1 Master
Station Address	Type controller station address number
Transmit Retries	3
ACK Timeout	50
Reply Message Wait	5
Polling Mode	Select from the following modes: ◦ **Message Based** Poll the slave using message instruction ◦ **Slave can initiate message** for slave to slave broadcast ◦ **Standard**. to have the schedule poll for the slave
Error Detection	BCC
Duplicate Detection	Enabled (checked)

Topic	Description
Protocol	DF-1 Slave
Station Address	Type controller station address number
Transmit Retries	3
Slave Poll Timeout	3000
EOT Suppression	Disable (unchecked)
Error Detection	BCC
Duplicate Detection	Enabled (checked)

Topic	Description
Protocol	DF-1 Slave
Station Address	Type controller station address number
Enable Store and Forward	Enable box (checkmark) to use store and forward tag
Error Detection	BCC

6. Click **OK**.

For Block Transfers

For block transfer messages, add the following modules to the I/O configuration of the controller:

For block-transfers over this network:	Add these modules to the I/O configuration:
ControlNet	Local communication module (for example, 1756-CNB module) Remote adapter module (for example, 1771-ACN module)
Universal remote I/O	Local communication module (for example, 1756-DHRIO module) One remote adapter module (for example, 1771-ASB module) for each rack, or portion of a rack, in the chassis Block-transfer module (optional)

Specify a Communication Method or Module Address

Use the following table to select a communication method or module address for the message:

If the destination device is	Select	And specify	
Logix 5000 controller	CIP	No other specifications required.	
PLC-5 controller over an EtherNet/IP network			
PLC-5 controller over a ControlNet network			
SLC 5/05 controller			
PLC-5 controller over a DH+ network	DH+	Channel	Channel A or B of the 1756-DHRIO module that is connected to the DH+ network.
SLC controller over a DH+ network		Source Link	Link ID assigned to the backplane of the controller in the routing table of the 1756-DHRIO module. The source node in the routing table is automatically the slot number of the controller.
PLC-3 processor		Destination Link	Link ID of the remote DH+ link where the target device resides.
PLC-2 processor		Destination Node	Station address of the target device, in octal.
		If there is only one DH+ link and you did not use the RSLinx Classic software to configure the DH/RIO module for remote links, specify 0 for both the Source Link and the Destination Link.	
Application on a workstation that is receiving an unsolicited message routed over an EtherNet/IP or ControlNet network through RSLinx	CIP with Source ID This lets the application receive data from a controller.	Source Link	Remote ID of the topic in RSLinx Classic software, or the shortcut in FactoryTalk Linx.

If the destination device is	Select	And specify	
Classic or FactoryTalk Linx software		Destination Link	Virtual Link ID set up in RSLinx Classic or FactoryTalk Linx software (0...65535).
		Destination Node	Destination ID (0...77 octal) provided by the application to RSLinx Classic or FactoryTalk Linx. For a DDE topic in RSLinx Classic , use 77.
		The slot number of the ControlLogix controller is used as the Source Node.	
Block transfer module over a universal remote I/O network	RIO	Channel	Channel A or B of the 1756-DHRIO module that is connected to the RIO network.
		Rack	Rack number (octal) of the module.
		Group	Group number of the module.
		Slot	Slot number of the module.
Block transfer module over a ControlNet network	ControlNet	Slot	Slot number of the module.

Choose a Cache Option

Depending on the configuration of an MSG instruction, it may use a connection to send or receive data.

Message type:	Communication method:	Uses a connection:
CIP data table read or write		Your option(1)
PLC-2, PLC-3, PLC-5, or SLC (all types)	CIP CIP with Source ID	
	DH+	X
CIP generic		Your option(2)
Block-transfer read or write		X

1. CIP data table read or write messages can be connected or unconnected. For most applications, Rockwell Automation recommends that you leave CIP data table read or write messages connected.
2. CIP generic messages can be connected or unconnected. But, for most applications, we recommend you leave CIP generic messages unconnected.

If a MSG instruction uses a connection, you have the option to leave the connection open (cache) or close the connection when the message is done transmitting.

If you:	Then:
Cache the connection	The connection stays open after the MSG instruction is done. This optimizes execution time. Opening a connection each time the message executes increases execution time.
Do not cache the connection	The connection closes after the MSG instruction is done. This frees up that connection for other uses.

The controller has the following limits on the number of connections that you can cache.

If you have this controller:	Then you can cache:
CompactLogix 5370 or ControlLogix 5570	Up to 32 connections.
ControlLogix 5580	Up to 256 connections.

If several messages go to the same device, the messages may be able to share a connection.

If the MSG instructions are to:	And they are:	Then:
Different devices		Each MSG instruction uses 1 connection.
Same device	Enabled at the same time	Each MSG instruction uses 1 connection.
	NOT enabled at the same time	The MSG instruction uses 1 connection and 1 cached buffer. They share the connection and the buffer.

Tip: To share a connection, if the controller alternates between sending a block-transfer read message and a block-transfer write message to the same module, both messages count as one connection. Caching both messages counts as one on the cache list.

Guidelines

As you plan and program your MSG instructions, follow these guidelines:

Guideline	Details
For each MSG instruction, create a control tag.	Each MSG instruction requires its own control tag.
	Data type = MESSAGE
	Scope = controller
	The tag cannot be part of an array or a user-defined data type.
Keep the source and/or destination data at the controller scope.	A MSG instruction can access only tags that are in the Controller Tags folder (controller scope).
If your MSG is to a device that uses 16-bit integers, use a buffer of INTs in the MSG and DINTs throughout the project.	If your message is to a device that uses 16-bit integers, such as a PLC-5 or SLC 500 controller, and it transfers integers (not REALs), use a buffer of INTs in the message and DINTs throughout the project.

Guideline	Details
	This increases the efficiency of your project because Logix controllers execute more efficiently and use less memory when working with 32-bit integers (DINTs).
	To convert between INTs and DINTs, see the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.
Cache the connected MSGs that execute most frequently.	Cache the connection for those MSG instructions that execute most frequently, up to the maximum number permissible for your controller revision. This optimizes execution time because the controller does not have to open a connection each time the message executes.
For the CompactLogix 5370 or ControlLogix 5570 controllers , if you want to enable more than 16 MSGs at one time, use some type of management strategy. For the ControlLogix 5580 controllers, if you want to enable more than 256 MSGs at one time, use some type of management strategy.	For the CompactLogix 5370 or ControlLogix 5570 controllers, if you enable more than 16 MSGs at one time, some MSG instructions may experience delays in entering the queue. For the ControlLogix 5580 controllers, if you enable more than 256 MSGs at one time, some MSG instructions may experience delays in entering the queue. To help make sure that each message executes, use one of these options:
	Enable each message in sequence.
	Enable the messages in groups.
	Program a message to communicate with multiple devices. For more information, see the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.
	Program logic to coordinate the execution of messages. For more information, see the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.
(For the CompactLogix 5370 or ControlLogix 5570 controllers only) Keep the number of unconnected and uncached MSGs less than the number of unconnected buffers.	The controller can have 10...40 unconnected buffers. The default number is 10 for the CompactLogix 5370 or ControlLogix 5570 controllers.
	If all the unconnected buffers are in use when an instruction leaves the message queue, the instruction errors and does not transfer the data.
	You can increase the number of unconnected buffers (40 max), but continue to follow guideline 5.
	To increase the number of unconnected buffers, see the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

Specify SLC Messages

Use the SLC message types to communicate with SLC and MicroLogix controllers. The following table specifies which data types the instruction allows you access. The table also shows the corresponding Logix 5000 data type.

For this SLC or MicroLogix data type:	Use this Logix 5000 data type:
F	REAL
L (MicroLogix 1200 and 1500 controllers)	DINT
N	INT

Specify Block Transfer Messages

The block-transfer message types are used to communicate with block-transfer modules over a Universal Remote I/O network.

To:	Select this command:
Read data from a block-transfer module This message type replaces the BTR instruction	Block-Transfer Read
Write data to a block-transfer module This message type replaces the BTW instruction	Block-Transfer Write

To configure a block-transfer message, follow these guidelines:

- The source (for BTW) and destination (for BTR) tags must be large enough to accept the requested data, except for MESSAGE, AXIS, and MODULE structures.
- Specify how many 16-bit integers (INT) to send or receive. You can specify from 0 to 64 integers.

Tip: To have the block-transfer module determine how many 16-bit integers to send (BTR), or to have the controller send 64 integers (BTW), type **0** for the number of elements.

Get System Value (GSV) and Set System Value (SSV)

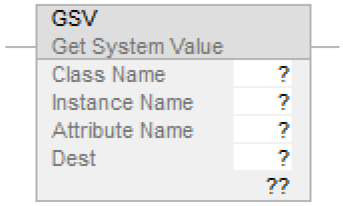
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

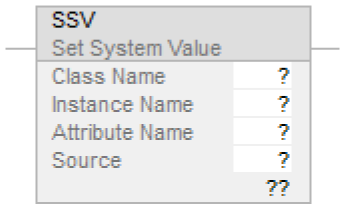
he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram





Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data

Operand	Type	Format	Description
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

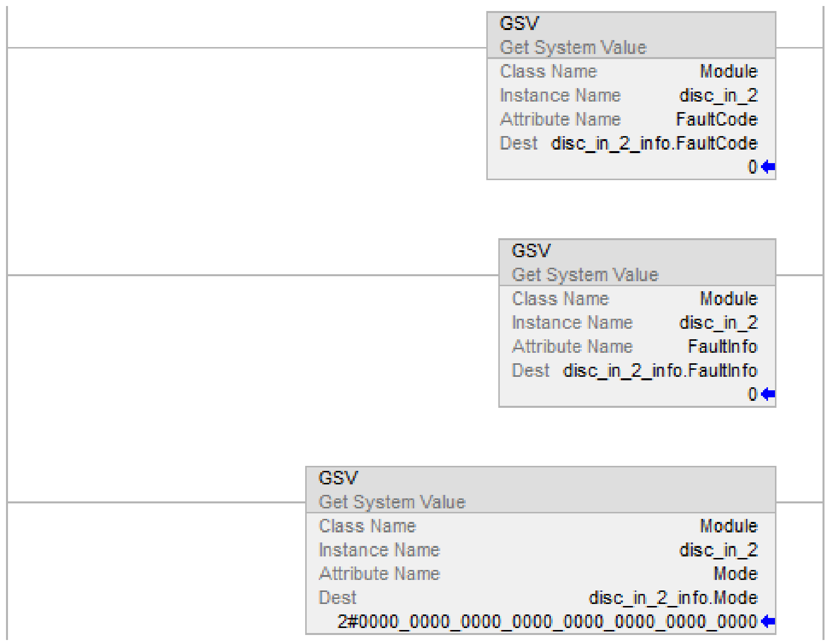
When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

You must test and confirm that the instructions do not change data that you do not want to change.

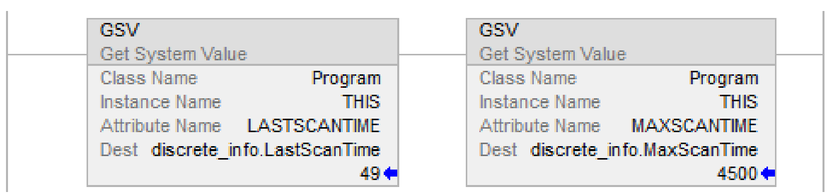
The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1



Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

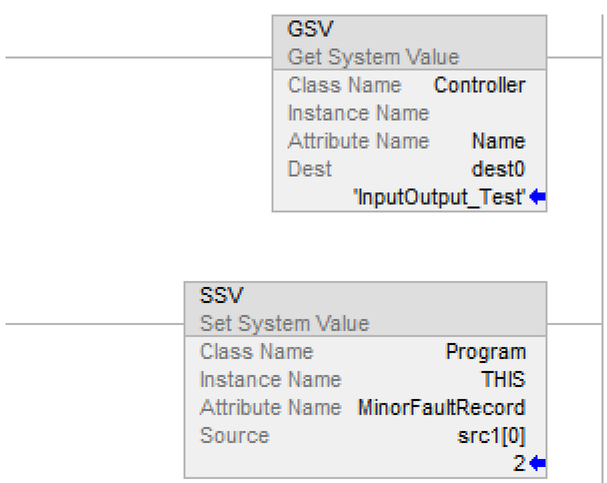
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

Get System Value (GSV) and Set System Value (SSV)

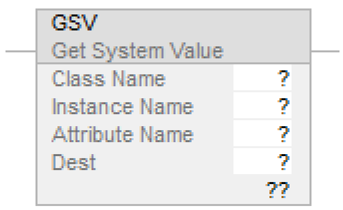
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

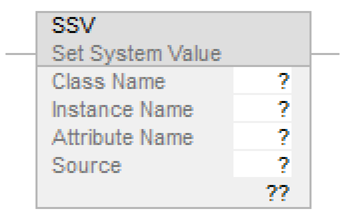
he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram





Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

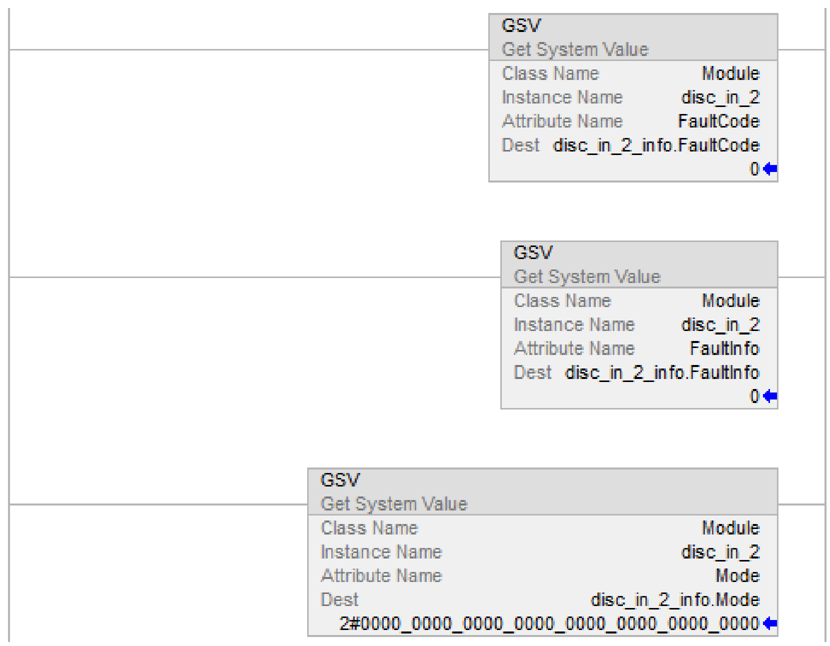
When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

You must test and confirm that the instructions do not change data that you do not want to change.

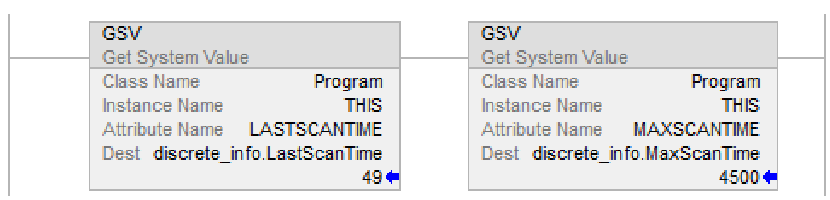
The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1



Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

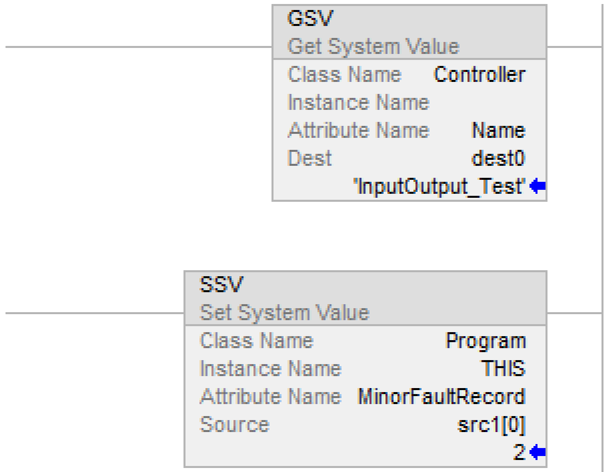
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

Get System Value (GSV) and Set System Value (SSV)

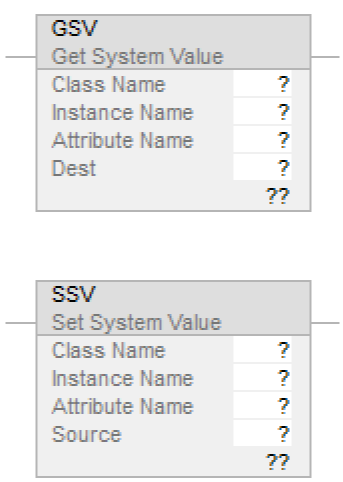
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram



Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

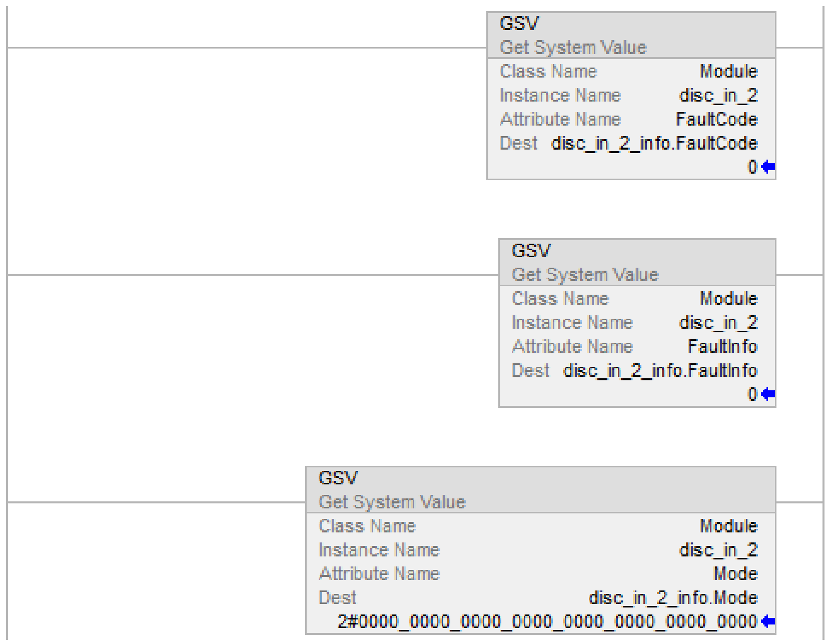
When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

You must test and confirm that the instructions do not change data that you do not want to change.

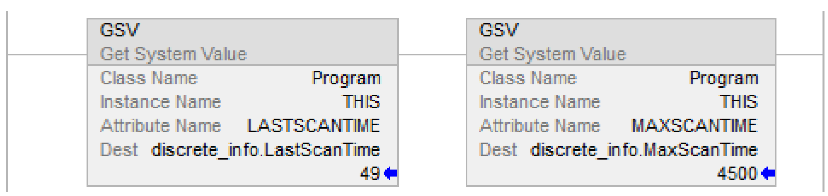
The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1



Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

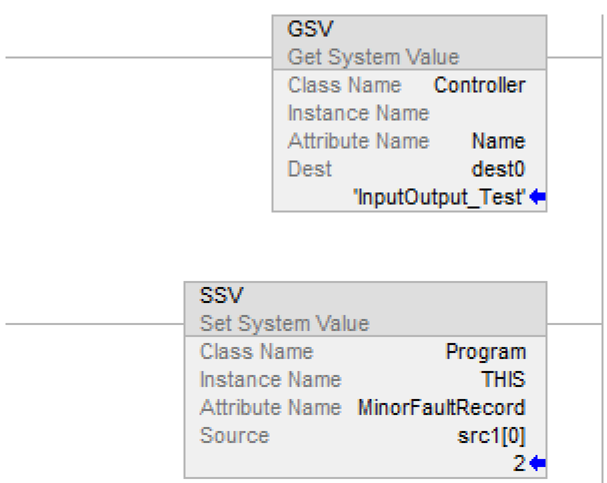
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

Get System Value (GSV) and Set System Value (SSV)

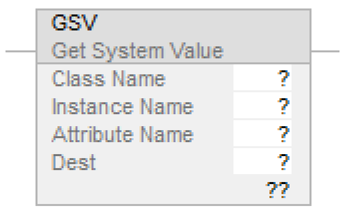
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

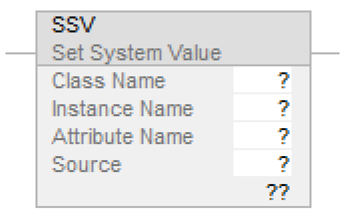
he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram





Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

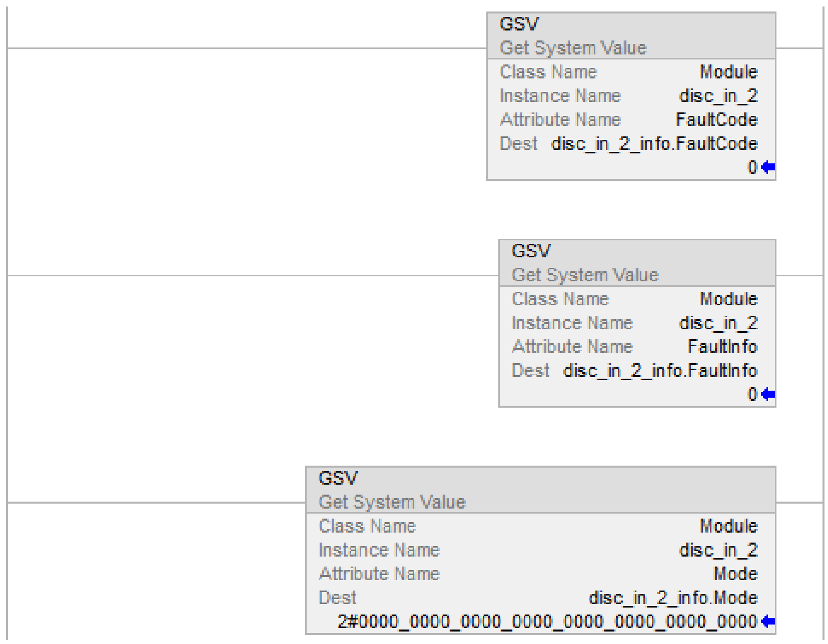
When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

You must test and confirm that the instructions do not change data that you do not want to change.

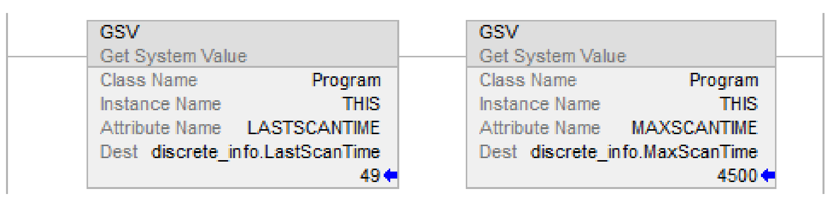
The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1



Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

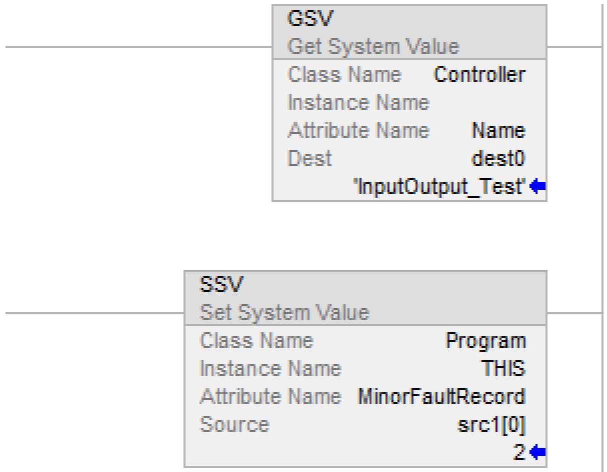
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

Get System Value (GSV) and Set System Value (SSV)

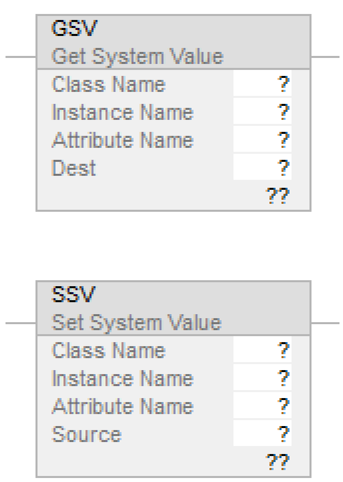
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram



Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

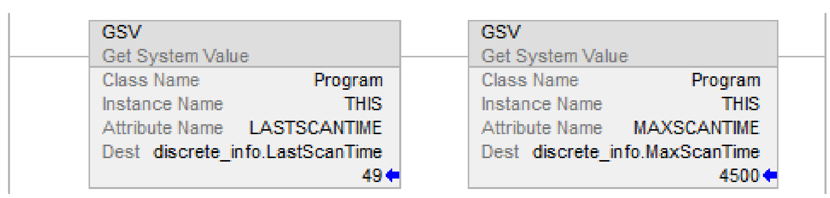
You must test and confirm that the instructions do not change data that you do not want to change.

The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1

Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

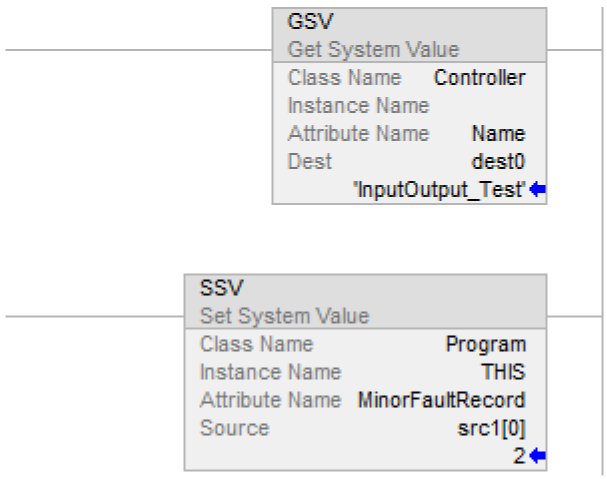
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

Get System Value (GSV) and Set System Value (SSV)

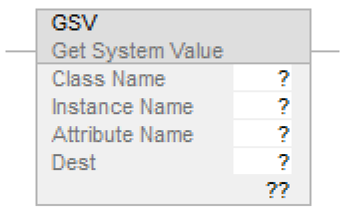
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

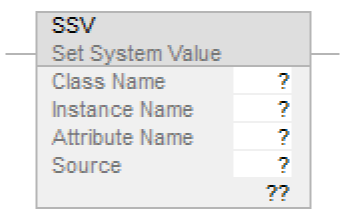
he GSV/SSV instructions get and set controller system data that is stored in objects.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

Available Languages

Ladder Diagram





Function Block

These instructions are not available in function block.

Structured Text

GSV(ClassName,InstanceName,AttributeName,Dest)

SSV(ClassName,InstanceName,AttributeName,Source)

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Class name		name	The name of object class
Instance name		name	The name of specific object, when object requires name
Attribute name		name	The attribute of object The data type depends on the attribute you select
Destination (GSV)	SINT INT DINT REAL structure TIME32 TIME DT LDT	tag	The destination for attribute data
Source (SSV)	SINT INT DINT REAL structure TIME32 TIME LTIME DT LDT	tag	The tag that contains data you want to copy to the attribute

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: When you use the GSV Instruction with the WallClock class and the CSTOffset attribute with the TIME32 data type, you must create the TIME32 data type tag a TIME32[2] array tag.

Description

The GSV/SSV instructions get and set controller status data that is stored in objects. The controller stores status data in objects. There is no status file, as in the PLC-5 processor.

When true, the GSV instruction retrieves the specified information and places it in the destination. When true, the SSV instruction sets the specified attribute with data from the source.

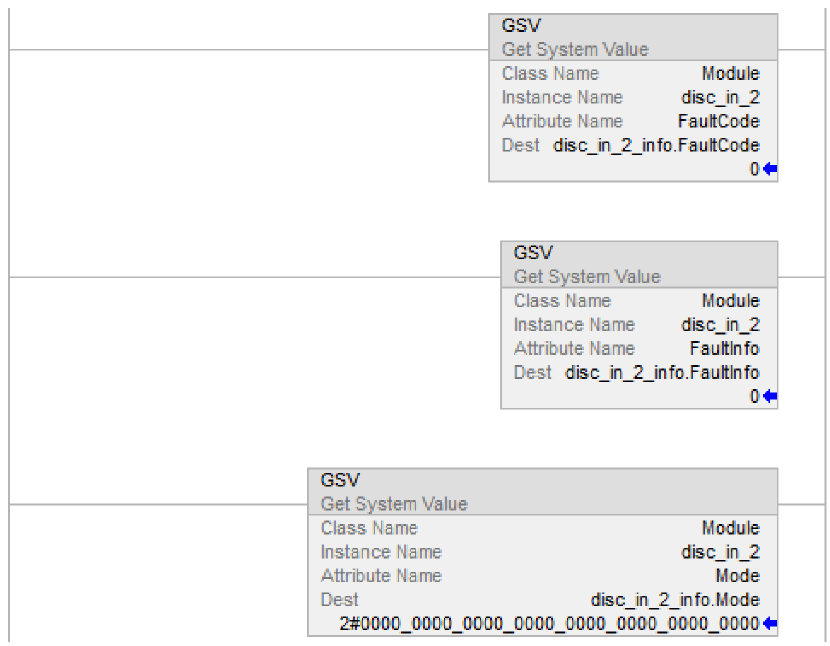
When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the attributes. For the SSV instruction, the software displays only those attributes you can set (SSV).

NOTE: CAUTION: Use the SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

You must test and confirm that the instructions do not change data that you do not want to change.

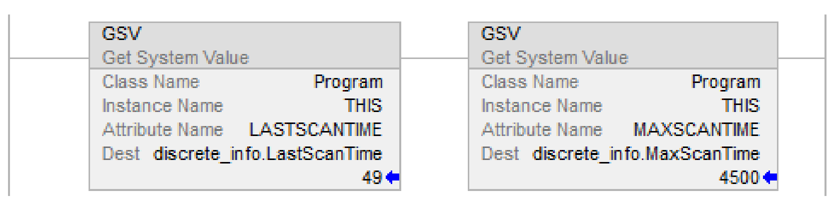
The SSV instructions write and the GSV instructions read past a member into other members of a tag. If the tag is too small, the instructions do not write or read the data. They log a minor fault instead.

Example 1



Member_A is too small for the attribute. So the GSV instruction writes the last value to Member_B.

Example 2



My_Tag is too small for the attribute. So the GSV instruction stops and logs a minor fault. The Destination tag remains unchanged.

GSV/SSV Objects define each object's attributes and their associated data types. For example, the MajorFaultRecord attribute of the Program object requires a DINT[11] data type.

Affects Math Status Flags

No.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
There is an invalid object address	4	5
The specified object that does not support GSV/SSV	4	6
There is an invalid attribute	4	6
There was not enough information supplied for an SSV instruction	4	6
The GSV destination was not large enough to hold the requested data	4	7

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

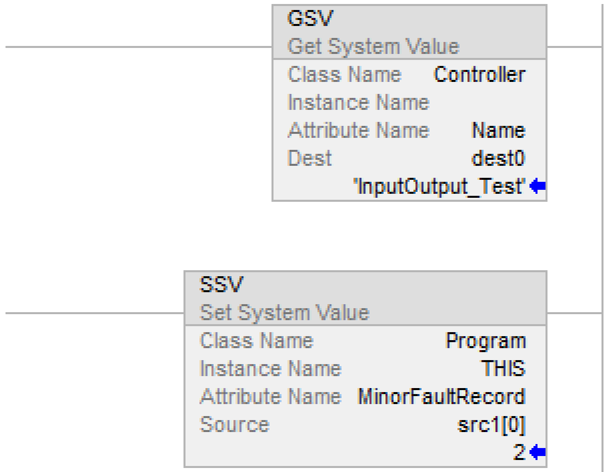
Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagrams table.
Normal Execution	See rung-condition-in is true in the Ladder Diagrams table.
Postscan	See Postscan in the Ladder Diagrams table.

Example

Ladder Diagrams



Structured Text

GSV (Program,THIS,LASTSCANTIME,dest1);

SSV (Program, THIS, MinorFaultRecord, src[0]);

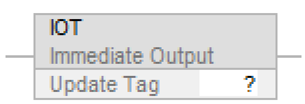
Immediate Output (IOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The IOT instruction immediately updates the specified output data (output tag of an I/O module or produced tag). The connection to the module must be open to enable the IOT instruction to execute.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

IOT (output_tag)

Operands

Ladder Diagram

Operand	TYPE	FORMAT	DESCRIPTION
Update Tag		Tag	Tag that contains data you want to copy to the attribute tag that you want to update; either: Output tag of an I/O module or Produced tag

Structured Text

The operands are the same as those for the ladder diagram IOT instruction.

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The IOT instruction overrides the requested packet interval (RPI) of an output connection and sends fresh data over the connection.

An output connection is a connection that is associated with the output tag of an I/O module or with a produced tag. If the connection is for a produced tag, the IOT instruction also sends the event trigger to the consuming controller. This allows the IOT instruction to trigger an event task in the consuming controller.

To use an IOT instruction and a produced tag to trigger an event task in a consumer controller, check the Programmatically (IOT Instruction) Send Event Trigger to Consumer checkbox on the Connection tab of the **Tag Properties** dialog box.

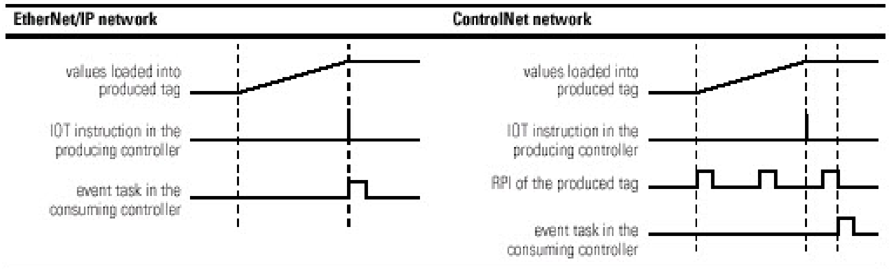
Tip: For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers , when controlling 5069 I/O over a remote network, an optimization is used to group module connections configured with the same RPI rate into one packet for sending over the network. If the IOT is used on one of these tags, the IOT may cause immediate update of some data tags for other modules that are configured at the same RPI and in the same backplane and are being grouped together with that tag. If this is not desirable, it can be avoided by making the RPI not exactly equal to the RPI other module connections.

Tip: When using this instruction with a ControlLogix Redundancy system, outputs controlled by this instruction may not be bumpless during a redundancy switchover.

The type of network between the controllers determines when the consuming controller receives the new data and event trigger via the IOT instruction.

Over this network	The consuming device receives the data and event trigger
Backplane	Immediately
EtherNet/IP	Immediately
ControlNet	Within the actual packet interval (API) of the consumed tag (connection)

The following diagrams compare the receipt of data via an IOT instruction over EtherNet/IP and ControlNet networks.



Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction updates the connection of the specified tag resets the RPI timer of the connection.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	N/A
Normal execution	See rung-condition-in is true in the Ladder Diagram
Postscan	N/A

Example

When the IOT instruction executes, it immediately sends the values of the Local:5:O tag to the output module.

Ladder Diagram

Structured Text

IOT (Local:5:0);

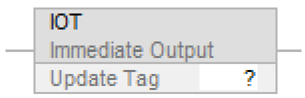
Immediate Output (IOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The IOT instruction immediately updates the specified output data (output tag of an I/O module or produced tag). The connection to the module must be open to enable the IOT instruction to execute.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

IOT (output_tag)

Operands

Ladder Diagram

Operand	TYPE	FORMAT	DESCRIPTION
Update Tag		Tag	Tag that contains data you want to copy to the attribute tag that you want to update; either: Output tag of an I/O module or Produced tag

Structured Text

The operands are the same as those for the ladder diagram IOT instruction.

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The IOT instruction overrides the requested packet interval (RPI) of an output connection and sends fresh data over the connection.

An output connection is a connection that is associated with the output tag of an I/O module or with a produced tag. If the connection is for a produced tag, the IOT instruction also sends the event trigger to the consuming controller. This allows the IOT instruction to trigger an event task in the consuming controller.

To use an IOT instruction and a produced tag to trigger an event task in a consumer controller, check the Programmatically (IOT Instruction) Send Event Trigger to Consumer checkbox on the Connection tab of the **Tag Properties** dialog box.

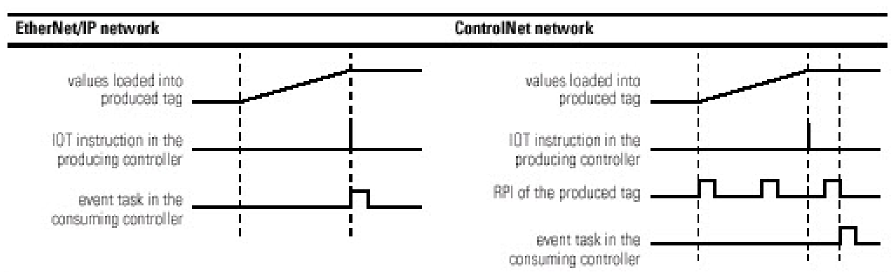
Tip: For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers , when controlling 5069 I/O over a remote network, an optimization is used to group module connections configured with the same RPI rate into one packet for sending over the network. If the IOT is used on one of these tags, the IOT may cause immediate update of some data tags for other modules that are configured at the same RPI and in the same backplane and are being grouped together with that tag. If this is not desirable, it can be avoided by making the RPI not exactly equal to the RPI other module connections.

Tip: When using this instruction with a ControlLogix Redundancy system, outputs controlled by this instruction may not be bumpless during a redundancy switchover.

The type of network between the controllers determines when the consuming controller receives the new data and event trigger via the IOT instruction.

Over this network	The consuming device receives the data and event trigger
Backplane	Immediately
EtherNet/IP	Immediately
ControlNet	Within the actual packet interval (API) of the consumed tag (connection)

The following diagrams compare the receipt of data via an IOT instruction over EtherNet/IP and ControlNet networks.



Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction updates the connection of the specified tag resets the RPI timer of the connection.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	N/A
Normal execution	See rung-condition-in is true in the Ladder Diagram
Postscan	N/A

Example

When the IOT instruction executes, it immediately sends the values of the Local:5:0 tag to the output module.

Ladder Diagram

Structured Text

IOT (Local:5:0);

Reference (REF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

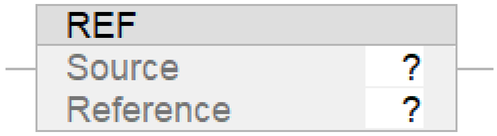
The Reference (REF) Instruction associates a reference with the address of an axis or coordinate system concrete tag.

This is a transitional instruction. Follow these steps when using it:

- - In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
  - - In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Instructions

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

REF(Source, Reference);

Operands

Operand	Type	Format	Description
Source	AXIS_CIP_DRIVE AXIS_CONSUMED AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_VIRTUAL COORDINATE_SYSTEM	Immediate Tag	Name of the axis or coordinate system to reference.
Reference	REF_TO_AXIS_CIP_DRIVE REF_TO_AXIS_CONSUMED REF_TO_AXIS_GENERIC_DRIVE REF_TO_AXIS_SERVO REF_TO_AXIS_SERVO_DRIVE REF_TO_AXIS_VIRTUAL REF_TO_COORDINATE_SYSTEM	Tag	Name of the reference to be populated.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.

Execution

Condition/State	Action Taken
Prescan	The instruction uses the Source address to populate the Reference.

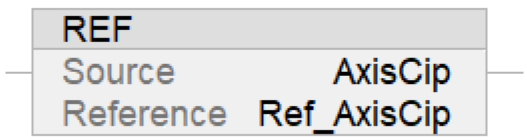
Condition/State	Action Taken
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in. The instruction uses the Source address to populate the Reference.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The instruction uses the Source address to populate the Reference.
Postscan	The instruction uses the Source address to populate the Reference.

Error Codes

None specific to this instruction.

Example

Relay Ladder



Structured Text

REF(AxisCip, Ref_AxisCip);

Access System Values

This procedure will help you to get or use status information about your Logix 5000 controller.

If you want to:	Refer to this help topic:
use specific key words in your logic to monitor specific events	Monitor Status Flags on page 281
get or set system values	Get and Set System Data on page 267
get information about the memory of the controller	Determine Controller Memory Information on page 259

Access the AddOnInstructionDefinition Object

The **AddOnInstructionDefinition** object lets you customize instructions for sets of commonly-used logic, provides a common interface to this logic, and provides documentation for the instruction.

For details, see the Controllers Add-On Instructions Programming Manual, publication 1756-PM010.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
LastEditDate	LINT	GSV	None	Timestamp of the last edit to an Add-On Instruction.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
MajorRevision	DINT	GSV	None	Major revision number of the Add-On Instruction.
MinorRevision	DINT	GSV	None	Minor revision number of the Add-On Instruction.
Name	String	GSV	GSV	Name of the Add-On Instruction.
RevisionExtendedText	String	GSV	None	Text describing the revision of the Add-On Instruction.
SafetySignature ID	DINT	GSV	None	In a safety project, the ID number, date, and timestamp of an Add-On Instruction definition.
SignatureID	DINT	GSV	None	32-bit identification number of an Add-On Instruction definition.
Vendor	String	GSV	None	Vendor that created the Add-On Instruction.

Access the ALARMBUFFER object

The ALARMBUFFER object is part of the Publisher/Subscriber infrastructure. The Publisher/Subscriber infrastructure is part of the Logix controller communication subsystem. The Logix controller communication subsystem implements Publisher/Subscriber messaging patterns for CIP, which lets other devices receive messages sent by the controller subsystem. Currently, Digital and Analog Alarms and Batch Equipment Phase subsystems use the Publisher/Subscriber Infrastructure to deliver messages through CIP to subscribing applications.

Use the ALARMBUFFER object to help you determine the existence of connections to the Publisher/Subscriber subsystem and their status. An AlarmBuffer object instance exists for every subscribing application. This means that an AlarmBuffer object may exist at one point in time, but not exist at another time. For this reason, a Get System Value (GSV) instruction returns a status as part of the destination tag (INT[0].0). When the status bit is zero, this most likely means that the AlarmBuffer object no longer exists.

Attribute	Data Type	Instruction	Description	
AlarmBufferInstance	DINT[n]	GSV	Returns the AlarmBuffer object IDs.	
			DINT[0]	Number of AlarmBuffer objects.
			DINT[1...(n-1)]	AlarmBuffer object IDs.

Attribute	Data Type	Instruction	Description	
			If the number of AlarmBuffer objects is greater than n-1, only the IDs of the first (n-1) objects are returned. You do not have to specify an AlarmBuffer Instance ID for this attribute.	
AlarmBufferStatus	INT[2]	GSV	Returns the status of the specified AlarmBuffer object. You have to specify the AlarmBuffer Instance ID to get the status of that individual instance.	
			INT[0].0	1-AlarmBufferStatus Attribute is valid. 0-AlarmBufferStatus Attribute is invalid.
			INT[1]	AlarmBuffer Status Attribute value.
			The Status attribute contains the following:	
			INT[1].0	1-Multi-message packets enabled. 0-Multi-message packets disabled.
			INT[1].1	1-Buffer is enabled. 0-Buffer is disabled.
			INT[1].2	1-Data stored in the buffer. 0-Buffer is empty.
			INT[1].3	1-Buffer is full. 0-Buffer is not full.
			INT[1].4	1-Initialization Status messages WILL NOT be sent (at subscription time and on Redundancy switchover). 0-Initialization Status messages WILL be sent.
			All other bits are reserved and are set to 0.	
BufferSize	INT[2]	GSV	Returns the buffer size (in kB) of the specified AlarmBuffer Object. You have to specify the Alarm Buffer Instance ID to get the buffer size of that individual instance.	
			INT[0].0	1-BufferSize Attribute is valid.

Attribute	Data Type	Instruction	Description	
				0-BufferSize Attribute is invalid.
			INT[1]	Buffer Size Attribute value.
BufferUsage	INT[2]	GSV	Returns the percentage of buffer space used by the specified AlarmBuffer Object. You have to specify the AlarmBuffer Instance ID to get the buffer usage value of that individual instance.	
			INT[0].1	1-BufferUsage Attribute is valid. 0-BufferUsage Attribute is invalid.
			INT[1]	BufferUsage Attribute value.
SubscriberName	STRING	GSV	Returns the subscriber name of the specified AlarmBuffer object. You have to specify the AlarmBuffer Instance ID to get the subscriber name of that individual instance. Any string type can be referenced as a destination tag. If the Subscriber Name cannot fit into the provided destination tag string, then only the part of the name that can fit in the destination tag is provided by the instruction. If the AlarmBuffer object instance specified by the instance ID does not exist when the instruction is called, then the string length (.LEN member) is set to zero. Note that if no subscriber name is provided when AlarmBuffer object is created by a subscriber, then the subscriber name attribute is set to a device serial number associated with a connection through which the Create service on the AlarmBuffer object was called.	

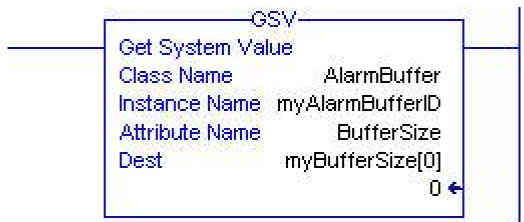
GSV Instruction Example

Your program can contain a GSV instruction to obtain the list of current AlarmBufferInstances in the controller. This instruction will return the total count of alarm buffer objects currently present in the controller (DINT[0]) along with the associated AlarmBuffer object Instance ID (DINT[1] – DINT[n-1]) for each AlarmBuffer object that is present in the controller. The GSV instruction displays the value of the number of AlarmBuffer objects (DINT[0]) under the Dest (destination) tag name.

Your program can use the AlarmBuffer object Instance ID to obtain information related to a specific instance of the AlarmBuffer object that is present in the controller. A status word (INT[0]), indicating valid or invalid data, is returned in the destination tag for the AlarmBufferStatus, BufferSize, and BufferUsage attributes, as the alarm buffer objects can be created and deleted at any time. The returned value is in (INT[1]) when the Attribute Name equals AlarmBufferStatue, BufferSize, or BufferUsage. The returned value is the subscriber name when the Attribute Name is SubscriberName. No status is returned for the SubsriberName attribute.

Ladder Diagram

Following is an example of the GSV instruction retrieving the AlarmBuffer object IDs.



Although the GSV of the AlarmBufferInstances returns the values into an array, you cannot use the array address to get attribute values for that instance. You must copy or move the value in myAlarmBufferInstances[x], (where x = 1, 2, 3,...) to a direct (unindexed) tag like the myAlarmBufferID shown in the following illustration.

Following is an example of the GSV instruction retrieving the buffer size of the AlarmBuffer object.

The number that is displayed under the Dest (destination) tag name is the valid or invalid bit value when the Attribute Name is AlarmBufferStatus, BufferSize or BufferUsage.

Structured Text

Following is an example of the GSV instruction retrieving the AlarmBuffer object IDs.

```
GSV(AlarmBuffer, AlarmBufferInstances, myAlarmBufferInstances[0]);
```

Following is an example of the GSV instruction retrieving the AlarmBuffer Object.

```
GSV(AlarmBuffer, myAlarmBufferID, BufferSize, myBufferSize[0]);
```

Access the Axis object

The AXIS object provides status information about an axis. Specify the axis tag name to determine which AXIS object you want.

For more information about the AXIS object, see the *SERCOS and Analog Motion Configuration and Startup User Manual*, publication MOTION-UM001.

When an attribute is marked with an asterisk (*), it means that the attribute is located in both the controller and in the motion module.When you use an SSV instruction to write one of these values, the controller automatically updates the copy in the module. However, this process is not immediate. The axis status tag, ConfigUpdateInProcess is provided to indicate when this process is complete.

For example, if you perform an SSV to the PositionLockTolerance, ConfigUpdateInProcess of the Axis tag is set until an update to the module is successful. Therefore, the logic following the SSV could wait on this bit resetting before continuing in the program.

Attribute	Data Type	Instruction	Description
* AccelerationFeedForwardGain	REAL	GSV SSV	The torque command output % necessary to generate the commanded acceleration.
ACStopMode	SINT	GSV SSV	The type of stop to perform your axis • 0 = fast stop • 1 = fast shutdown • 2 = hard shutdown
ActualPosition	REAL	GSV	The actual position in position units of your axis.
ActualVelocity	REAL	GSV	The actual velocity of your axis in position units/second.
AnalogInput1	REAL	GSV SSV	This attribute applies only to an axis associated Analog Input 2, a Kinetix7000 Drive. This attribute with an interger range of +/-16384, represents the analog value of an analog device connected to the Kinetix7000 drive's analog input(s). These inputs are useful for web/converting applications with load cell (measuring web force on a roller) or dancer (measuring web force/position directly), which can be directly connected to the drive controlling the web.
AverageVelocity	REAL	GSV	The average velocity of your axis in position units/second.
AverageVelocityTimebase	REAL	GSV SSV	The timebase in seconds of the average velocity of your axis.
AxisConfigurationState	SINT	GSV	The state of the axis configuration • 0 - 126 = not yet configured • 127 = invalid consumed axis data due to incompatible revisions between produced and consumer • 128 = configured

Attribute	Data Type	Instruction	Description
			• 3 = waiting on reply • 4 = configured
Bandwidth	REAL	GSV SSV	The unity gain bandwidth (Hz) that the controller uses to calculate the gains for a Motion Apply Axis Tuning (MAAT) instruction.
C2CConnectionInstance	DINT	GSV	The connection instance of the controller producing the axis data.
C2CMapTableInstance	DINT	GSV	The map instance of the controller producing the axis data.
CommandPosition	REAL	GSV	The command position of your axis in position units.
CommandVelocity	REAL	GSV	The command velocity of your axis in position units.
ConversionConstant	REAL	GSV SSV	The conversion factor used to convert from your units to feedback counts in counts/position unit.
DampingFactor	REAL	GSV SSV	The value used in calculating the maximum position servo bandwidth during the execution of the Motion Run Axis Tuning (MRAT) instruction.
*DriveFaultAction	SINT	GSV SSV	The operation performed when a drive fault occurs. • 0 = shutdown the axis • 1 = disable the drive • 2 = stop the commanded motion • 3 = change the status bit only
DynamicsConfigurationBits	DINT	GSV SSV	Revision 16 improved how the controller handles changes to an S-curve profile. Do you want to return to revision 15 or earlier behavior for S-curves? NO — Leave these bits ON (default). YES — Turn OFF one or more of these bits:
			To turn off this change / **Turn off this bit**
			Reduced S-curve Stop Delay This change applies to the Motion Axis Stop (MAS) instruction. / 0

Attribute	Data Type	Instruction	Description	
			It lets you use a higher deceleration jerk to stop an accelerating axis more quickly. The controller uses the deceleration jerk of the stopping instruction if it is more than the current acceleration jerk.	
			Reduced S-curve Velocity Reversals Before revision 16, you could cause an axis to momentarily reverse direction if you decreased the deceleration jerk while the axis was decelerating. This typically happened if you tried to restart a jog or move with a lower deceleration rate while the axis was	1

Attribute	Data Type	Instruction	Description
			stopping. This change prevents the axis from reversing in those situations.
			Reduced S-curve Velocity Overshoots 2 You can cause an axis to overshoot its programmed speed if you decrease the acceleration jerk while the axis is accelerating. This change keeps to overshoot to no more than 50% of the programmed speed.
*FeedbackFaultAction	SINT	GSV SSV	The operation performed when an encoder loss fault occurs. • 0 = shutdown the axis • 1 = disable the drive • 2 = stop the commanded motion • 3 = change the status bit only
*FeedbackNoiseFaultAction	SINT	GSV SSV	The operation performed when an encoder noise fault occurs. • 0 = shutdown the axis • 1 = disable the drive • 2 = stop the commanded motion • 3 = change the status bit only
*FrictionCompensation	REAL	GSV SSV	The fixed output level in volts used to compensate for static friction.

Attribute	Data Type	Instruction	Description
GroupInstance	DINT	GSV SSV	The instance number of the motion group that contains your axis.
HardOvertravelFaultAction	SINT	GSV SSV	• 0 = shutdown • 1= disable the drive • 2 = stop motion • 3 = status only
HomeConfigurationBits	DINT	GSV SSV	The motion configuration bits for your axis. • 0 = home direction • 1 = home switch normally closes • 2 = home marker edge negative
HomeMode	SINT	GSV SSV	The homing mode for your axis. • 0 = passive homing • 1 = active homing (default) • 2 = absolute
HomePosition	REAL	GSV SSV	The homing position of your axis in position units.
HomeReturnSpeed	REAL	GSV SSV	The homing return speed of your axis in position units/second.
HomeSequence	SINT	GSV SSV	The homing sequence type for your axis. • 0 = immediate homing • 1 = switch homing • 2 = marker homing • 3 = switch-marker homing (default)
HomeSpeed	REAL	GSV	The homing speed of your axis in position units/second.
Instance	DINT	GSV	The instance number of the axis.
InterpolatedActualPosition	REAL	GSV	For time-based position captures, this attribute provides the interpolated actual axis position. The position is specified in position units, and is based on the value of the InterpolationTime attribute. To interpolate an actual axis position, use an SSV instruction to set the InterpolationTime attribute.
InterpolatedCommandPosition	REAL	GSV	For time-based position captures, this attribute provides the interpolated command axis position. The position is specified in position units, and is based on the value of the InterpolationTime attribute.

Attribute	Data Type	Instruction	Description
			To interpolate a command axis position, use an SSV instruction to set the InterpolationTime attribute.
InterpolationTime	DINT	GSV SSV	Use this attribute to provide a reference for time-based position captures. To interpolate a position, use an SSV instruction to set the InterpolationTime attribute. The controller then updates the following attributes: • InterpolatedActualPosition • InterpolatedCommandPosition To supply a value for InterpolationTime, you can use any event that produces a CST timestamp, such as: • RegistrationTime attribute • timestamp of a digital output The InterpolationTime attribute uses only the lower 32 bits of a CST timestamp.
MapTableInstance	DINT	GSV	The I/O map instance of the servo module.
MasterOffset	REAL	GSV	Position offset that is currently applied to the master of a position cam. Specified in position units of the master axis.
MaximumAcceleration	REAL	GSV SSV	The maximum acceleration of your axis in position units/second^2.
MaximumDeceleration	REAL	GSV SSV	The maximum deceleration of your axis in position units/second^2.
*MaximumNegativeTravel	REAL	GSV SSV	The maximum negative travel limit in position units.
*MaximumPositiveTravel	REAL	GSV SSV	The maximum positive travel limit in position units.
MaximumSpeed	REAL	GSV SSV	The maximum speed of your axis in position units/second.
ModuleChannel	SINT	GSV	The channel of your servo module.
MotionStatusBits	DINT	GSV	The motion status bits for your axis. (In the AXIS structure, this is the MotionStatus member.

Bit	Bit name	Meaning
0	AccelStatus	acceleration
1	DecelStatus	deceleration
2	MoveStatus	move

Attribute	Data Type	Instruction	Description		
			3	JogStatus	jog
			4	GearingStatus	gear
			5	HomingStatus	home
			6	StoppingStatus	stop
			7	AxisHomedStatus	homed status
			8	PositionCamStatus	position cam
			9	TimeCamStatus	time cam
			10	PositionCamPendingStatus	position cam pending
			11	TimeCamPendingStatus	time cam pending
			12	GearingLockStatus	gearing lock
			13	PositionCamLockStatus	position cam lock
			14	MasterOffsetMoveStatus	master offset move
			15	CoordinatedMotionStatus	coordinate motion
			16	TransformStateStatus	transform state
			17	ControlledByTransformStatus	control by transform
*OutputLPFilterBandwidth	REAL	GSV SSV	The bandwidth (Hz) of the servo low-pass digital output filter.		
*OutputLimit	REAL	GSV SSV	The value in volts of the maximum servo output voltage of your axis.		
*OutputOffset	REAL	GSV SSV	The value in volts used to offset the effects of the cumulative offsets of the servo module DAC output and the servo drive input.		
PositionError	REAL	GSV	The difference between the actual and command position of an axis.		
*PositionErrorTolerance	REAL	GSV SSV	The amount of position error in position units that the servo tolerates before issuing a position error fault.		

Attribute	Data Type	Instruction	Description	
PositionIntegratorError	REAL	GSV	The sum of the position error for an axis in position units.	
*PositionIntegralGain	REAL	GSV SSV	The value (1/msec^2) used to achieve accurate axis positioning despite disturbances such as static friction and gravity.	
*PositionLockTolerance	REAL	GSV SSV	The amount of position error in position units that the servo module tolerates when giving a true position locked status indication.	
*PositionProportionalGain	REAL	GSV SSV	The value (1/msec) the controller multiples with the position error to correct for the position error.	
PositionServoBandwidth	REAL	GSV SSV	The unity gain bandwidth that the controller uses to calculate the gains for a Motion Apply Axis Tuning (MAAT) instruction.	
*PositionUnwind	DINT	GSV SSV	The value used to perform the automatic unwind of the rotary axis in counts/revolution.	
ProcessStatus	INT	GSV	The status of the last Motion Run Hookup Diagnostic (MRHD) instruction.	
			Value	**Meaning**
			0	test process successful
			1	test in progress
			2	test process aborted by the user
			3	test exceeded 2-second timeout
			4	test process failed due to servo fault
			5	insufficient test increment
ProgrammedStopMode	SINT	GSV SSV	The type of stop to perform on your axis.	
			Value	**Meaning**
			0	fast stop
			1	fast shutdown
			2	hard shutdown
Registration1Position	REAL	GSV	The registration position for your axis in position units.	
RegistrationTime	DINT	GSV	You can use this attribute to supply a timestamp for time-based position captures: • the RegistrationTime attribute contains the lower 32 bits of the CST timestamp of an axis registration event • The CST timestamp is measured in microseconds	

Attribute	Data Type	Instruction	Description
			• To interpolate a position based on an axis registration event: ◦ Use a GSV instruction to get the value of the RegistrationTime attribute. ◦ Use an SSV instruction to set the InterpolationTime attribute to the value of the RegistrationTime attribute.
RotaryAxis	SINT	GSV Tag	0 = Linear 1 = Rotary When the Rotary Axis attribute is set true (1), it lets the axis unwind. This gives infinite position range by unwinding the axis position whenever the axis moves through a complete physical revolution. The number of encoder counts per physical revolution of the axis is specified by the Position Unwind attribute. For Linear operation, the counts don't roll over. They are limited to +/- 2 billion.
ServoFaultBits	DINT	GSV	The servo fault bits for your servo loop. (In the AXIS structure, this is the AxisEvent member.)

Bit	Bit name	Meaning
0	PosSoftOvertravelFault	positive overtravel fault
1	NegSoftOvertravelFault	negative overtravel fault
2	NegSoftOvertravelFault	position error fault
3	FeedbackFault	encoder channel A loss fault
4	FeedbackFault	encoder channel B loss fault
5	FeedbackFault	encoder channel Z loss fault
6	FeedbackNoiseFault	encoder noise fault
7	DriveFault	drive fault
8	ModuleSyncFault	synchronous connection fault
9	ModuleHardwareFault	servo hardware fault

Attribute	Data Type	Instruction	Description
ServoOutputLevel	REAL	GSV	The output voltage level in volts for your axis servo loop.
ServoStatusBits	DINT	GSV	The status bits for your servo loop. (In the AXIS structure, this is the ServoStatus member.)

Bit	Bit name	Meaning

Attribute	Data Type	Instruction	Description		
			0	ServoActionStatus	servo action
			1	DriveEnableStatus	drive enable
			2	OutputLimitStatus	output limit
			3	PositionLockStatus	position lock
			13	TuneStatus	tuning process
			14	ProcessStatus	test diagnostic
			15	ShutdownStatus	axis shutdown
*SoftOvertravelFaultAction	SINT	GSV SSV	The operation performed when a soft overtravel fault occurs. 0 = shutdown the axis 1 = disable the drive 2 = stop the commanded motion 3 = change the status bit only		
StartActualPosition	REAL	GSV	The actual position in position units of your axis when new commanded motion starts for the axis.		
StartCommandPosition	REAL	GSV	The command position in position units of your axis when new commanded motion starts for the axis.		
StartMasterOffset	REAL	GSV	The master offset when the last Motion Axis Move (MAM) instruction executed either of these types of moves: • AbsoluteMasterOffset • IncrementalMasterOffset Specified in position units of the master axis.		
StrobeActualPosition	REAL	GSV	The actual position in position units of an axis when the Motion Group Strobe Position (MGSP) instruction executes.		
StrobeCommandPosition	REAL	GSV	The command position in position units of an axis when the Motion Group Strobe Position (MGSP) instruction executes.		
StrobeMasterOffset	REAL	GSV	The master offset when the Motion Group Strobe Position (MGSP) instruction executes. Specified in position units of the master axis.		
TestDirectionForward	SINT	GSV	The direction of axis travel during the Motion Run Hookup Diagnostic (MRHD) instruction as seen by the servo module. 0 = negative (reverse) direction 1 = positive (forward) direction		
TuneAcceleration	REAL	GSV	The acceleration value in position units/sec^2 measured during the last Motion Run Axis Tuning (MRAT) instruction.		
TuneAccelerationTime	REAL	GSV	The acceleration time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction.		

Attribute	Data Type	Instruction	Description
TuneDeceleration	REAL	GSV	The deceleration value in position units/sec measured during the last Motion Run Axis Tuning (MRAT) instruction.
TuneDecelerationTime	REAL	GSV	The deceleration time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction.
TuneInertia	REAL	GSV	The inertia value in mV/Kcounts/second for the axis as calculated from the measurements the controller made during the last Motion Run Axis Tuning (MRAT) instruction.
TuneRiseTime	REAL	GSV	The axis rise time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction.
TuneSpeedScaling	REAL	GSV	The axis drive scaling factor in mV/Kcounts/sec measured during the last Motion Run Axis Tuning (MRAT) instruction.
TuneStatus	INT	GSV	The status of the last Motion Run Axis Tuning (MRAT) instruction. • 0 = tune process successful • 1 = tuning in progress • 2 = tune process aborted by the user • 3 = tune exceeded 2-second timeout • 4 = tune process failed due to servo fault • 5 = axis reached tuning travel limit • 6 = axis polarity set incorrectly • 7 = tune speed is too small to make measurements
TuningConfigurationBits	DINT	GSV SSV	The tuning configuration bits for your axis. • 0 = turning direction (0=forward, 1=reverse) • 1 = tune position error integrator • 2 = tune velocity error integrator • 3 = tune velocity feedforward bit • 4 = acceleration feedforward • 5 = tune velocity low-pass filter
TuningSpeed	REAL	GSV SSV	The maximum speed in position units/second initiated by the Motion Run Axis Tuning (MRAT) instruction.
TuningTravelLimit	REAL	GSV SSV	The travel limit used by the Motion Run Axis Tuning (MRAT) instruction to limit the action during tuning.
VelocityCommand	REAL	GSV	The current velocity reference in position units/second to the velocity servo loop for an axis.
VelocityError	REAL	GSV	The difference in position units/second between the commanded and actual velocity of a servo axis.
VelocityFeedback	REAL	GSV	The actual velocity in position units/second of your axis as estimated by the servo module.
*VelocityFeedforwardGain	REAL	GSV SSV	The velocity command output % necessary to generate the commanded velocity.
*VelocityIntegralGain	REAL	GSV SSV	The value (1/msec) that the controller multiplies with the VelocityIError value to correct the velocity error.

Attribute	Data Type	Instruction	Description
VelocityIntegratorError	REAL	GSV	The sum of the velocity error for a specified axis.
*VelocityProportionalGain	REAL	GSV SSV	The value (1/msec) that the controller multiplies with the VelocityError to correct the velocity error.
*VelocityScaling	REAL	GSV SSV	The value used to convert the output of the servo loop into the equivalent voltage to the drive.
VelocityServoBandwidth	REAL	GSV SSV	The bandwidth (Hz) of the drive as calculated from the measurements made during the last Motion Run Axis Tuning (MRAT) instruction.
WatchPosition	REAL	GSV	The watch position in position units of your axis.

Access the Controller object

The **Controller** object provides status information about controller execution.

Attribute	Data Type	Instruction	Description
Audit Value	DINT[2], LINT	GSV	The audit value is a unique value that is generated when a project is downloaded to the controller or loaded from removable storage. When a change is detected, this value is updated. To specify which changes are monitored, use the ChangesToDetect attribute.
ChangesToDetect	DINT[2], LINT	GSV, SSV	Used to specify which changes are monitored. When a monitored change occurs, the Audit Value is updated.
CanUseRPIFromProducer	DINT	GSV	Identifies whether to use the RPI specified by the producer. • 0. Do not use the RPI specified by the producer. • 1. Use the RPI specified by the producer.
ControllerLog Execution Modification Count	DINT	GSV SSV	Number of controller log entries that originate from a program/task properties

Attribute	Data Type	Instruction	Description
			change, an online edit, or a controller timeslice change. It can also be configured to include log entries originating from forces. The number is reset if RAM enters a bad state. The number is not capped at the largest DINT, and a rollover can occur.
ControllerLog TotalEntryCount	DINT	GSV SSV	Number of controller log entries since the last firmware upgrade. The number is reset if RAM enters a bad state. The number is capped at the largest DINT.
DataTablePad Percentage	INT	GSV	Percentage (0...100) of free data table memory set aside.
IgnoreArrayFaultsDuringPostScan	SINT	GSV SSV	Used to configure the suppression of selected faults encountered when an SFC action is postscanned. Only valid when SFCs are configured for automatic reset. • 0. This value does not not suppress faults during postscan execution. This is the default and recommended behavior. • 1. This value automatically suppresses major faults 4/20 (Array subscript too large) and 4/83 (Value out of range) while postscanning SFC actions. When a fault is suppressed, the controller uses an internal fault handler to automatically clear the fault. This causes the faulted instruction to be skipped, with execution resuming at the following instruction.

Attribute	Data Type	Instruction	Description
			Because the fault handler is internal, you do not have to configure a fault handler to get this behavior. In fact, even if a fault handler is configured, a suppressed fault will not trigger it.
InhibitAutomatic FirmwareUpdate	BOOL	GSV SSV	Identifies whether to enable the firmware supervisor. • 0. This value executes the firmware supervisor. • 1. This value does not execute the firmware supervisor.
KeepTestEditsOnSwitch over	SINT	GSV	Identifies whether to maintain test edits on controller switchover. • 0. This value automatically untests edits at switchover, • 1. This value continues testing edits at switchover.
Name	String	GSV	Name of the controller.
Redundancy Enabled	SINT	GSV	Identifies whether the controller is configured for redundancy. • 0. This value indicates the controller is not configured for redundancy. • 1. This value indicates the controller is configured for redundancy.
ShareUnused TimeSlice	INT	GSV SSV	Identifies how the continuous task and the background tasks shared any unused timeslice. • 0. This value indicates that the operating system does not give control to the continuous task even if background is complete.

Attribute	Data Type	Instruction	Description
			• 1. This value indicates that a continuous task runs even if the background tasks are complete. This is the default value. • 2. This value or greater logs a minor fault and leaves the setting unchanged.
TimeSlice	INT	GSV SSV	Percentage of available CPU (10-90) that is assigned to communications. This value cannot change when the keyswitch is in the Run position.

Access the ControllerDevice object

The **ControllerDevice** object identifies the physical hardware of the controller.

Attribute	Data Type	Instruction	Description
DeviceName	SINT[33]	GSV	ASCII string that identifies the marketing description of the controller. The first byte contains a count of the number of ASCII characters returned in the array string.
ProductCode	INT	GSV	Each value identifies the type of controller: 15 SoftLogix5800 40 1756-L1 43 1769-L20 44 1769-L30 49 PowerFlex® with DriveLogix5725 50 1756-L53 51 1756-L55 52 PowerFlex with DriveLogix5730 53 Studio 5000 Logix Emulate 54 1756-L61 55 1756-L62 56 1756-L63

Attribute	Data Type	Instruction	Description
			57 1756-L64 64 1769-L31 65 1769-L35E 67 1756-L61S 68 1756-L62S 69 1756-LSP 72 1768-L43 74 1768-L45 76 1769-L32C 77 1769-L32E 78 1769-L35C 79 1756-L60M03SE 80 1769-L35CR 85 1756-L65 86 1756-L63S 87 1769-L23E-QB1 88 1769-L23-QBFC1 89 1769-L23E-QBFC1 92 1756-L71 93 1756-L72 94 1756-L73 95 1756-L74 96 1756-L75 101 1768-L43S 102 1768-L45S 106 1769-L30ER 107 1769-L33ER 108 1769-L36ERM 109 1769-L30ER-NSE 110 1769-L33ERM 146 1756-L7SP 147 1756-L72S 148 1756-L73S 149 1769-L24ER-QB1B 150 1769-L24ER-QBFC1B 151 1769-L27ERM-QBFC1B 152 1769-L19ER-BB1B 153 1769-L16ER-BB1B 154 1769-L18ER-BB1B 155 1769-L18ERM-BB1B 156 1769-L30ERM 158 1756-L71S 164 1756-L81E 165 1756-L82E

Attribute	Data Type	Instruction	Description
			166 1756-L83E 167 1756-L84E 168 1756-L85E 171 1756-L8SP 176 1769-L30ERMS 177 1769-L33ERMS 178 1769-L36ERMS 186 5069-L46ERMW 188 5069-L310ER_NSE 189 5069-L306ERM 190 5069-L310ERM 191 1756-MPC 192 5069-L320ERM 193 5069-L330ERM 194 5069-L340ERM 195 5069-L350ERM 196 5069-L306ER 201 1756-L81ENSE 202 1756-L82ENSE 203 1756-L83ENSE 204 1756-L84ENSE 205 1756-L85ENSE 211 1756-L81ES 212 1756-L82ES 213 1756-L83ES 214 1756-L84ES 216 5069-L310ER 217 5069-L320ER 218 5059-L330ER 219 5069-L340ER 220 5069-L350ER 221 5069-L310ERMS2 222 5069-L320ERMS2 223 5069-L330ERMS2 224 5069-L340ERMS2 225 5069-L350ERMS2 226 5069-L306ERMS2 228 5069-L380ERM 229 5069-L380ERMS2 230 5069-L3100ERM 231 5069-L3100ERMS2 233 1769-L37ERMO 234 1769-L37ERMOS

Attribute	Data Type	Instruction	Description
			235 5069-L306ERS2 236 5069-L310ERS2 237 5069-L320ERS2 238 5069-L330ERS2 239 5069-L340ERS2 240 5069-L350ERS2 241 5069-L380ERS2 242 5069-L3100ERS2 243 5069-L306ERMS3 244 5069-L310ERMS3 245 5069-L320ERMS3 246 5069-L330ERMS3 247 5069-L340ERMS3 248 5069-L350ERMS3 249 5069-L380ERMS3 250 5069-L3100ERMS3 255 1769-L38ERM 256 1769-L38ERMS 257 1769-L37ERM 258 1769-L37ERMS 282 1756-L81EP 283 1756-L83EP 284 1756-L85EP 285 5069-L320ERP 286 5069-L340ERP 290 5069-L4100ERMW 291 5069-L450ERMW 292 5069-L4200ERMW 293 5069-L430ERMW 330 5015-AENFT *The product code list may not be complete.
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.

Attribute	Data Type	Instruction	Description
Status	INT	GSV	Device Status Bits 7…4 Meaning 0000 Reserved 0001 Flash update in progress 0010 Reserved 0011 Reserved 0100 Flash is bad 0101 Faulted modes 0110 Run 0111 Program Fault Status Bits 11…8 Meaning 0001 Recoverable minor fault 0010 Unrecoverable minor fault 0100 Recoverable major fault 1000 Unrecoverable major fault Controller Status Bits 13…12 Meaning 01 Keyswitch in run 10 Keyswitch in program 11 Keyswitch in remote 15…14 Meaning 01 Controller is changing modes 10 Debug mode if controller 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.

Access the CoordinateSystem object

The COORDINATESYSTEM object provides status information about motion coordinate system execution.

Attribute	Data Type	Instruction	Meaning
CoordinateMotionStatus	DINT	GSV SSV	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated.

Attribute	Data Type	Instruction	Meaning
AccelStatus	BOOL	GSV SSV	Sets when vector is accelerating. Clears when a blend is in process or when vector move is at speed or decelerating.
DecelStatus	BOOL	GSV SSV	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating or when move completes.
ActualPosToleranceStatus	BOOL	GSV SSV	Sets for Actual Tolerance termination type only. The bit is set after the following two conditions have been met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system's Actual Tolerance value. It remains set after the instruction completes. It is reset when a new instruction is started.
CommandPosToleranceStatus	BOOL	GSV SSV	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system's Command Tolerance value and remains set after the instruction completes. It is reset when a new instruction is started.
RobotJointsDirectionSenseBits	DINT	GSV SSV	Set of bits that define the Joints direction sense. By default, the Joints direction sense bits are zero. This indicates that the user convention is the same as the Rockwell Kinematics convention. If any of the Joints have the opposite convention to the Rockwell convention,

Attribute	Data Type	Instruction	Meaning
			set the corresponding Joints direction sense bit to 1.
StoppingStatus	BOOL	GSV SSV	The Stopping Status bit is cleared when the MCCM instruction executes.
MoveStatus	BOOL	GSV SSV	Sets when MCCM begins axis motion. Clears on the .PC bit of the last motion instruction or a motion instruction executes which causes a stop.
MoveTransitionStatus	BOOL	GSV SSV	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes which causes a stop. Indicates not on path.
MovePendingStatus	BOOL	GSV SSV	The move pending bit is set once a coordinated motion instruction is queued. Once the instruction has begun executing, the bit will be cleared, provided no subsequent coordinated motion instructions have been queued in the mean time.In the case of a single coordinated motion instruction, the status bit may not be detected by the user in the Logix Designer application since the transition from queued to executing is faster than the coarse update. The real value of the bit comes in the case of multiple instructions. As long as an instruction is in the instruction queue, the pending bit will

Attribute	Data Type	Instruction	Meaning
			be set. This provides the Logix Designer application programmer a means of stream-lining the execution of multiple coordinated motion instructions. Ladder logic containing coordinated motion instructions can be made to execute faster when the programmer allows instructions to be queued while a preceding instruction is executing. When the MovePendingStatus bit is clear, the next coordinated motion instruction can be executed (that is, setup in the queue).
MovePendingQueueFullStatus	BOOL	GSV SSV	Sets when the instruction queue is full. It clears when the queue has room to hold another new coordinated move instruction.
TransformSourceStatus	BOOL	GSV SSV	The coordinate system is the source of an active transform.
TransformTargetStatus	BOOL	GSV SSV	The coordinate system is the target of an active transform.
CoorMotionLockStatus	BOOL	GSV SSV	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit

Attribute	Data Type	Instruction	Meaning
			is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCS is initiated.
coordinateDefinition	DINT	GSV	Coordinate Definition for the coordinates in geometry
zeroAngleOffset4	REAL	GSV/SSV	Zero Angle Orientation for the fourth axis of non-Cartesian geometries.
zeroAngleOffset5	REAL	GSV/SSV	Zero Angle Orientation for the fifth axis of non-Cartesian geometries.
zeroAngleOffset6	REAL	GSV/SSV	Zero Angle Orientation for the sixth axis of non-Cartesian geometries.
linkLength3	REAL	GSV/SSV	Linear length of the wrist link of a robot.
ballScrewPitch	REAL	GSV/SSV	Pitch of SCARA Independent Coupled Screw.
ActiveToolFrameID	DINT	GSV/tag	Active Tool Identifier specified by user in the MCTO instruction.
MaxOrientationSpeed	REAL	GSV/SSV	Maximum speed of the orientation axes of the coordinate system.
MaxOrientationAccel	REAL	GSV/SSV	Maximum acceleration of the orientation axes of the coordinate system.
MaxOrientationDecel	REAL	GSV/SSV	Maximum deceleration of the orientation axes of the coordinate system.
ActiveWorkFrameID	REAL	GSV/Tag	Active work frame

Attribute	Data Type	Instruction	Meaning
SwingArmOffsetA3	REAL	GSV/SSV	The offset along the X-axis from the center of the bottom base plate to the Joint 4 frame for 5-axis Delta geometry.
SwingArmOffsetD3	REAL	GSV/SSV	The offset along the Z axis from the center of the bottom base plate to the Joint 4 frame for 5-Axis Delta geometry.
SwingArmOffsetA4	REAL	GSV/SSV	The offset along the X-axis J4 frame to the Joint 5 frame for 5-Axis Delta geometry.
SwingArmOffsetD4	REAL	GSV/SSV	The offset along the Z-axis J4 frame to the Joint 5 frame for 5-axis Delta geometry.
SwingArmOffsetD5	REAL	GSV/SSV	The offset along the Z-axis J5 frame to the EOA frame for 5-axis Delta geometry.
SwingArmCouplingRatioNumerator	INT, DINT	GSV/SSV	The ratio of the rotation axis to the tilt axis.
SwingArmCouplingRatioDenominator	INT, DINT	GSV/SSV	The ratio of the rotation axis to the tilt axis.
SwingArmCouplingDirection	INT, DINT	GSV/SSV	Relative direction of the coupled J4 rotational axis to the J5 tilt axis for Delta J1J2J3J4J5 Robot geometry.

Access the CST object

The coordinated system time (CST) object provides coordinated system time for the devices in one chassis.

Attribute	Data Type	Instruction	Description
CurrentStatus	INT	GSV	Current status of the coordinated system time. Each bit has a specific meaning: • 0. Timer hardware faulted. The internal timer hardware of the device is in a faulted state. • 1. Ramping enabled. The current value of the timer's lower 16+ bits

Attribute	Data Type	Instruction	Description
			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 master CST object via a system time update. • 4. Local network master. The CST object is the local network master time source. • 5. Relay mode. The CST object is acting in a time relay mode. • 6. Duplicate master detected. A duplicate local network time master is 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] TIME32[2] TIME	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

Attribute	Data Type	Instruction	Description
			request value, as reported in the CurrentStatus attribute.

Access the DF1 object

The DF1 object provides an interface to the DF1 communication driver.

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).
Diagnostic Counters	INT[19]	GSV	Array of diagnostic counters for the DF1 communication driver.

Word offset		DF1 point-to-point	DF1 slaveMaster
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

Word offset		DF1 point-to-point	DF1 slaveMaster
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

Duplicate Detection	SINT	GSV	Enables duplicate message detection. Each value has a specific meaning: • 0. Duplicate message detection disabled. • Non zero. Duplicate message detection enabled.
Embedded ResponseEnable	SINT	GSV	Enables embedded response functionality (point-to-point only). Each value has a specific meaning: • 0. Initiated only after one is received. This is the default. • 1. Enabled unconditionally.
EnableStoreFwd	SINT	GSV	Enables the store and forward behavior when receiving a message. Each value has a specific meaning: • 0. Do not forward message • Non zero. See the store and forward table when receiving a message. This is the default.
ENQTransmit Limit	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

Word offset		DF1 point-to-point	DF1 slaveMaster
			packets (slave only). Each value has a specific meaning: • 0. EOT suppression disabled (disabled). • Non zero. EOT suppression enabled.
ErrorDetection	SINT	GSV	Specifies the error-detection scheme. Each value has a specific meaning: • 0. BCC. This is the default. • 1. CRC.
MasterMessageTransmit	SINT	GSV	Current value of the master message transmission (master only). Each value has a specific meaning: • 0. Between station polls. This is the default. • 1. In poll sequence. This take the place of the station number of the master.
MaxStation Address	SINT	GSV	Current value (0 to 31) of the maximum node address on a DH-485 network. Default is 31.
NAKReceiveLimit	SINT	GSV	The number of NAKs received in response to a message before stopping transmission (point-to-point communication only). Valid value 0 to 127. Default is 3.
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 to 255. Default is 0.

Word offset		DF1 point-to-point	DF1 slaveMaster
PollingMode	SINT	GSV	Current polling mode (master only). Default setting is 1. Each value has a specific meaning: • 0. Message-based, but don't allow slaves to initiate messages. • 1. Message-based, but allow slaves to initiate messages. This is the default. • 2. Standard, single-message transfer per node scan. • 3. Standard, multiple-message transfer per node scan.
ReplyMessage Wait	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 to 65,535. Delay in counts of 20 msec periods. The default is 5 periods (100 msec).
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 to 32,767. Delay in counts of 20 msec periods. The default is 3000 periods (1 minute).
StationAddress	INT	GSV	Current station address of the port. Valid value 0 to 254. Default is 0.
TokenHoldFactor	SINT	GSV	Current value (1 to 4) of the maximum number of messages sent by this node before passing the token on a DH-485 network. Default is 1.
TransmitRetries	SINT	GSV	Number of times to resend a message without getting an acknowledgment (master and slave only). Valid value 0 to 127. Default is 3.
PendingACK Timeout	DINT	SSV	Pending value for the ACKTimeout attribute.

Word offset		DF1 point-to-point	DF1 slaveMaster
Pending Duplicate Detection	SINT	SSV	Pending value for the DuplicateDetection attribute.
Pending Embedded ResponseEnable	SINT	SSV	Pending value for the EmbeddedResponse attribute.
PendingEnable StoreFwd	SINT	SSV	Pending value for the EnableStoreFwd attribute.
PendingENQ TransmitLimit	SINT	SSV	Pending value for the ENQTransmitLimit attribute.
PendingEOT Suppression	SINT	SSV	Pending value for the EOTSuppression attribute.
PendingError Detection	SINT	SSV	Pending value for the ErrorDetection attribute.
PendingMaster Message Transmit	SINT	SSV	Pending value for the MasterMessageTransmit attribute.
PendingMax StationAddress	SINT	SSV	Pending value for the MaxStationAddress attribute.
PendingNAK ReceiveLimit	SINT	SSV	Pending value for the NAKReceiveLimit attribute.
PendingNormal PollGroupSize	INT	SSV	Pending value for the NormalPollGroupSize attribute.
PendingPolling Mode	SINT	SSV	Pending value for the PollingMode attribute.
PendingReply MessageWait	DINT	SSV	Pending value for the ReplyMessageWait attribute.
PendingSlavePollTimeout	DINT	SSV	Pending value for the SlavePollTimeout attribute.
PendingStation Address	INT	SSV	Pending value for the StationAddress attribute.
PendingToken HoldFactory	SINT	SSV	Pending value for the TokenHoldFactor attribute.
PendingTransmitRetries	SINT	SSV	Pending value for the TransmitRetries attribute.

Access the FaultLog object

The FaultLog object provides fault information about the controller.

Attribute	Data Type	Instruction	Description
MajorEvents	INT	GSV SSV	The number of major faults that occurred since this counter was reset.

Attribute	Data Type	Instruction	Description
MajorFaultBits	DINT	GSV SSV	Individual bits indicate the reason for the current major fault. Each bit has a specific meaning: 1 Power loss 3 I/O 4 Instruction execution (program) 5 Fault Handler 6 Watchdog 7 Stack 8 Mode change 11 Motion
MinorEvents	INT	GSV SSV	The number of minor faults that occurred since this counter was reset.
MinorFaultBits	DINT	GSV SSV	Individual bits indicate the reason for the current minor fault. Each bit has a specific meaning: 4 - Instruction execution (program) 6 - Watchdog 9 - Serial port 10 - Energy Storage Module (ESM), or Uninterruptable Power Supply (UPS) 20 - License/a required CodeMeter license is missing or missing.

Access the HardwareStatus object

The **HardwareStatus** object is used to obtain status information about the UPS, fans, and temperatures with GSV instructions for the CompactLogix 5480 controller projects. This object is supported in Ladder Diagram and Structured Text routines and in Add-On Instructions.

Attribute	Data Type		Instruction	Description
FanSpeeds	Array of:		GSV	Speed of the fans.
	Number of Fans	SINT		If the number of fans supported by the product is zero, then the device does not support fans.

Attribute	Data Type		Instruction	Description
	Fan Speed	SINT[9] for 2 fans: SINT[0] = Number of fans SINT[1-4] = Fan #1 speed SINT[5-8] = Fan #2 speed		RPM
FanStatus	Array of:		GSV	Indicates whether the fan is faulted.
	Number of Fan Status Indicators	SINT		If the number of fans supported by the product is zero, then the device does not support fan status.
	Fan Status	SINT[3] for 2 fans: SINT[0] = Number of fans SINT[1] = Fan #1 status SINT[2] = Fan #2 status		• 0. Fan is not faulted • 1. Fan is faulted
TemperatureFaultLevels	Array of:		GSV	The fault level in degrees Celsius
	Number of Temperature Fault Level	SINT		If the number of temperature fault level is zero, then the device does not support temperature fault levels.
	Temperature Fault Level	SINT[3] for 1 temperature sensor: SINT[0] = Number Temperature Fault Levels SINT[1-2] = Temperature Fault Level #1		Temperature in degrees Celsius
Temperatures	Array of:		GSV	Temperature values in degrees Celsius
	Number of Temperatures	SINT		If the number of temperatures supported by product is zero, then the device does not support temperatures.

Attribute	Data Type		Instruction	Description
	Temperature	SINT[3] for 1 temperature sensor: SINT[0] = Number of Temperatures SINT[1-2] = Temperature #1		Temperature in degrees Celsius
UPSBatteryFailure	SINT		GSV	Indicates whether the UPS battery has failed. • 0. The connected UPS battery has detected no faults. • 1. The connected UPS detected an issue with the connected battery.
UPSBuffering	SINT		GSV	Indicates whether the UPS is providing power from the battery. • 0. UPS is not providing power from the battery. • 1. UPS is providing power from the battery.
UPSInhibited	SINT		GSV	Requests UPS to remove power. • 0. The controller does not want power to be removed at this time. • 1. UPS is to stop providing power.
UPSReady	SINT		GSV	Indicates whether the UPS is ready based on: charged >= 85%, no wiring failure, input voltage sufficient, and inhibit signal is inactive. • 0. UPS not ready • 1. UPS ready

Attribute	Data Type	Instruction	Description
UPSSupported	SINT	GSV	Indicates whether the UPS is supported. • 0. Not supported • 1. Supported

Access the Message object

Access the Message object through the GSV/SSV instructions. Specify the message tag name to determine which Message object you want. The Message object provides an interface to setup and trigger peer-to-peer communications. This object replaces the MG data type of the PLC-5 processor.

Attribute	Data Type	Instruction	Description
ConnectionPath	SINT[82]	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 TIME32	GSV SSV	Requested packet rate of the connection.
MessageType	SINT	GSV SSV	Specifies the type of message. The value has a specific meaning: • 0. Not initialized
Port	SINT	GSV SSV	Indicates which port the message should be sent on. Each value has a specific meaning: • 1. Backplane. • 2. Serial port.
Timeout Multiplier	SINT	GSV SSV	Determines when a connection should be considered timed out and closed. Each value has a specific meaning: • 0. Connection times out in four times the update rate. This is the default. • 1. Connection times out in eight times the update rate. • 2. Connection times out in 16 times the update rate.

Attribute	Data Type	Instruction	Description
Unconnected Timeout	DINT TIME32	GSV SSV	Timeout in microseconds for all unconnected messages. The default is 30,000,000 microseconds (30 s).

Access the Module object

The Module object provides status information about a module. To select a particular Module object, set the Object Name operand of the GSV/SSV instruction to the module name. The specified module must be present in the I/O Configuration section of the controller organizer and must have a device name.

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. Each value has a specific 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

Attribute	Data Type	Instruction	Description
			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. • **16#9000. Firmware Updating.** Firmware supervisor is attempting to flash the module. • **16#A000. Configuring.** Controller is downloading configuration to the module.
FaultCode	INT	GSV	A number that identifies a module fault, if one occurs.
FaultInfo	DINT	GSV	Provides specific information about the Module object fault code.
Firmware SupervisorStatus	INT	GSV	Identifies current operating state of the firmware

Attribute	Data Type	Instruction	Description
			supervisor feature. Each value has specific meaning: • 0. Module updates are not being executed. • 1. Module updates are being executed.
ForceStatus	INT	GSV	Specifies the status of forces. Each bit has specific 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.
LEDStatus	INT	GSV	Specifies the current state of the I/O status indicator on the front of the controller.(1) Each value has a specific meaning: • 0. Status indicator 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. You do not enter an object name with this attribute because this attribute applies to the entire collection of modules.

Attribute	Data Type	Instruction	Description
Mode	INT	GSV SSV	Specifies the current mode of the Module object. Each bit has a specific 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.
Path	SINT Array	GSV	Specifies the path to the module being referenced. This is a new attribute starting in version 24 software. Each byte has a specific meaning: • 0-1. Length of the path in bytes. If 0, length of the SINT array is insufficient to hold the returned module path. If SINT array length is insufficient to hold the path, the array is zeroed out, and a minor fault is logged.

(1) The 1756-L7x controllers do not have a status indicator display on the front of the controller, but do use this functionality.

Related information

Module Faults: 16#0000 - 16#00ff on page 283
Module Faults: 16#0100 - 16#01ff on page 287
Module Faults: 16#0200 - 16#02ff on page 295
Module Faults: 16#0300 - 16#03ff on page 298
Module Faults: 16#0800 - 16#08ff on page 303
Module Faults: 16#fd00 - 16#fdff on page 303

Module Faults: 16#fe00 - 16#feff on page 305

Module Faults: 16#ff00 - 16#ffff on page 309

Access the Routine object

The Routine object provides status information about a routine. Specify the routine name to determine which Routine object that you want.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
Instance	DINT	GSV	GSV	Provides the instance number for this routine object. Valid values are 0 through 65,535.
Name	String	GSV	GSV	Name of the routine.
SFCPaused	INT	GSV	None	In an SFC routine, indicates whether the SFC is paused. Each value has a specific meaning: • 0. SFC is not paused. • 1. SFC is paused.
SFCResuming	INT	GSV SSV	None	In an SFC routine, indicates whether the SFC is resuming execution. Each value has a specific meaning: • 0. SFC is not executing. This attribute is automatically set to 0 at the end of a scan in which the chart was executed. • 1. SFC is executing. Step and action timers will retain their previous value if configured to do so. This attribute is automatically set to 1 on the

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
				first scan after a chart is no longer paused.

Access the Redundancy object

The REDUNDANCY object provides status information about the redundancy system.

For This Information	Get This Attribute	Data Type	GSV/ SSV	Description	
Redundancy status of the entire chassis	ChassisRedundancy State	INT	GSV	If	Then
				16#2	Primary with synchronized secondary
				16#3	Primary with disqualified secondary
				16#4	Primary with no secondary
				16#10	Primary that's locked for update
Redundancy state of the partner chassis	PartnerChassis RedundancyState	INT	GSV	If	Then
				16#8	Synchronized secondary
				16#9	Disqualified secondary with primary
				16#E	No partner
				16#12	Secondary that's locked for update
Redundancy status of the controller	ModuleRedundancy State	INT	GSV	If	Then
				16#2	Primary with synchronized secondary
				16#3	Primary with disqualified secondary
				16#4	Primary with no secondary

For This Information	Get This Attribute	Data Type	GSV/ SSV	Description	
				16#6	Primary with synchronizing secondary
				16#F	Primary that's locking for update
				16#10	Primary that's locked for update
Redundancy state of the partner	PartnerModule RedundancyState	INT	GSV	If	Then
				16#7	Synchronizing secondary
				16#8	Synchronized secondary
				16#9	Disqualified secondary with primary
				16#E	No partner
				16#11	Secondary that is locking for update
				16#12	Secondary that is locked for update
Results of the compatibility checks with the partner controller	CompatibilityResu lts	INT	GSV	If	Then
				0	Undetermined
				1	No compatible partner
				2	Fully compatible partner
Status of the synchronization (qualification) process	Qualification InProgress	INT	GSV	If	Then
				-1	Synchronization (qualification) is not in progress
				0	Unsupported
				1...999	For modules that can measure their completion percentage, the percent of synchronization (qualification) that is complete

For This Information	Get This Attribute	Data Type	GSV/ SSV	Description	
				50	For modules that cannot measure their completion percentage, synchronization (qualification) is in progress
				100	Synchronization (qualification) is complete
Keyswitch settings of the controller and its partner match or do not match	KeyswitchAlarm	DINT	GSV	If	Then
				0	One of the following is true: The keyswitches match No partner is present
				1	keyswitches do not match
Position of the keyswitch of the partner	PartnerKeyswitch	DINT	GSV	If	Then
				0	Unknown
				1	RUN
				2	PROG
				3	REM
Status of the minor faults of the partner (if the ModuleRedundancy State indicates a partner is present)	PartnerMinorFaults	DINT	GSV	This bit	Means this minor fault
				1	Power-up fault
				3	I/O fault
				4	Problem with an instruction (program)
				6	Periodic task overlap (watchdog)
				9	Problem with the serial port (not available for 1756-L7x projects)
				10	Low battery or issue with the

For This Information	Get This Attribute	Data Type	GSV/ SSV	Description	
					energy storage module
Mode of the partner	PartnerMode	DINT	GSV	If	Then
				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
In a pair of redundant chassis, identification of a specific chassis without regard to the state of the chassis	PhysicalChassisID	INT	GSV	If	Then
				0	Unknown
				1	Chassis A
				2	Chassis B
Slot number of the Redundancy module (for example, 1756-RM, 1756-RM2) in the chassis	SRMSlotNumber	INT	GSV		
Size of the last crossload Size of the last crossload if you had a secondary chassis	LastDataTransferSize	DINT	GSV	This attribute gives the size of data that was or would have been crossloaded in the last scan. The size in DINTs (4-byte words). You must configure the controller for redundancy. You don't need a secondary chassis. Is there a synchronized secondary chassis	
				YES	This gives the number of

For This Information	Get This Attribute	Data Type	GSV/ SSV	Description	
					DINTs that was crossloaded in the last scan.
				NO	This gives the number of DINTs that would have been crossloaded in the last scan
Size of the biggest crossload Size of the biggest crossload if you had a secondary chassis	MaxDataTransferSize	DINT	GSV SSV	The size in DINTs (4-byte words). You must configure the controller for redundancy. You don't need a secondary chassis. To reset this value, use an SSV instruction with a Source value of 0. Is there a synchronized secondary chassis?	
				YES	This gives the biggest number of DINTS that was crossloaded.
				NO	This gives the biggest number of DINTs that would have been crossloaded.

Access the Program object

The Program object provides status information about a program. Specify the program name to determine the Program object you want.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
DisableFlag	SINT	GSV SSV	None	Controls this program's execution. Each value has a specific meaning: • 0. Execution enabled. • Non zero. Execution disabled.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
	DINT	GSV	GSV	A non-zero value disables.
LastScanTime	DINT TIME32	GSV SSV	None	Time to execute program the last time it was executed. Time is in microseconds.
MaxScanTime	DINT TIME32	GSV SSV	None	Maximum recorded execution time for this program. Time is in microseconds.
MajorFault Record	DINT[11]	GSV SSV	GSV SSV	Records major faults for this program.
MinorFault Record	DINT[11]	GSV SSV	GSV SSV	Records minor faults for this program.
Name	String	GSV	GSV	Name of the program.

Tip: Rockwell Automation recommends creating a user-defined structure to simplify access to either Fault Record attribute:

Name	Data Type	Style	Description
Timestamp	LINT	Decimal	The time at which the fault occurred. Units are microseconds since Jan 1, 1970.
Type	INT	Decimal	Fault type (program, I/O, and so forth)
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)

Access the MotionGroup object

The MOTIONGROUP object provides status information about a group of axes for the servo module. Specify the motion-group tag name to determine which MOTIONGROUP object you want.

Attribute	Data Type	Instruction	Description
Alternate1UpdateMultiplier	SINT, INT, or DINT	GSV	The update period for axes that are associated with the Alternate 1 Update Schedule.
Alternate1UpdatePeriod	DINT	GSV	The update period for axes that are associated with the Alternate 1 Update Schedule. Value is product of Alternate 1 Update Period and Coarse Update Period.
Alternate2UpdateMultiplier	SINT, INT, or DINT	GSV	The update period for axes that are associated with the Alternate 2 Update Schedule.
Alternate2UpdatePeriod	DINT	GSV	The update period for axes that are associated with the Alternate 2 Update Schedule. The value is product of Alternate 1 Update Period and Coarse Update Period.
AutoTagUpdate	SINT, INT, or DINT	GSV SSV	Controls the automatic conversion and update of Motion Status attributes.
CoarseUpdatePeriod	DINT	GSV	The Coarse Update Period commonly referred to as the Base Update Period.
Cycle Start Time	DINT[2] DT LINT	GSV	This 64-bit value (ms) corresponds to the Timer Event that starts the update cycle.
INSTANCE	DINT	GSV	The instance number of this MOTION_GROUP object
MaximumInterval	DINT[2] TIME32[2] TIME	GSV SSV	The maximum interval between successive executions of this task.
MinimumInterval	DINT[2] TIME32[2] TIME	GSV	The minimum interval between successive executions of this task.
StartTime	DINT[2] DT LINT	GSV	The value of Wall Clock Time when the last execution of the task was started
TaskAverageIOTime	DINT TIME32	GSV SSV	The Average motion task input to output time, that is, the elapsed time from motion task

Attribute	Data Type	Instruction	Description
			start to send of connection data. (Time Constant = 250 CUP)
TaskAverageScanTime	DINT TIME32	GSV SSV	The average motion task scan time. (Time Constant = 250 CUP)
TaskLastIOTime	DINT TIME32	GSV	The last motion task input to output time, that is, the elapsed time from motion task start to send of connection data.
TaskLastScanTime	DINT TIME32	GSV	The last motion task scan time. (Elapsed Time)
TaskMaximumIOTime	DINT TIME32	GSV SSV	The maximum motion task input to output time, that is, the elapsed time from motion task start to send of connection data.
TaskMaximumScanTime	DINT TIME32	GSV SSV	The maximum motion task scan time. (Elapsed Time)
Time Offset	DINT[2] TIME32[2] TIME	GSV	The time offset value between Wall Clock Time and the local timer value for the controller associated with the current Cycle Start Time value.

Access the Message object

Access the Message object through the GSV/SSV instructions. Specify the message tag name to determine which Message object you want. The Message object provides an interface to setup and trigger peer-to-peer communications. This object replaces the MG data type of the PLC-5 processor.

Attribute	Data Type	Instruction	Description
ConnectionPath	SINT[82]	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 TIME32	GSV SSV	Requested packet rate of the connection.

Attribute	Data Type	Instruction	Description
MessageType	SINT	GSV SSV	Specifies the type of message. The value has a specific meaning: • 0. Not initialized
Port	SINT	GSV SSV	Indicates which port the message should be sent on. Each value has a specific meaning: • 1. Backplane. • 2. Serial port.
Timeout Multiplier	SINT	GSV SSV	Determines when a connection should be considered timed out and closed. Each value has a specific meaning: • 0. Connection times out in four times the update rate. This is the default. • 1. Connection times out in eight times the update rate. • 2. Connection times out in 16 times the update rate.
Unconnected Timeout	DINT TIME32	GSV SSV	Timeout in microseconds for all unconnected messages. The default is 30,000,000 microseconds (30 s).

Access the Safety object

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Safety Controller object provides safety status and safety signature information. The SafetyTask and SafetyFaultRecord attributes can capture information about non-recoverable faults.

See the GuardLogix Controllers User Manual, publication 1756-UM020.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
SafetyLockedState	SINT	GSV	None	Indicates whether the controller is safety locked or unlocked.

SafetySILConfiguration	SINT	GSV	None	Specifies the safety SIL configuration. • 2 — SIL2/PLd • 3 — SIL3/PLe
SafetyStatus	INT	GSV	None	Specifies the safety status. Each value has a specific meaning: : • 1000000000000 000 — Safety task OK. • 1000000000000 001 — Safety task inoperable. • 00000000000000 000 — Partner missing. • 00000000000000 001 — Partner unavailable. • 00000000000000 010 — Hardware incompatible. • 00000000000000 011 — Firmware incompatible.
SafetySignature Exists	SINT	GSV	GSV	Indicates whether the safety task signature is present.
SafetySignature ID (Applicable to Compact GuardLogix 5370, and GuardLogix 5570 controllers only)	SINT	GSV	None	32-bit identification number.
SafetySignature (Applicable to Compact GuardLogix 5370, and GuardLogix 5570 controllers only)	String	GSV	None	32-bit identification number includes ID number plus date and time stamp.
SafetyTaskFault Record	DINT[11]	GSV	None	Records safety task faults.
SafetySignatureIDLong (Applicable to Compact GuardLogix 5380 and	SINT[33]	GSV	None	32 byte Safety signature ID in byte array. The 1st byte is the size of of the

GuardLogix 5580 controllers only)				safety signature ID in bytes and remaining 31 bytes is the signature ID.
SafetySignatureIDHex (Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only)	String	GSV	None	64 character Hexadecimal string representation of signature ID
SafetySignatureDateTime (Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only).	String	GSV	None	27 character date time of a safety signature in the format of mm/dd/yyyy, hh:mm:ss.iii<AM or PM>

Access the Task object

The TASK object provides status information about a task. Specify the task name to determine which TASK object you want.

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
DisableUpdateOutputs	DINT	GSV SSV	None	Enables or disables the processing of outputs at the end of a task. • Set the attribute to 0 to enable the processing of outputs at the end of the task. • Set the attribute to 1 (or any non-zero value) to disable the processing of outputs at the end of the task.
EnableTimeOut	DINT	GSV SSV	None	Enables or disables the timeout function of an event task. • Set the attribute to 0 to disable the timeout function. • Set the attribute to 1 (or any non-zero value) to enable the timeout function.
InhibitTask	DINT	GSV SSV	None	Prevents the task from executing. If a task is inhibited, the controller still prescans the task when the controller

Attribute	Data Type	Instruction within Standard Task	Instruction within Safety Task	Description
				transitions from Program mode to Run or Test mode. • Set the attribute to 0 to enable the task • Set the attribute to 1 (or any non-zero value) to inhibit (disable) the task
Instance	DINT	GSV	GSV	Provides the instance number of this TASK object. Valid values are 0...31.
LastScanTime	DINT TIME32	GSV SSV	None	Time it took to execute this program the last time it was executed. Time is in microseconds.
MaxInterval	DINT[2] TIME32[2] TIME	GSV SSV	None	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.
MaxScanTime	DINT TIME32	GSV SSV	None	Maximum recorded execution time for this program. Time is in microseconds.
MinInterval	DINT[2] TIME32[2] TIME	GSV SSV	None	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.
Name	String	GSV	GSV	Name of the task.
OverlapCount	DINT	GSV SSV	GSV SSV	The number of times that the task was triggered while it was still executing. Valid for an event or periodic task. To clear the count, set the attribute to 0.
Priority	INT	GSV SSV	GSV	Relative priority of this task as compared to the other tasks. Valid values 0...15.
Rate	DINT	GSV SSV	GSV	The time interval between executions of the task. Time is in microseconds.

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Instruction within Standard Task</th><th>Instruction within Safety Task</th><th>Description</th></tr>
<tr><td>StartTime</td><td>DINT[2]
DT
LINT</td><td>GSV
SSV</td><td>None</td><td>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.</td></tr>
<tr><td>Status</td><td>DINT</td><td>GSV
SSV</td><td>None</td><td>Provides status information about the task. Once the controller sets one of these bits, you must manually clear it.
To determine if:
an EVENT instruction triggered the task (event task only), examine bit 0
a timeout triggered the task (event task only), examine bit 1
an overlap occurred for this task, examine bit 2</td></tr>
<tr><td>SynchronizeRedundancyDataDisabled</td><td>DINT</td><td>GSV</td><td>None</td><td>Indicates if runtime tag crossloading for standard tasks is enabled in a redundancy application.
0 indicates that tag crossloading for the standard tasks is enabled.
1 indicates that tag crossloading for the standard tasks is disabled.</td></tr>
<tr><td>Watchdog</td><td>DINT</td><td>GSV
SSV</td><td>GSV</td><td>Time limit for execution of all programs associated with this task. Time is in microseconds.
If you enter 0, these values are assigned:<table><tr><th>Time:</th><th>Task Type:</th></tr><tr><td>0.5 sec</td><td>periodic</td></tr><tr><td>5.0 sec</td><td>continuous</td></tr></table></td></tr>
</table>

Access the TimeSynchronize object

The TIMESYNCHRONIZE object provides a Common Industrial Protocol (CIP) interface to the IEEE 1588 (IEC 61588) Standard for a precision clock synchronization protocol for networked measurement and control systems. Access the TIMESYNCHRONIZE object through the GSV/SSV instructions.

For more information about this object, refer to the Deploying Scalable Time Distribution within a Converged Plantwide Ethernet Architecture Design Guide.

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Instruction</th><th>Description</th></tr>
<tr><td>ClockType</td><td>INT</td><td>GSV</td><td>The type of clock.<table><tr><td>Bit</td><td>Type of Clock</td></tr></table></td></tr>
</table>

Attribute	Data Type	Instruction	Description	
			0	Ordinary Clock
			1	Boundary Clock
			2	Peer-to-peer transparent clock
			3	End-to-end transparent clock
			4	Management Node
			All other bits are reserved.	
CurrentTimeMicroseconds	DINT[2] DT LINT	GSV	Current value of System Time in microseconds.	
CurrentTimeNanoseconds	LINT LDT	GSV	Current value of System Time in nanoseconds.	
DomainNumber	SINT	GSV	The PTP clock domain. The value is between 0…255. The default is 0.	
CurrentTimeMicroseconds	LINT DINT	GSV	Current value of System Time in microseconds.	
CurrentTimeNanoseconds	LINT LDT	GSV	Current value of System Time in nanoseconds.	
DomainNumber	SINT	GSV	The PTP clock domain. The value is between 0…255. The default is 0.	
GrandMasterClockInfo	Structure	GSV	Property information about the grandmaster clock. Requires 24 bytes of storage.	
Grandmaster Clock Information structure:				
ClockIdentity	SINT[8]			
ClockClass	INT			
TimeAccuracy	INT			
OffsetScaledLogVariance	INT			
CurrentUtcOffset	INT			
TimePropertyFlags	INT			
TimeSource	INT			
Priority1	INT			
Priority2	INT			
IsSynchronized	DINT	GSV	Local clock is synchronized with a master.	
			Value	Meaning
			0	Not synchronized
			1	Synchronized
LocalClockInfo	Structure	GSV	Property information about the local clock.	

Attribute	Data Type	Instruction	Description
			Requires 20 bytes of storage.
Local Clock Information structure:			
ClockIdentity	SINT[8]		
ClockClass	INT		
TimeAccuracy	INT		
OffsetScaledLogVariance	INT		
CurrentUtcOffset	INT		
TimePropertyFlags	INT		
TimeSource	INT		
ManufactureIdentity	DINT	GSV	The IEEE OUI (Organization Unique Identity) for the manufacturer.
MaxOffsetFromMaster	LINT DINT	GSV / SSV	Maximum offset from master in nanoseconds.
MeanPathDelayToMaster	LINT DINT	GSV	Average path delay from master to local clock in nanoseconds.
NumberOfPorts	INT	GSV	The number of ports of this clock.
OffsetFromMaster	LINT DINT	GSV	The calculated difference between the local clock and the master clock, based on the most recent Sync message, in nanoseconds.
PTPEnable	DINT	GSV / SSV	The enable status for CIP Sync/PTP/Time Synchronization on the device. Value: Meaning 0: Disable 1: Enable
ParentClockInfo	Structure	GSV	Property information about the parent clock. Requires 16 bytes of storage.
Parent Clock Information structure:			
ClockIdentity	SINT[8]		
PortNumber	INT		
ObservedOffsetScaledLogVariance	INT		
ObservedPhaseChangeRate	DINT		
PortEnableInfo	Structure	GSV	The port enable configuration of each port on the device. Size = 2 + (No. of Enabled Ports x 4) Maxsize = 42 bytes

Attribute	Data Type	Instruction	Description
Port Enable status structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
PortEnable	INT		
PortLogAnnounceIntervalInfo	Structure	GSV	The interval between successive "Announce" messages issued by a master clock on each PTP port of the device. Size = 2 + (No. of Enabled Ports x 4) Maxsize = 42 bytes
Port Log Announce Interval structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
PortLogAnnounceInterval	INT		
PortLogSyncIntervalInfo	Structure	GSV	The interval between successive Sync messages issued by a master on each PTP port of the device. Size = 2 + (No. of Enabled Ports x 4) Maxsize = 42 bytes
Port Log Sync Interval structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
PortLogAnnounceInterval	INT		
PortPhysicalAddressInfo	Structure	GSV	The physical and protocol address of each port of the device. Size = 2 + (No. of Enabled Ports x 36) Maxsize = 362 byts
Port Physical Address structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
Protocol	SINT[16]		
SizeOfAddress	INT		
Port Address	SINT[16]		
PortProfileIdentityInfo	Structure	GSV	Profile of each port of the device. Size = 2 + (No. of Enabled Ports x 10)

Attribute	Data Type	Instruction	Description
			Maxsize = 102
Port Profile Identity structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
ClockIdentity	SINT[8]		
PortProtocolAddressInfo	Structure	GSV	The network and protocol address of each port of the device. Size = 2 + (No. of Enabled Ports x 22) Maxsize = 222
Port Protocol Address structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
NetworkProtocol	INT		
SizeOfAddress	INT		
PortAddress	SINT[16]		
PortStateInfo	Structure	GSV	The current state of each PTP port on the device. Size = 2 + (No. of Enabled Ports x 4) Maxsize = 42 bytes
Port State structure:			
NumberOfPorts	INT		Maximum number of ports is 10.
Structure repeated for the number of ports:			
PortNumber	INT		
PortState	INT		
Priority1	SINT	GSV / SSV	Priority1 (Master Override) value for the local clock. **Note:** Value is Unsigned.
Priority2	SINT	GSV / SSV	Priority2 (Tie Breaker) value for the local clock. **Note:** Value is Unsigned.
ProductDescription	Structure	GSV	Product description of the device that contains the clock. Requires 68 bytes of storage.
Product Description structure:			
Size	DINT		
Description	SINT[64]		
RevisionData	Structure	GSV	Revision data of the device that contains the clock. Requires 36 bytes of storage.

Attribute	Data Type	Instruction	Description
Revision Data structure:			
Size	DINT		
Revision	SINT[32]		
StepsRemoved	INT	GSV	The number of CIP Sync Regions between the local clock and the grandmaster (that is, the number of boundary clocks +1)
SystemTimeAndOffset	Structure	GSV	System time in microseconds and the offset to the local clock value.
System Time and Offset structure:			
SystemTime	LINT DINT		
SystemOffset	LINT DINT		
UserDescription	Structure	GSV	User description of the device that contains the clock. Requires 132 bytes of storage.
User Description structure:			
Size	DINT		
Description	SINT[128]		

Access the WallClockTime object

The WallClockTime object provides a timestamp that the controller can use for scheduling.

Tip: Setting the WALLCLOCKTIME object is limited to no more than one update every 15 seconds.

IMPORTANT: To ensure proper time is read using the GSV instruction, include the WALLCLOCKTIME GSV in only one user task.

IMPORTANT: To ensure proper time is read using the GSV instruction, place the UID/UIE instruction pair around the WALLCLOCKTIME GSV instances in user tasks that can be interrupted by WALLCLOCKTIME GSV instances in other tasks. No UID/UIE pair is required when the WALLCLOCKTIME GSV exists in only one user task.

IMPORTANT:

When disabling PTP on a controller, to give the controller time to process the disable, use a two-second delay before setting the WallClockTime (WCT) in the controller. Otherwise, there is a risk of the grandmaster clock overwriting the WCT.

Attribute	Data Type	Instruction	Description
ApplyDST	SINT	GSV SSV	Identifies whether to enable daylight savings time. Each value has a specific meaning: • 0. Do not adjust for daylight savings time. • Non zero. Adjust for daylight savings time.
CSTOffset	DINT[2] TIME32[2] TIME	GSV SSV	Positive offset from the CurrentValue of the CST object (coordinated system time). Value in microseconds. The default is 0.
CurrentValue	DINT[2] DT LINT	GSV SSV	Current value of the wall clock time. The number of microseconds elapsed since 0000 hours 1 January 1970. **Note:** You can set this value to no later than 12/29/2068. The CST and WALLCLOCKTIME objects are mathematically related in the controller. For example, if adding the CST CurrentValue and the WALLCLOCKTIME CSTOffset, the result is the WALLCLOCKTIME CurrentValue.
DateTime	DINT[7] DATETIMESTRUCT	GSV SSV	The date and time. Each value has a specific meaning: • DINT[0]. Year • DINT[1]. Month (1...12) • DINT[2]. Day (1...31) • DINT[3]. Hour (0...23) • DINT[4]. Minute (0...59) • DINT[5]. Seconds (0...59) • DINT[6]. Microseconds (0...999,999)

Attribute	Data Type	Instruction	Description
DSTAdjustment	INT	GSV SSV	The number of minutes to adjust for daylight saving time.
LocalDateTime	DINT[7] DATETIMESTRUCT	GSV SSV	Current adjusted local time. Each value has a specific meaning: • DINT[0]. Year • DINT[1]. Month (1...12) • DINT[2]. Day (1...31) • DINT[3]. Hour (0...23) • DINT[4]. Minute (0...59) • DINT[5]. Seconds (0...59) • DINT[6]. Microseconds (0...999,999)
TimeZoneString	INT	GSV SSV	Time zone for the time value.

Determine Controller Memory Information

This is information is not applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers. In these controllers, the memory used attributes are not supported or accesible.

The memory of the controller is divided into I/O memory and expansion memory. This table shows how the controller uses each type of memory:

This	Uses memory from
I/O tags	I/O memory
produced tags	
consumed tags	
communication via MSG instructions	
communication with workstations	
tags other than I/O, produced, or consumed tags	expansion memory
logic routines	
communication with polled (OPC/DDE) tags that use RSLinx Classic.	I/O memory and expansion memory

Note that the controller returns values in the number of 32-bit words. To see a value in bytes, simply multiply by 4.Use this procedure to get the following information about the controller's memory:

- available (free) I/O and expansion memory
- total I/O and expansion memory
- largest contiguous block of I/O and expansion memory

Get Memory Information From the ControllerTo get memory information from the controller, execute a Message (MSG) instruction that is configured as follows:From the Message Properties dialog - 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
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 expansion 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 expansion memory
	source_array[11]	0	Null value
Source Length	12		Write 12 bytes (12 SINTs).
Destination	*INT_array* of type INT[29]		

From the Message Properties dialog - Communication 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	
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 expansion 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 expansion memory
	source_array[11]	0	Null value
Source Length	12	Write 12 bytes (12 SINTs).	
Destination	*INT_array* of type INT[29]		

Choose the Memory Information You WantThe MSG instruction returns the following information to INT_array (the destination tag of the MSG instruction).

IMPORTANT: For a 1756-L55M16 controller, the MSG instruction returns two values for each expansion memory category. To determine the free or total expansion memory of a 1756-L55M16 controller, add both values for the category.

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 expansion 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 expansion 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 expansion 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 expansion 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
1756-L55M16 controllers only—largest contiguous block of additional free expansion 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 expansion 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

Convert INTs to a DINTThe 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.

DeviceNet Status Codes

The following are the DeviceNet Status Codes.

Status Code	Description of Status	Recommended Action
0-63	DeviceNet node address of scanner or slave device.	None.
65	The AutoScan option is on and the scanner is in idle mode.	None.
67	Scanner is Secondary scanner.	None.
68	Primary scanner has detected no Secondary scanner.	Configure another scanner to be the Secondary scanner.
69	Primary and Secondary configurations are mismatched.	Check configuration of the Secondary scanner.

Status Code	Description of Status	Recommended Action
70	The address of the scanner is already in use by another device on the network.	Change the address of the scanner to an unused address.
71	Invalid data in scan list.	Use RSNetWorx software to reconfigure the scan list.
72	Slave device stopped communicating. If communication is not reestablished with the slave device during the next attempt, status code will change to 78.	• Verify slave device's power and network connections. • If slave device is polled, verify that interscan delay time is adequate for the device to return data. • Verify that slave device is functioning properly.
73	Slave device's identity information does not match electronic key in scanner.	• Make sure that the correct slave device is connected at this address. • Make sure that the slave device matches the specified electronic key (vendor, product code, product type). • Verify that slave device is functioning properly.
74	Scanner detected data overrun on DeviceNet communication port.	• Check network communication traffic. • Verify that slave device is functioning properly.
75	Either or both of the following are present. • The scanner does not have a scan list. • The scanner has not received communication from any other device.	Verify that the scanner has the following. • A configured scan list. • A properly-wired connection to the network.
76	No direct network traffic for scanner. The scanner hears other network communication but does not hear any directed to it.	None.
77	During initialization, the data size expected by the slave device does not match the size in the corresponding scan list entry.	• Use RSNetWorx software to check the slave device and the scan list for the correct input and output sizes for the slave device. • Verify that slave device is functioning properly.
78	Slave device is configured in scan list, but is not communicating.	• Verify slave device's power and network connections. • If the slave device is polled, make sure the interscan delay is long

Status Code	Description of Status	Recommended Action
		enough for the slave device to return its data. • If needed, use RSNetWorx software to perform the following. ◦ Add the slave device to the DeviceNet network. ◦ Delete the slave device from scanner's scan list. ◦ Inhibit the slave device in the scanner's scan list. • Verify that slave device is functioning properly.
79	Scanner has failed to transmit a message.	• Make sure that the scanner is connected to a valid network. • Check for disconnected cables. • Verify network baud rate.
80	Scanner is in idle mode.	If desired, put scanner in run mode by doing the following. • Putting the controller in run/remote run mode using the keyswitch on the controller or through the Logix Designer application AND • Turning on bit O.CommandRegister.Run for the scanner.
81	Controller has set the scanner to faulted mode.	Bit O.CommandRegister.Fault for the scanner is on. Correct condition that

Status Code	Description of Status	Recommended Action
		caused controller to set this bit and then turn this bit off.
82	Error detected in sequence of fragmented I/O messages from slave device.	• Use RSNetWorx software to perform the following. ◦ Check scan list entry for the slave device to make sure that its input and output data sizes are correct. ◦ Check the configuration of the slave device. • Verify that slave device is functioning properly.
83	Slave device returns error responses when the scanner attempts to communicate with it.	• Use RSNetWorx software to perform the following. ◦ Check the accuracy of the scan list. ◦ Check the configuration of the slave device. The slave device may be in another scanner's scan list. • Cycle power to the slave device. • Verify that slave device is functioning properly.
84	Scanner is initializing the DeviceNet network.	None. This code clears itself once the scanner attempts to initialize all the slave devices on the network.
85	During runtime, the data size sent by the slave device does not match the size in the corresponding scan list entry.	Since variable length poll data is not supported, verify that the slave device is functioning properly.
86	The slave device is in idle mode or not producing data while the scanner is in run mode.	• Check the configuration and status of the slave device. • If you set up a master/slave relationship between 2 scanners, make sure both scanners are in run mode.
87	Scanner cannot listen to shared inputs from slave device because the owning scanner has not established communication with that slave device.	• Verify the owning scanner connection and configuration. • Slave device may not be producing data.
88	Scanner cannot listen to shared inputs from slave device because I/O parameters (for example, polled or strobed, electronic key, data size) for that	In this scanner, reconfigure the I/O parameters for the shared inputs scan list entry so that they match those same parameters in the owning scanner.

Status Code	Description of Status	Recommended Action
	slave device are configured differently between this scanner and the owning scanner.	
89	Scanner failed to configure a slave device using the Automatic Device Recovery (ADR) parameters.	Make sure that you installed a compatible slave device.
90	Controller has set the scanner to disabled mode.	If desired, enable the scanner by turning off bit O.CommandRegister.DisableNetwork for the scanner.

91	Bus-off condition likely due to cable or signal errors.	• Cycle power to the scanner, slave device(s), and/or network.
		• Verify that all devices are set to the same baud rate.
		• Check DeviceNet cabling to make sure no short circuits exist between CAN (blue and white) wires and power or shield (black, red, and shield) wires.
		• Check the media system for the following noise sources.
		  ◦ Device located near high-voltage power cable.
		  ◦ Incorrect or no termination resistor used.
		  ◦ Improper grounding.
		• Device on network producing noise or incorrect data on the network.

92	DeviceNet cable not supplying power to the scanner's communication port.	• Verify the network's 24V dc power supply is operating properly. • Verify good cable condition. • Check cable connections to the scanner.
95	The scanner's firmware is being updated or a configuration is being downloaded.	None. Do not disconnect the scanner while the update is in process, otherwise, existing data in scanner memory will be lost.
97	The controller has placed the scanner in halt mode.	Bit O.CommandRegister.HaltScanner for the scanner is on. Turn this bit off and then cycle scanner power.
98	General firmware error.	Replace device.
99	System failure.	Replace device.

Get and Set System Data

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:

- The GSV instruction retrieves the specified information and places it in the destination.
- The SSV instruction sets the specified attribute with data from the source.

Attention: Use the SSV instruction carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

To get or set a system value:

1. Open the Logix Designer application project.
2. From the **Help** menu, click **Contents**.
3. Click **Index**.
4. Type **gsv/ssv objects** and click **Display**.
5. Click the required object.

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

6. In the list of attributes for the object, identify the attribute that you want to access.
7. 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])	Create a user-defined data type that matches the organization of data used by the attribute. Then create a tag for the attribute and use the data type you created.

8. In your ladder logic routine, enter the appropriate instruction.

To	Enter this instruction
get the value of an attribute	GSV
set the value of an attribute	SSV

9. Assign the required operands to the instruction.

   Refer to the GSV/SSV instruction for information on these operands.

GSV/SSV Programming Example

The following examples use GSV instruction to get fault information.

Example 1: Getting I/O Fault Information

This example gets fault information from the I/O module disc_in_2 and places the data in a user-defined structure disc_in_2_info.

Ladder Diagram

Structured Text

```
GSV(MODULE,disc_in_2,FaultCode,disc_in_2_info.FaultCode);
```

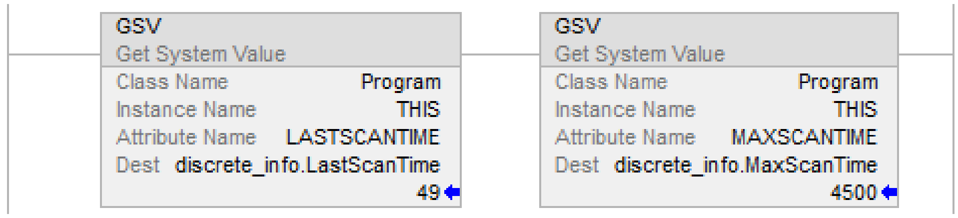
```
GSV(MODULE,disc_in_2,FaultInfo,disc_in_2_info.FaultInfo);
```

```
GSV(MODULE,disc_in_2,Mode,disc_in_2_info.Mode);
```

Example 2: Getting Program Status Information

This example gets status information about program discrete and places the data in a user-defined structure discrete_info.

Ladder Diagram



Structured Text

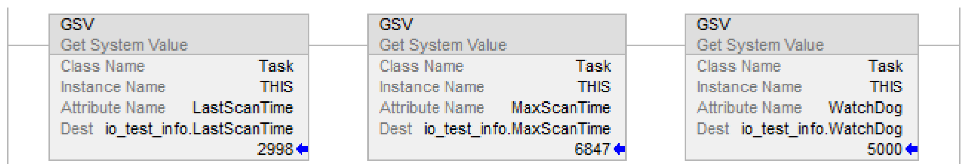
```
GSV(PROGRAM,DISCRETE,LASTSCANTIME,discrete_info.LastScanTime);
```

```
GSV(PROGRAM,DISCRETE,MAXSCANTIME,discrete_info.MaxScanTime);
```

Example 3: Getting Task Status Information

This example gets status information about task IO_test and places the data in a user-defined structure io_test_info.

Ladder Diagram



Structured Text

```
GSV(TASK,IO_TEST,LASTSCANTIME,io_test_info.LastScanTime);
```

```
GSV(TASK,IO_TEST,MAXSCANTIME,io_test_info.MaxScanTime);
```

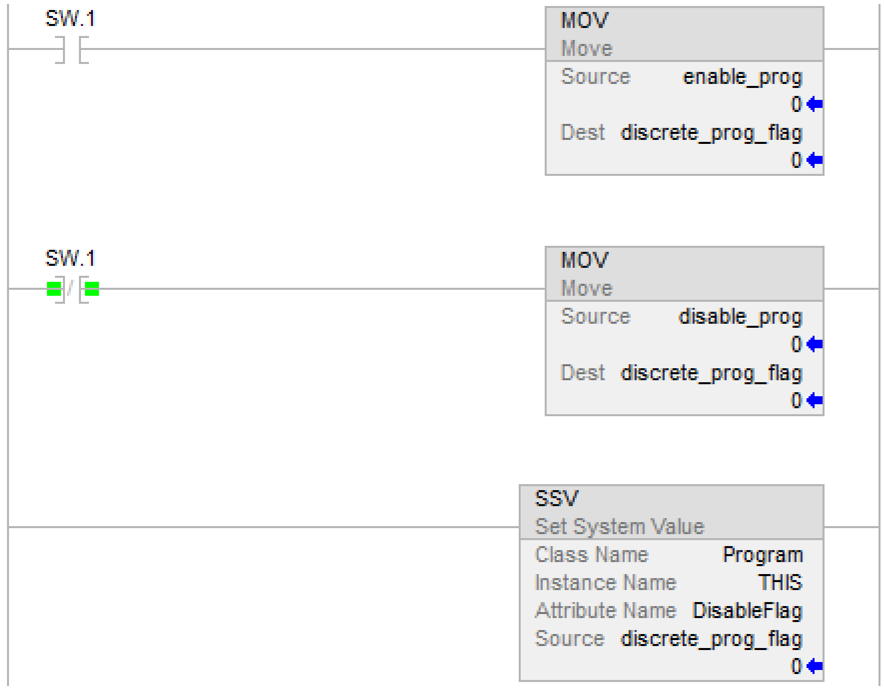
```
GSV(TASK,IO_TEST,WATCHDOG,io_test_info.Watchdog);
```

Setting Enable and Disable Flags

The following example uses the SSV instruction to enable or disable a program. You could also use this method to enable or disable an I/O module, which is a program solution similar to using inhibit bits with a PLC-5 processor.

Based on the status of SW.1, place the appropriate value in the disable flag attribute of program discrete.

Ladder Diagram



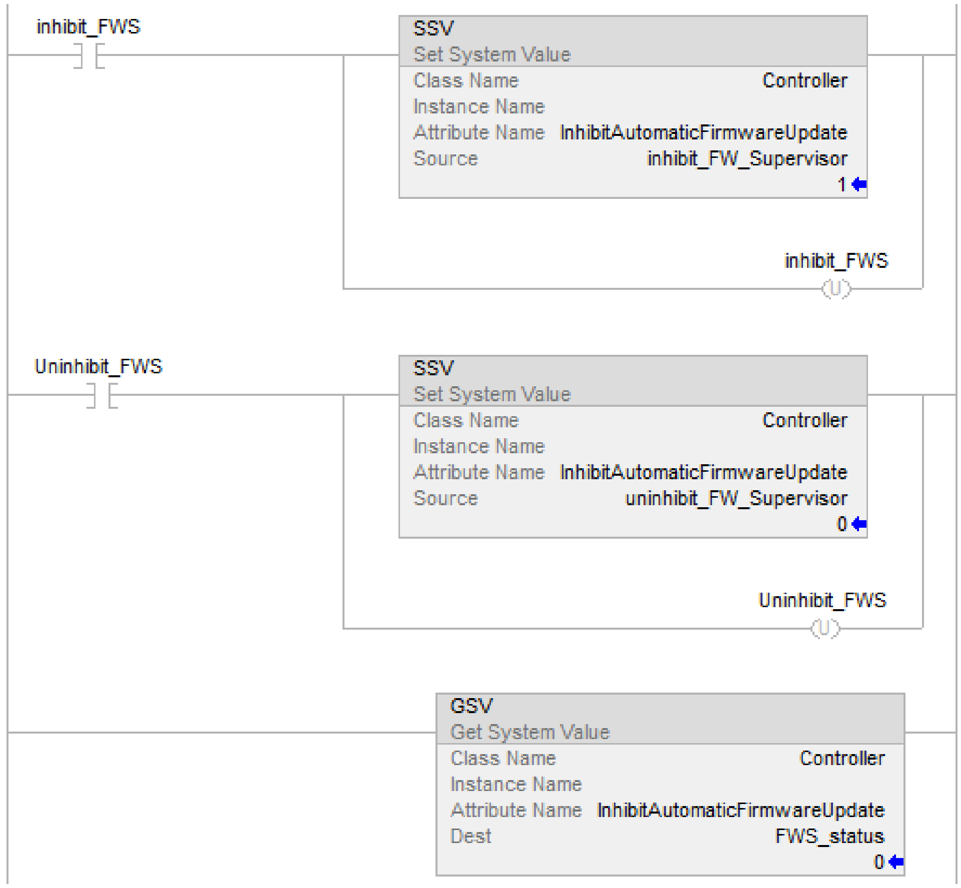
Structured Text

```
IF SW.1 THEN

discrete_prog_flag := enable_prog;  ELSE

  discrete_prog_flag := disable_prog;

END_IF;

SSV(PROGRAM,DISCRETE,DISABLEFLAG,discrete_prog_flag);
```

Inhibiting and Uninhibiting FirmwareSupervisor Automatic Firmware Update

The following example uses the GSV/SSV instruction to inhibit or uninhibit the Automatic Firmware Update attribute of the controller. If you write a value of 1, it inhibits the feature. If you write a value of 0, the feature is uninhibited. The status of the attribute can also be read with a GSV.

Ladder Diagram



GSV/SSV Objects

When entering a GSV/SSV instruction, specify the object and its attribute to access. In some cases, there will be more than one instance of the same type of object. Be sure to specify the object name. For example, each task has its own TASK object that requires specifying the task name to gain access.

IMPORTANT: The SSV attributes must be uploaded to be saved to the project.

IMPORTANT: For the GSV instruction, only the specified size of data is copied to the destination. For example, if the attribute is specified as a SINT and the destination is a DINT, only the lower 8 bits of the DINT destination are updated, leaving the remaining 24 bits unchanged.

IMPORTANT: The alarm buffer was removed from the subscription functions for alarming in the v21 firmware, and is no longer available. GSV instructions that previously referenced the alarm buffer attribute are invalidated when verifying the project. It is the responsibility of the programmer to correctly change any application code that relied on this attribute.

These are the GSV/SSV objects. The objects available for access are dependent on the controller.

- AddOnInstructionDefinition on page 196
- Axis on page 200

GSV/SSV Safety Objects

For safety tasks, the GSV and SSV instructions are more restricted.

Tip: SSV instructions in safety and standard tasks cannot set bit 0 (major fault on error) in the mode attribute of a safety I/O module.

For safety objects, the following table shows which attributes you can get values for using the GSV instruction and which attributes you can set using the SSV instruction in safety and standard tasks.

WARNING: CAUTION: Use the GSV/SSV instructions carefully. Making changes to objects can cause unexpected controller operation or injury to personnel.

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
			GSV	SSV	GSV	SSV
Safety Task	Instance	Provides instance number of this task object. Valid values are 0...31.	✓		✓	
	MaximumInterval	The max time interval between successive			✓	✓

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		executions of this task.				
	MaximumScanTime	Max recorded execution time (ms) for this task.			✓	✓
	MinimumInterval	The min time interval between successive executions of this task.			✓	✓
	Priority	Relative priority of this task as compared to other tasks. Valid values are 0...15.	✓		✓	
	Rate	Period for the task (in ms), or timeout value for the task (in ms).	✓		✓	
	Watchdog	Time limit (in ms) for execution of all programs associated with this task.	✓		✓	
	DisableUpdateOutputs	Enables or disables the processing of outputs at the end of a task. • Set the attribute to 0 to enable the processing of outputs at the end of the task.			✓	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		• Set the attribute to 1 (or any non-zero value) to disable the processing of outputs at the end of the task.				
	EnableTimeOut	Enables or disables the timeout function of a task. • Set the attribute to 0 to disable the timeout function. • Set the attribute to 1 (or any non-zero value) to enable the timeout function.			✓	
	InhibitTask	Prevents the task from executing. If a task is inhibited, the controller still prescans the task when the controller transitions from Program mode			✓	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		to Run or Test mode. • Set the attribute to 0 to enable the task • Set the attribute to 1 (or any non-zero value) to inhibit (disable) the task				
	LastScanTime	Time it took to execute this program the last time it was executed. Time is in microseconds.			✔	
	Name	The name of the task				
	OverlapCount	The number of times that the task was triggered while it was still executing. Valid for an event or periodic task. To clear the count, set the attribute to 0.			✔	
	StartTime	Value of WALLCLOCKTIME when the last execution of the task was started. DINT[0] contains the lower 32 bits			✔	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		of the value; DINT[1] contains the upper 32 bits of the value.				
	Status	Provides status information about the task. Once the controller sets one of these bits, you must manually clear it. To determine if: • an EVENT instruction triggered the task (event task only), examine bit 0 • a timeout triggered the task (event task only), examine bit 1 • an overlap occurred for this task, examine bit 2			✔	
Safety Program	Instance	Provides the instance number of the program object.	✔		✔	
	MajorFaultRecord	Records major faults for this program.	✔	✔	✔	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
	MaximumScanTime	Max recorded execution time (ms) for this program.			✔	✔
	Disable Flag	Controls this program's execution. Each value has a specific meaning: • 0. Execution enabled. • Non zero. Execution disabled.			✔	
	MaximumScanTime	Maximum recorded execution time (ms) for this program.			✔	
	Minor Fault Record	Records minor faults for this program.			✔	
	LastScanTime	Time it took to execute this program the last time it was executed. Time is in microseconds.			✔	
	Name	The name of the task.				
Safety Routine	Instance	Provides the instance number for this routine object. Valid values are 0...65,535.	✔			
Safety Controllers	SafetyLockedState (SINT)	Indicates whether the controller is			✔	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		safety-locked or -unlocked.				
	SafetySILConfiguration (SINT)	Specifies the safety SIL configuration as: • 2 = SIL2/PLd • 3 = SIL3/PLe	✔		✔	
	SafetyStatus (INT) (Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only).	Applications configured for SIL3/PLe, specify the safety status as: • Safety task OK. (1100000000000000) • Safety task inoperable. (1100000000000011) • Partner missing. (0100000000000000) • Partner unavailable. (0100000000000001) • Hardware incompatible (0100000000000010) • Firmware incompatib			✔	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		le. (01000000 000000011) **Tip:** For applications configured for SIL2/PLd, bits 15, 0, and 1 should be ignored if they can be different values based on the slot +1 of the Primary Controller. See the above status for meaning. Applications configured for SIL2/PLd, specify the safety task as: • Safety task OK (x1000000 000000xx) • Safety task inoperable (x1000000 0000001xx)				
	SafetyStatus (INT) (Applicable to Compact GuardLogix 5370 and GuardLogix 5570 controllers only).	Specifies the safety status as: • Safety task OK. (10000000 00000000) • Safety task inoperable. (10000000 00000001) • Partner missing.			✓	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
		(00000000 00000000) • Partner unavailable. (00000000 00000001) • Hardware incompatible (00000000 00000010) • Firmware incompatible. (00000000 00000011)				
	SafetySignature Exists (SINT)	Indicates whether the safety signature is present.	✔		✔	
	SafetySignatureID (DINT) (Applicable to Compact GuardLogix 5370, and GuardLogix 5570 controllers only)	32-bit identification number.			✔	
	SafetySignature (String) (Applicable to Compact GuardLogix 5370, and GuardLogix 5570 controllers only)	ID number plus date and time stamp.			✔	
	SafetyTaskFault Record (DINT)	Records safety task faults.			✔	
	SafetySignatureIDLong SINT [33]	The first byte is the size of the safety signature			✔	

Safety Object	Attribute Name	Attribute Description	Accessible from the Safety Task		Accessible from the Standard Task	
	(Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only)	ID in bytes and the remaining 32 bytes contain the content of the 32-byte Safety signature ID.				
	SafetySignatureIDHex(String) (Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only)	64 character Hexadecimal sting representation of signature ID			✓	
	SafetySignature DateTime(String) (Applicable to Compact GuardLogix 5380 and GuardLogix 5580 controllers only)	27 character date time of a safety signature in the format of mm/dd/yyyy, hh:mm:ss.iii<AM or PM>			✓	

Monitor Status Flags

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

- The status keywords are *not* case sensitive.
- Because the status flags can change so quickly, the Logix Designer application does *not* display the status of the flags (that is, even when a status flag is set, an instruction that references that flag is not highlighted).
- You *cannot* define a tag alias to a keyword.

You can use these keywords:

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, or • 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

To determine if:	Use:
an arithmetic operation causes a carry or borrow that tries to use bits that are outside of the data type For example: • adding 3 + 9 causes a carry of 1 • subtracting 25 - 18 causes a borrow of 10	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

Select the Message Type

After entering the MSG instruction and specifying the MESSAGE structure, select the **Configuration** tab of the Message Configuration dialog to specify the details of the message.

The **Configuration** tab also includes a check box for setting and clearing the .TO bit.

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

If the target device is a:	Select one of these message types:
Logix 5000 controller	CIP data table read CIP data table write
I/O module that you configure using the Logix Designer application	Module Reconfigure CIP Generic
PLC-5® controller*	PLC-5 typed read PLC-5 typed write PLC-5 word range read PLC-5 word range write
SLC™ controller* MicroLogix™ controller*	SLC typed read SLC typed write
Block transfer module*	block transfer read block transfer write
PLC-3® processor*	PLC-3 typed read PLC-3 typed write PLC-3 word range read

If the target device is a:	Select one of these message types:
	PLC-3 word range write
PLC-2® processor*	PLC-2 unprotected read PLC-2 unprotected write

* When redundancy is enabled for ControlLogix 5580 or GuardLogix 5580 controllers, these message types are not supported.

Specify this configuration information:

In this field:	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. I/O structure tags and Booleans are not supported. All other data types, for example INT, DINT, can be used.
Number of Elements	The number of elements you read/write depends on the message type and on the type of data you are using. For "word range" and "unprotected" messages, the size of an element is indicated in the dialog box. For CIP and "typed" messages, an element is a single element of the array that you specify as the source of a write or destination of a read
Destination Element	If you select a read message type, the Destination Tag is the first element of the tag in the Logix 5000 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.

Module Faults: 16#0000 - 16#00ff

These are the module faults: 16#0000 - 16#00ff

Code	String	Explanation and Possible Causes/Solutions
16#0001	Connection Error.	A connection to a module failed.
16#0002	Resource unavailable.	Either: • there are not enough connections available either for the controller or for the communication module being used to connect through. Check the connection use of the controller or communication

		module. If all of the connections are used, try to free some of the used connections or add another module to route the errant connection through. • the I/O memory limits of the controller are exceeded. Check the I/O memory available and make program or tag changes if needed. • the I/O module targeted does not have enough connections available. Check the number of controllers making a connection to this I/O module and verify that the number of connections is within the limits of the I/O module.
16#0005	Connection Request Error: Bad Class	The controller is attempting to make a connection to the module and has received an error. Either: • the configured address for the connection to the module is incorrect. • the module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.

		Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application. If you are using a 1756-DHRIO module, verify that the Channel type selected in the software (DH+ or remote I/O network) matches the module's rotary switch settings.
16#0006	Connection Request Error: Bad Class.	Either: • the response buffer is too small to handle the response data. • the module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0007	Connection Request Error: Bad Class.	A service request is unconnected, but should be connected.
16#0008	Service Request Error: Unsupported Service	The controller is attempting to request a service from the module that is not supported by the module.

16#0009	Module Configuration Invalid: parameter error. **Tip:** Additional Fault Information for this fault will be displayed as a hex code on the Connection Tab.	The configuration for the module is invalid. The module configuration may have been changed in the Data Monitor or programmatically. If available for the module, access the Connections tab of the Module Properties dialog box for the additional fault code. The additional fault code indicates the configuration parameter that is causing the fault. You may have to correct multiple parameters before this fault is cleared and connection is properly established.
16#000A	An attribute in the Get_Attributes_List or Set_Attributes_List has a non-zero status.	Either: • a connection is being created where the connection type is invalid. • an object attribute or tag value is invalid. If an object attribute or tag is invalid, export the Logix Designer file, then re-import it. Reschedule the ControlNet network after re-importing if applicable.
16#000C	Service Request Error: Invalid mode/state for service request.	The controller is attempting to request a service from the module and has received an error. First, verify that the module is not faulted. For an I/O module, this may indicate that the module has one of these conditions: • Limited communication, but has a Major Fault • A firmware update needs to be completed or is currently being completed. Refer to the Module Info tab to determine the exact cause.
16#000D	Object already exists.	An I/O map instance is created where the instance is already in use.
16#000E	Attribute value cannot be set.	A MSG instruction is configured to change an attribute value that cannot be changed.
16#000F	Access permission denied for requested service.	A MSG instruction has been configured to delete a map object that cannot be deleted.

16#0010	Mode or state of module does not allow object to perform requested service.	The state of the device prevents a service request from being handled.
16#0011	Reply data too large.	The reply to a message has a data size that is too large for the destination. Change the destination to a tag that can handle the data size and type being returned.
16#0013	Module Configuration Rejected: Data size too small.	The configuration for the module is invalid - not enough configuration data was sent. Verify that the correct module is being targeted.
16#0014	Undefined or unsupported attribute.	A MSG instruction is configured to change an attribute that does not exist.
16#0015	Module Configuration Rejected: Data size too large.	The configuration for the module is invalid - too much configuration data was sent. Verify that the correct module is being targeted.

Module Faults: 16#0100 - 16#01ff

These are the module faults: 16#0100 - 16#01ff

Code	String	Explanation and Possible Causes/Solutions
16#0100	Connection Request Error: Module in Use.	• The connection being accessed is already in use. Either: • The controller is attempting to make a specific connection to a module and the module cannot support more than one of these connections. • The target of a connection recognizes that the owner is attempting to remake a connection that is already running.
16#0103	Service Request Error: CIP transport class not supported.	Either: • The controller is requesting services not supported by the module. • The module in use (that is, the physical module) is different than the module specified in the I/O

Code	String	Explanation and Possible Causes/Solutions
		configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0106	Connection Request Error: Module owned and configured by another controller. Module may accept only one connection if Unicast is used.	An ownership conflict occurred for the connection. One of these conditions exists: • The Connection Request to this module has been rejected due to an Ownership conflict with another Owner (for example, another Controller). This may occur with modules such as output modules that only allow a single Owner to configure and control its outputs. This fault may also occur if the module is configured as Listen Only and supports only one connection. • If the Owner is connected to the module using a Unicast connection over EtherNet/IP, other connections to the module fail since the Owner controls the one connection.

Code	String	Explanation and Possible Causes/Solutions
		If the Owner is connected to the module using a Multicast connection over EtherNet/IP, Unicast connections to the module fail since the Owner controls the one connection. Configure both the Owner and the Listen-Only connection as Multicast.
16#0107	Connection Request Error: Unknown type.	A connection being accessed was not found.
16#0108	Connection Request Error: Connection type (Multicast/Unicast) not supported.	The controller is requesting a connection type not supported by the module. One of these conditions exists: • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application. • You may have configured a consumed tag to use a Unicast

Code	String	Explanation and Possible Causes/Solutions
		connection, but the producing controller does not support Unicast connections.
16#0109	Connection Request Error: Invalid connection size. **Tip:** Additional Error Information for this fault will be displayed as the tag name associated with the connection instance number that has the fault.	The connection size is inconsistent with that expected. Either: • the controller is attempting to set up a connection with the module and cannot - the size of the connection is invalid. • the controller may be attempting to connect to a tag in a producing controller whose size does not match the tag in this controller. • the module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • the fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application. If the module is a 1756 ControlNet module, verify that the chassis size is correct.

Code	String	Explanation and Possible Causes/Solutions
		For remote I/O adapters, verify that the rack size and/or rack density is correct.
16#0110	Connection Request Error: Module not configured.	The controller is attempting to set up a Listen Only connection with the module and cannot - the module has not been configured and connected to by an Owner (for example, another Controller). This controller is not an Owner of this module because it is attempting to establish a Listen Only connection, which requires no module configuration. It cannot connect until an Owner configures and connects to the module first.
16#0111	Requested Packet Interval (RPI) out of range.	Either: • the Requested Packet Interval (RPI) specified is invalid for this module or for a module in the path to this module. See the Advanced tab to enable the RPI from the producer. • the module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O

Code	String	Explanation and Possible Causes/Solutions
		configuration tree of the Logix Designer application. • for Listen Only connections: the RPI set by the owner of this module is slower than the one requested. Either increase the requested RPI or decrease the RPI the owner controller is using. See the Connection tab in the Module Properties dialog box for valid RPI values.
16#0113	Connection Request Error: Module connection limit exceeded.	The number of connections is greater than what is available on the module. The number of connections must be reduced or the hardware must be upgraded. To reduce the number of connections: • Change the Flex I/O communication adapter Comm Format from Input or Output configuration to **Rack Optimization**. When the Comm Format changes, the adapter must be removed and recreated in the I/O configuration tree. • If the configuration uses messaging over ControlNet, sequence the messages to reduce the number that are executing at the same time, or reduce the number of messages. Messages (MSG instructions) also use connections.
16#0114	Electronic Keying Mismatch: Electronic keying product code and/or vendor ID mismatch.	The Product Code of the actual module hardware does not match the Product Code of the module created in the software. Electronic Keying failed for this module. You may have a mismatch between the module created in the software and the actual module hardware.
16#0115	Electronic Keying Mismatch: Electronic Keying product type mismatch.	The Product Type of the actual module hardware does not match the Product Type of the module created in the software.

Code	String	Explanation and Possible Causes/Solutions
		Electronic Keying failed for this module. You may have a mismatch between the module created in the software and the actual module hardware.
16#0116	Electronic Keying Mismatch: Major and/or Minor revision invalid or incorrect.	The Major and/or Minor revisions of the module do not match the Major and/or Minor revisions of the module created in the software. Verify that you have specified the correct Major and Minor Revision if you have chosen Compatible Module or Exact Match keying. Electronic Keying failed for this module. You may have a mismatch between the module created in the software and the actual module hardware.
16#0117	Connection Request Error: Invalid Connection Point. **Tip:** Additional Error Information for this fault appears as the tag name associated with the controller to controller (C2C) that has the fault.	The connection is to an invalid port or port that is already in use. One of these conditions exists: • Another controller owns this module and has connected with a Communications Format different than the one chosen by this controller. Verify that the Communications Format chosen is identical to that chosen by the first owner controller of the module. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the

Code	String	Explanation and Possible Causes/Solutions
		same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application. • The controller may be attempting to connect to a nonexistent tag in a producing controller.
16#0118	Module Configuration Rejected: Format error.	An invalid configuration format is used. One of these conditions exists: • The configuration class specified does not match the class supported by the module. • The connection instance is not recognized by the module. • The path specified for the connection is inconsistent. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.

Code	String	Explanation and Possible Causes/Solutions
		Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0119	Connection Request Error: Module not owned.	The controlling connection is not open. Where a Listen Only connection is requested, the controlling connection is not open.
16#011A	Connection Request Error: Out of Connection Resources	The controller is attempting to set up a connection with the module and cannot - resources required are unavailable. If the module is a 1756 ControlNet module, up to five controllers can make Rack Optimization connections to the module. Verify that this number has not been exceeded. If the module is a 1794-ACN15, 1794-ACNR15, or 1797-ACNR15 adapter, only one controller can make a Rack Optimization connection to the module. Verify that this number has not been exceeded.

Module Faults: 16#0200 - 16#02ff

These are the module faults: 16#0200 - 16#02ff.

Code	String	Explanation and Possible Causes/Solutions
16#0203	Connection timed out.	The owner or originator recognizes that the target device is on the network or backplane, however, I/O data and messages are not being responded to. In other words, the target can be reached, but its response is not as expected. For example, this fault may be indicated where multicast Ethernet packets are not returned. When this fault occurs, the controller usually attempts to continuously remove and remake the connection.

Code	String	Explanation and Possible Causes/Solutions
		If you are using FLEX I/O modules, verify that you are using the correct terminal device.
16#0204	Connection Request Error: Connection request timed out.	The controller is attempting to make a connection, however, the target module is not responding. The device also appears to be missing from the backplane or network. To recover, take these actions: • Verify that the module has not been removed and is still functioning and receiving power. • Verify that the correct slot number has been specified. • Verify that the module is properly connected to the network. If you are using FLEX I/O modules, verify that the correct terminal block is in use.
16#0205	Connection Request Error: Invalid parameter.	Either: • The controller is attempting to set up a connection with the module and has received an error - a parameter is in error. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not

Code	String	Explanation and Possible Causes/Solutions
		support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0206	Connection Request Error: Requested size too large.	Either: • The controller is attempting to set up a connection with the module and has received an error - the request size is too large. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.

Module Faults: 16#0300 - 16#03ff

These are the module faults: 16#0300 - 16#03ff

Code	String	Explanation and Possible Causes/Solutions
16#0301	Connection Request Error: Out of buffer memory.	One of these conditions may exist: • The controller is attempting to set up a connection with the module and has received an error - a module in the path is out of memory. • The controller may be attempting to connect to a tag in a producing controller that is not marked as being produced. • The controller may be attempting to connect to a tag in a producing controller. That tag may not be configured to allow enough consumers. • Reduce the size or number of connections through this module. • One of the network modules between the module and the controller may be out of memory. Check network configuration of the system. • The module may be out of memory. Check system configuration and capabilities of module. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being

Code	String	Explanation and Possible Causes/Solutions
		connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0302	Connection Request Error: Out of communication bandwidth.	The controller is attempting to set up a connection with the module and has received an error - a module in the path has exceeded its communication bandwidth capacity. Increase the Requested Packet Interval (RPI) and reconfigure your network with RSNetWorx. Distribute the load on another bridge module.
16#0303	Connection Request Error: No bridge available.	The controller is attempting to set up a connection with the module and has received an error - a module in the path has exceeded its communication bandwidth capacity. Distribute the load on another bridge module.
16#0304	Not configured to send scheduled data.	The ControlNet module is not scheduled to send data. Use RSNetWorx software to schedule or reschedule the ControlNet network.
16#0305	Connection Request Error: ControlNet configuration in controller does not match configuration in bridge.	The ControlNet configuration in the controller does not match the configuration in the bridge module. This may occur because a ControlNet module was changed after the network was scheduled, or because a new control program has been loaded into the controller. Use RSNetWorx software to reschedule the connections.

Code	String	Explanation and Possible Causes/Solutions
16#0306	No ControlNet Configuration Master (CCM) available.	The ControlNet Configuration Master (CCM) cannot be found. The 1756-CNB and PLC-5C modules are the only modules capable of being a CCM and the CCM must be node number 1. Verify that a 1756-CNB or PLC-5C module is at node number 1 and is functioning properly. This fault may temporarily occur when the system is powered up and will be cleared when the CCM is located.
16#0311	Connection Request Error: Invalid port.	The controller is attempting to set up a connection with the module and has received an error. Verify that all modules in the I/O Configuration tree are the correct modules.
16#0312	Connection Request Error: Invalid link address.	The controller is attempting to set up a connection with the module and has received an error - an invalid link address has been specified. A link address can be a slot number, a network address, or the remote I/O chassis number and starting group. Verify that the chosen slot number for this module is not greater than the size of the rack. Verify that the ControlNet node number is not greater than the maximum node number configured for the network in RSNetWorx software.
16#0315	Connection Request Error: Invalid segment type.	The segment type or route is invalid. Either: • the controller is attempting to set up a connection with the module and has received an error - the connection request is invalid • the module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail.

Code	String	Explanation and Possible Causes/Solutions
		The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#0317	Connection Request Error: Connection not scheduled.	The controller is attempting to set up a ControlNet connection with the module and has received an error. Use RSNetWorx software to schedule or reschedule the connection to this module.
16#0318	Connection Request Error: Invalid link address - cannot route to self.	The controller is attempting to set up a connection with the module and has received an error - the link address is invalid. Verify that the associated ControlNet module has the correct slot and/or node number selected.
16#0319	Connection Request Error: No secondary resources available in redundant chassis.	The controller is attempting to set up a connection with the module and has received an error - the redundant module does not have the necessary resources to support the connection. Reduce the size or number of connections through this module or add another controller or ControlNet module to the system.

Code	String	Explanation and Possible Causes/Solutions
16#031a	Connection Request Error: Rack Connection Refused.	The controller is attempting to set up a Direct connection with the module and has received an error. A Rack Optimized connection has already been established to this module through the 1756-CNB/R in the same chassis. • Connect to this module via the 1756-CNB/R in the same chassis. • Connect to this module via a different 1756-CNB/R in order to use a Direct connection. • Change the first connection from Rack Optimized to Direct, and then reestablish the second direct connection. • Connect to this module from a controller in the same chassis as the module (do not connect via 1756-CNB/R.
16#031e	Connection Request Error: Cannot consume tag.	• The controller is attempting to connect to a tag in a producing controller and has received an error. • The controller is attempting to connect to a tag in a producing controller and that tag has already been used by too many consumers. Increase the maximum number of consumers on the tag.
16#031f	Connection Request Error: Cannot consume tag.	No SC (servicing controller) connection object was found that corresponds to a symbol instance.
16#0322	Connection Request Error: Connection point mismatch	A connection point mismatch has occurred. Either: • a new connection requested does not match the existing connection. Check the controllers that are using the connection and verify that all the configurations are identical. • the connection requested is not a listener or a controlling connection type.

Module Faults: 16#0800 - 16#08ff

These are the module faults: 16#0800 - 16#08ff

Code	String	Explanation and Possible Causes/Solutions
16#0800	Network link in path to module is offline.	No interpretation available.
16#0801	Incompatible multi-cast RPI.	No interpretation available.
16#0810	No target application data available.	The controlling application has not initialized the data to be produced by the target device. This may be caused when "Send Data" connections are configured in a target device and the controlling application for that target device has not initialized the data to be produced. For the target device associated with the "Send Data" connection reporting this connection error, start the controlling application and perform at least one write of data. Refer to the documentation for the target device and its controlling application for information on how to do this.
16#0814	Connection Request Error: Data Type Mismatch.	Invalid connection status information was found.

Module Faults: 16#fd00 - 16#fdff

The module faults: 16#fd00 - 16#fdff.

Code	String	Explanation and Possible Causes/Solutions
16#fd03	Connection Request Error: Required Connection missing	The controller is attempting to set up a connection with the module and has received an error - this module requires a particular set of connections and connection types, and one of those connection types is missing. • Contact Rockwell Automation technical support at Rockwellautomation.com.
16#fd04	Connection Request Error: No CST Master Detected	The controller is attempting to set up a connection with the module and has

Code	String	Explanation and Possible Causes/Solutions
		received an error - this module requires a CST master in the chassis. • Configure a module (typically a controller) in this chassis to be the CST master. • Contact Rockwell Automation technical support at Rockwellautomation.com.
16#fd05	Connection Request Error: No Axis or Group Assigned.	The controller is attempting to set up a connection with the module and has received an error - this module requires an axis or group table assigned. • Assign a Group or Axis. • Contact Rockwell Automation technical support at Rockwellautomation.com.
16#fd06	Transition Fault	The controller command to transition the SERCOS ring to a new phase returned an error from the module. Check for duplicate Drive Nodes.
16#fd07	Incorrect SERCOS Data Rate	An attempt to configure the SERCOS ring failed. The baud rate for all devices must be the same and supported by the drives and the SERCOS module.
16#fd08	SERCOS Comm Fault	Mainly two sets of faults may cause a Comm. Fault - Physical and interface faults. A possible source of physical faults is: • Broken ring • Loose connector • Fiber optics not clean • Electrical noise due to improper drive grounding • Too many nodes on the ring Interface errors are encountered when you are configuring third party drives. A possible source of interface errors is: • No SERCOS MST (Protocol Error) • Missed AT (drive did not send data when expected)

Code	String	Explanation and Possible Causes/Solutions
		• SERCOS timing error in phase 3 • Error in drive data returned to SERCOS module
16#fd09	Node Initialization Fault	An attempt by the controller to configure the node for cyclic operation returned an error.
16#fd0a	Axis Attribute Error	A bad response was received from a motion module.
16#fd0c	Error Different Grandmaster Fault	The end device has a different grandmaster than the controller.
16#fd1f	Bad Safety Protocol Format	An error occurred adding the safety network segment to a route.
16#fd20	No Safety Task	No safety task appears to be running.
16#fd22	Chassis Size Mismatch	Verify the number of physical expansion I/O modules configured for the controller and then update the number of modules selected from the Expansion I/O list on the General page in the Controller Properties dialog.
16#fd23	Chassis Size Exceeded	To verify the number of physical expansion I/O the controller supports, open the Controller Properties dialog and expand the Expansion I/O list on the General page. Configure the number of physical expansion I/O modules to match the selection in the Expansion I/O list.

Module Faults: 16#fe00 - 16#feff

The module faults: 16#fe00 - 16#feff.

Code	String	Explanation and Possible Causes/Solutions
16#fe01		An invalid configuration format was encountered.
16#fe02	Requested Packet Interval (RPI) out of range.	The Requested Packet Interval (RPI) specified is invalid for this module. • See the Connection tab for valid RPI values.

Code	String	Explanation and Possible Causes/Solutions
16#fe03		The input connection point has not been set.
16#fe04	Connection Request Error: Invalid input data pointer.	The controller is attempting to set up a connection with the module and has received an error.
16#fe05	Connection Request Error: Invalid input data size.	Either: • The controller is attempting to set up a connection with the module and has received an error. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#fe06		The input force point has not been set.
16#fe07		The output connection point has not been set.
16#fe08	Connection Request Error: Invalid output data pointer.	The controller is attempting to set up a connection with the module and has received an error.

Code	String	Explanation and Possible Causes/Solutions
16#fe09	Connection Request Error: Invalid output data size.	Either: • The controller is attempting to set up a connection with the module and has received an error. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#fe0a		The output force pointer has not been set.
16#fe0b	Invalid symbol string.	Either: • The tag to be consumed on this module is invalid. Verify that the tag is marked as being produced. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic

Code	String	Explanation and Possible Causes/Solutions
		keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#fe0c	Invalid PLC-5 instance number.	The controller is attempting to set up a connection with the PLC-5 and has received an error. Verify that the instance number specified has been properly specified in the PLC-5.
16#fe0d	Tag does not exist in peer controller.	The symbol instance number was found to not be set.
16#fe0e	Automatic Firmware Update in progress.	The module is currently being updated.
16#fe0f	Automatic Firmware Update Failed: Firmware file incompatible with the module.	Firmware supervisor has attempted to update an unsupported module.
16#fe10	Automatic Firmware Update Failed: Firmware file not found.	The firmware file to update the module cannot be found.
16#fe11	Automatic Firmware Update Failed: Firmware file invalid.	The firmware file is corrupted.
16#fe12	Automatic Firmware Update Failed.	An error has occurred while updating the module.
16#fe13	Automatic Firmware Update Failed: Detected Active Connections.	An active connection could not be made to the target module.
16#fe14	Automatic Firmware Update pending: Searching NVS file for appropriate module identity.	The firmware file is currently being read.

Code	String	Explanation and Possible Causes/Solutions
16#fe22		The target-to-originator netparams connection type is invalid.
16#fe23		The target-to-originator netparams connection does not specify whether unicast is allowed.

Module Faults: 16#ff00 - 16#ffff

These are the module faults: 16#ff00 - 16#ffff.

Code	String	Explanation and Possible Causes/Solutions
16#ff00	Connection Request Error: No connection instance.	The controller is attempting to set up a connection with the module and has received an error. Verify that the physical module is the same module type (or is a compatible module) as created in the software. If the module is a 1756-DHRIO module in a remote chassis (connected via a ControlNet network), verify that the network has been scheduled with RSNetWorx software. Even after the network has been scheduled with RSNetWorx for ControlNet software, if you are online and if the 1756-DHRIO module is configured for DH + network only, a #ff00 Module Fault (no connection instance) may occur. The module is properly communicating even though Faulted is displayed as its Status on the Module Properties dialog box. Disregard the error message and fault status and continue.
16#ff01	Connection Request Error: Path to module too long.	The controller is attempting to set up a connection with the module and has received an error. Verify that the path to this module is a valid length.
16#ff04		The remote controller's map instance attempted to access a connection while being in an invalid state.

Code	String	Explanation and Possible Causes/Solutions
16#ff08	Connection Request Error: Invalid path to module.	The controller is attempting to set up a connection with the module and has received an error. Verify that the path to this module is a valid length.
16#ff0b	Module Configuration Invalid: bad format.	Either: • The configuration for the module is invalid. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. The fault may occur even when the module passed the electronic keying test. This may result when Disable Keying or Compatible Module options were used in the module configuration instead of the Exact Match option. Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted. Check the module in use and verify that it exactly matches the module specified in the I/O configuration tree of the Logix Designer application.
16#ff0e	Connection Request Error: No connections accepted to bridge.	The controller is attempting to set up a connection with the module and has received an error.

Specify CIP Messages

The CIP Data Table Read and Write message types transfer data between Logix 5000 controllers.

Select this command	If you want to
CIP Data Table Read	Read data from another controller. The Source and Destination types must match.
CIP Data Table Write	Write data to another controller. The Source and Destination types must match.

Reconfigure an I/O Module

Use the Module Reconfigure message to send new configuration information to an I/O module.

During the reconfiguration, the following occurs:

- Input modules continue to send input data to the controller.
- Output modules continue to control their output devices.

A Module Reconfigure message requires this configuration properties.

In this property	Select
Message Type	Module Reconfigure

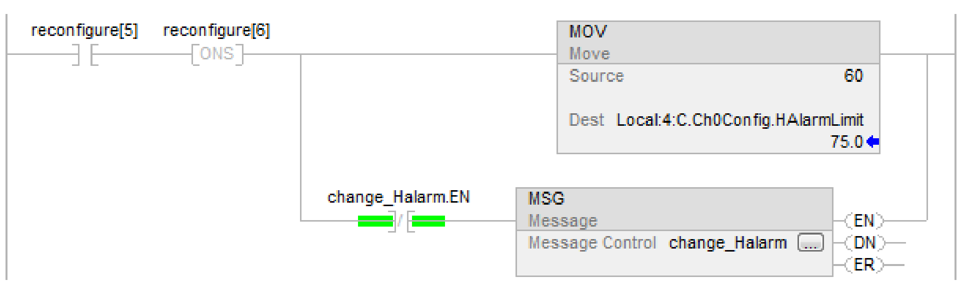
Example

Follow these steps to reconfigure an I/O module.

1. Set the required member of the configuration tag of the module to the new value.
2. Send a Module Reconfigure message to the module.
   When reconfigure[5] is set, set the high alarm to 60 for the local module in slot 4. The Module Reconfigure message then sends the new alarm value to the module. The one shot instruction prevents the rung from sending multiple messages to the module while the reconfigure[5] is on.

Tip: We recommend that you always include an XIO of the MSG.EN bit as an in-series MSG rung precondition.

Relay Ladder



Structured Text

IF reconfigure[5] AND NOT reconfigure[6]THEN Local:4:C.Ch0Config.HAlarmLimit := 60;

IF NOT change_Halarm.EN THEN MSG(change_Halarm);

END_IF; END_IF;

reconfigure[6] := reconfigure[5];

Specify CIP Generic Messages

IMPORTANT: ControlLogix modules have services that can be invoked by using a MSG instruction and choosing the CIP Generic message type.

If you want to	In this property	Type or select	
Perform a pulse test on a digital output module	Message Type	CIP Generic	
	Service Type	Pulse Test	
	Source	tag_name of type INT [5]	
		This array contains	Description
		tag_name[0]	Bit mask of points to test (test only one point at a time)
		tag_name[1]	Reserved, leave 0
		tag_name[2]	Pulse width (hundreds of µ, usually 20)
		tag_name[3]	Zero cross delay for ControlLogix I/O (hundreds of µ, usually 40)
		tag_name[4]	Verify delay
	Destination	Blank	
Get audit value	Message Type	CIP Generic	
	Service Type	Audit Value Get	
	Source Element	Cannot change this field, blank	
	Source Length	Cannot change this field, set to 0 bytes	
	Destination Element	This array contains	Description
		tag_name of type DINT[2] or LINT	This tag contains the Audit Value for the controller. **Important:** Rockwell Automation recommends using the DINT[2] data type to avoid limitations when working with LINT data types in Allen-Bradley® controllers.
Get controller events monitored for changes	Message Type	CIP Generic	
	Service Type	Changes to Detect Get	
	Source Element	Cannot change this field, blank	
	Source Length	Cannot change this field, set to 0 bytes	
	Destination Element	This array contains	Description

If you want to	In this property	Type or select	
		tag_name of type DINT[2] or LINT	This tag represents a bit mask of the changes monitored for the controller. **Important:** Rockwell Automation recommends using the DINT[2] data type to avoid limitations when working with LINT data types in Allen-Bradley controllers.
Set controller events monitored for changes	Message Type	CIP Generic	
	Service Type	Changes to Detect Set	
	Source Element	This array contains	Description
		tag_name of type DINT[2] or LINT	This tag represents a bit mask of the changes monitored for the controller. **Important:** Rockwell Automation recommends using the DINT[2] data type to avoid limitations when working with LINT data types in Allen-Bradley controllers.
	Source Length	Cannot change this field, set to 8 bytes	
	Destination Element	Cannot change this field, blank	
Reset electronic fuses on a digital output module	Message Type	CIP Generic	
	Service Type	Reset Electronic Fuse	
	Source	tag name of type DINT This tag represents a bit mask of the points to reset fuses on.	
	Destination	Leave blank	
Reset latched diagnostics on a digital input module	Message Type	CIP Generic	
	Service Type	Reset Latched Diagnostics (I)	
	Source	tag_name of type DINT This tag represents a bit mask of the points to reset diagnostics on.	
Reset latched diagnostics on a digital output module	Message Type	CIP Generic	
	Service Type	Reset Latched Diagnostics (O)	
	Source	tag_name of type DINT This tag represents a bit mask of the points to reset diagnostics on.	
Unlatch the alarm of an analog input module	Message Type	CIP Generic	
	Service Type	Select which alarm that you want to unlatch.	

If you want to	In this property	Type or select
		• Unlatch All Alarms (I) • Unlatch Analog High Alarm (I) • Unlatch Analog High High Alarm (I) • Unlatch Analog Low Alarm (I) • Unlatch Analog Low Low Alarm (I) • Unlatch Rate Alarm (I)
	Instance	Channel of the alarm to unlatch.
Unlatch the alarm of an analog output module	Message Type	CIP Generic
	Service Type	Select which alarm that you want to unlatch. • Unlatch All Alarms (O) • Unlatch High Alarm (O) • Unlatch Low Alarm (O) • Unlatch Ramp Alarm (O)
	Instance	Channel of the alarm to unlatch.

Get/Set Controller Events Monitored for Changes Bit Definitions

Tag Names	Data Type	Bit Definition
Get Controller Events Monitored for Changes Set Controller Events Monitored for Changes	DINT[0]	Each bit has a specific meaning: 0 Store to removable media through Logix Designer application 1 Online edits were accepted, tested, or assembled 2 Partial import online transaction completed 3 SFC Forces were enabled 4 SFC Forces were disabled 5 SFC Forces were removed 6 SFC Forces were modified 7 I/O Forces were enabled 8 I/O Forces were disabled 9 I/O Forces were removed 10 I/O Forces were changed 11 Firmware update from unconnected source 12 Firmware update via removable media 13 Mode change via workstation 14 Mode change via mode switch 15 A major fault occurred 16 Major faults were cleared 7 Major faults were cleared via mode switch 118 Task properties were modified

Tag Names	Data Type	Bit Definition
		19 Program properties were modified 20 Controller timeslice options were modified 21 Removable media was removed 22 Removable media was inserted 23 Safety signature created 24 Safety signature deleted 25 Safety lock 26 Safety unlock 27 Constant tag value changed 28 Constant tag multiple values changed 29 Constant tag attribute cleared 30 Tag set as constant 31 Custom log entry added
	DINT[1]	32 Change that affects correlation 33 Helps protect signature in Run mode attribute set 34 Helps protect signature in Run mode attribute cleared 35...63 Unused

Tip:

- Selecting the **CIP Generic** message type enables the **Large Connection** option on the **Communication** tab. Use large CIP Generic connections when a message is greater than 480 bytes. 500 bytes is typical, but there are headers at the front of the message. Large CIP connections are for messages up to 3980 bytes.
- The **Large Connection** box is enabled only when the **Connected** box is checked and **CIP Generic** is selected as the message type on the **Configuration** tab.
- The **Large Connection** option is available only in Logix Designer application, version 21.00.00 or later and RSLogix 5000 software, version 20.00.00 or later.

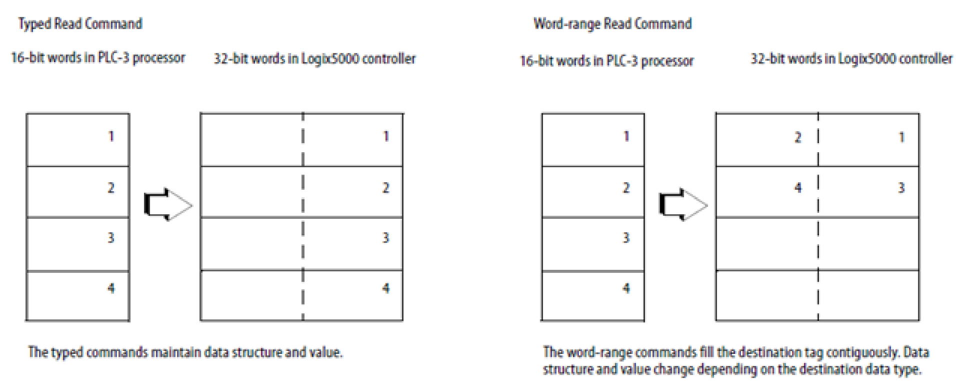
Specify PLC-3 Messages

The PLC-3 message types are designed for PLC-3 processors.

Select this command:	To:
PLC3 Typed Read	Read integer or REAL type data. For integers, this command reads 16-bit integers from the PLC-3 processor and stores them in SINT, INT, or DINT data arrays in the Logix 5000 controller and maintains data integrity.

Select this command:	To:
	This command also reads floating-point data from the PLC-3 and stores it in a REAL data type tag in the Logix 5000 controller.
PLC3 Typed Write	Write integer or REAL type data. This command writes SINT or INT data, to the PLC-3 integer file and maintains data integrity. You can write DINT data as long as it fits within an INT data type (−32,768 ≥ data ≤ 32,767). This command also writes REAL type data from the Logix 5000 controller to a PLC-3 floating-point file.
PLC3 Word Range Read	Read a contiguous range of 16-bit words in PLC-3 memory regardless of data type. This command starts at the address specified as the Source Element and reads sequentially the number of 16-bit words requested. The data from the Source Element is stored, starting at the address specified as the Destination Tag.
PLC3 Word Range Write	Write a contiguous range of 16-bit words from Logix 5000 memory regardless of data type to PLC-3 memory. This command starts at the address specified as the Source Tag and reads sequentially the number of 16-bit words requested. The data from the Source Tag is stored, starting at the address specified as the Destination Element in the PLC-3 processor.

The following diagrams show how the typed and word-range commands differ. The example uses read commands from a PLC-3 processor to a Logix 5000 controller.



Specify PLC-5 Messages

Use the PLC-5 message types to communicate with PLC-5 controllers.

Select this command:	To:
PLC-5 Typed Read	Read 16-bit integer, floating-point, or string type data and maintain data integrity.
PLC-5 Typed Write	Write 16-bit integer, floating-point, or string type data and maintain data integrity.
PLC-5 Word Range Read	Read a contiguous range of 16-bit words in PLC-5 memory regardless of data type. This command starts at the address specified as the Source Element and reads sequentially the number of 16-bit words requested. The data from the Source Element is stored, starting at the address specified as the Destination Tag.
PLC-5 Word Range Write	Write a contiguous range of 16-bit words from Logix 5000 memory regardless of data type to PLC-5 memory. This command starts at the address specified as the Source Tag and reads sequentially the number of 16-bit words requested. The data from the Source Tag is stored, starting at the address specified as the Destination Element in the PLC-5 processor.

Data types for PLC-5 Typed Read and Typed Write messages

The following table shows the data types to use with PLC-5 Typed Read and PLC-5 Typed Write messages.

For this PLC-5 data type:	Use this Logix 5000 data type:
B	INT
F	REAL
N	INT
	DINT (Only write DINT values to a PLC-5 controller if the value is ≥ – 32,768 and ≤ 32,767.)
S	INT
ST	STRING

The Typed Read and Typed Write commands also work with SLC 5/03 processors (OS303 and above), SLC 5/04 processors (OS402 and above), and SLC 5/05 processors.

Specify PLC-2 Messages

The PLC-2 message types are designed for PLC-2 processors.

Select this command:	To:
PLC2 Unprotected Read	Read 16-bit words from any area of the PLC-2 data table or the PLC-2 compatibility file of another processor.

Select this command:	To:
PLC2 Unprotected Write	Write 16-bit words to any area of the PLC-2 data table or the PLC-2 compatibility file of another processor.

The message transfer uses 16-bit words, so make sure the Logix 5000 tag appropriately stores the transferred data, typically as an INT array.

L8 controllers for tables—not bold

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

Compare Instructions

The compare instructions let you compare values by using an expression or a specific compare instruction.

Available Instructions

Ladder Diagram

CMP on page 327	EQ on page 319	GE on page 338	GT on page 330	IsINF on page 345	IsNAN on page 347	LE on page 356	LT on page 348	LIMIT on page 363	MEQ on page 372	NE on page 379

Function Block Diagram

FBD Block

EQ on page 319	GE on page 338	GT on page 330	LE on page 356	LT on page 348	LIMIT on page 363	MEQ on page 372	NE on page 379

Structured Text

Not available

If you want to:	Use this instruction:
compare values based on an expression	CMP
test whether two values are equal	EQ
test whether one value is greater than or equal to a second value	GE
test whether one value is greater than a second value	GT
test whether the source is infinite	IsINF
test whether the source is not a number	IsNAN
test whether one value is less than or equal to a second value	LE
test whether one value is less than a second value	LT
test whether one value is between two other values	LIMIT
pass two values through a mask and test whether they are equal	MEQ
test whether one value is not equal to a second value	NE

Compare values of different data types, such as floating point and integer.

The bold data types indicate optimal data types. An instruction executes at its fastest and with it lowest memory requirements if all the parameters of the instruction use the same optimal data type, typically DINT or REAL.

Equal To (EQ)

This table lists the controllers and applications that support this instruction.

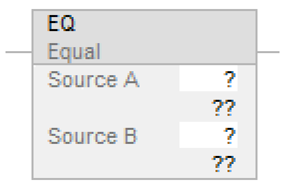
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the EQ instruction and the operator = test whether Source A is equal to Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from EQU to EQ.

Available Languages

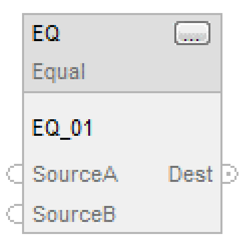
Ladder Diagram



Function Block Diagram

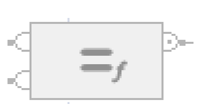
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Tip: Use the operator '=' with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source B
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME	immediate tag	Source to test against Source A

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		TIME32 LTIME DT LDT		

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value tag	String to test against Source B
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
EQU	FBD_COMPARE	tag	EQ structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to test against SourceB
SourceB	REAL	Value to test against SourceA

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is equal to SourceB. Cleared to false when SourceA is not equal to SourceB.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description
SourceA (top)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceB.
SourceB (bottom)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceA

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when SourceA is equal to SourceB. Cleared to false when SourceA is not equal to SourceB.

See FBD Functions on page 862.

Affects Math Status Flags

No

Major/Minor Faults

See *EQ String Compare Flow Chart* for faults.

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	**Action Taken**
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A and Source B are not NANs and Source A is equal to Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See EQU String Compare Flow Chart. If output is false Clear Rung-condition-out to false else Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

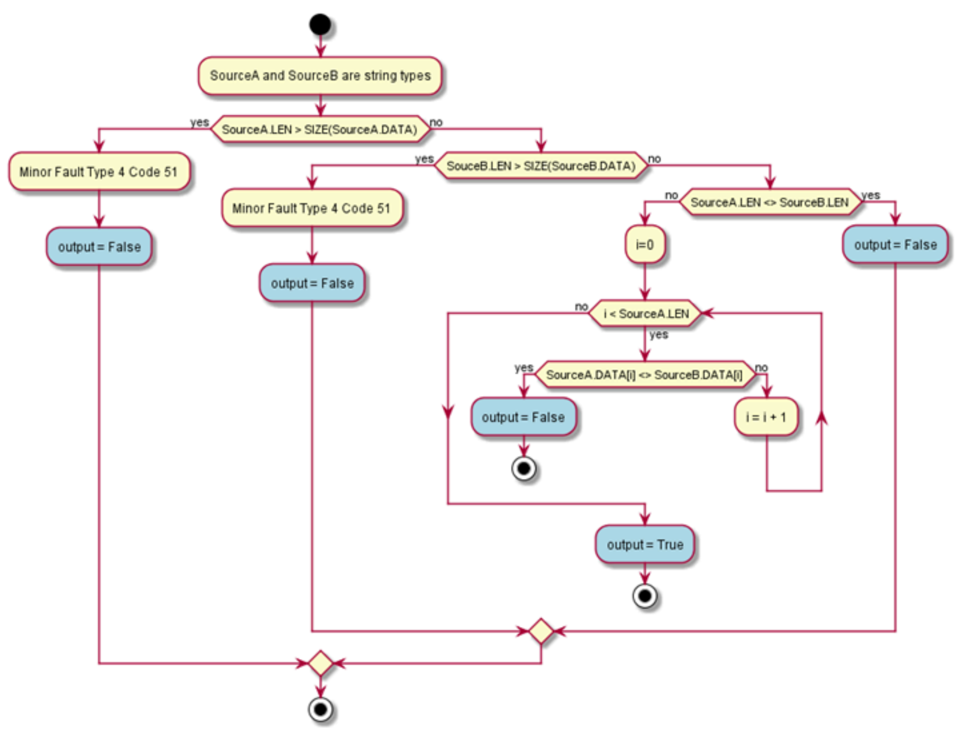
Condition/State	**Action Taken**
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** Set EnableOut to EnableIn If SourceA and SourceB are not NANs and SourceA is equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

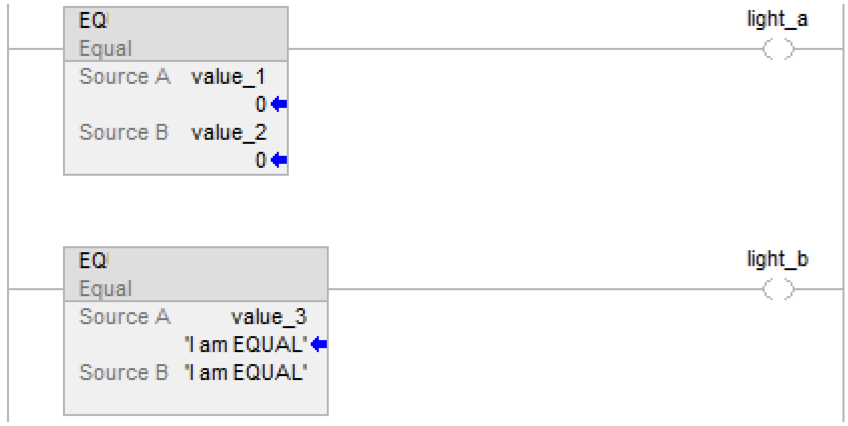
Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** If SourceA and SourceB are not NANs and SourceA is equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

EQ String Compare Flow Chart



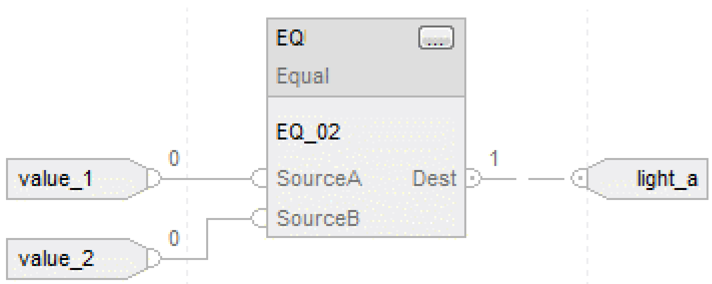
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

if value_1 = value_2 then

light_a := 1;

else

light_a := 0;

end_if;

if value_3 = 'I am EQUAL' then

light_b := 1;

else

light_b := 0;

end_if;

Compare (CMP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Define the CMP expression using operators, tags, and immediate values. Use parentheses ( ) to define sections of more complex expressions.

The advantage of the CMP instruction is that it allows complex expressions in one instruction.

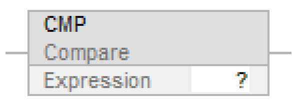
When evaluating the expression all non-REAL operands will be converted to REAL before the calculations are performed if any of the following conditions is true.

- Any operand in the expression is REAL.
- The expression contains SIN, COS, TAN, ASIN, ACOS, ATAN, LN, LOG, DEG or RAD.

There are rules for allowable operators in safety applications. See *Valid Operators.*

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

These are the operands for the CMP instruction.

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

The following is the Ladder Diagram operand.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Expression	SINT INT DINT REAL String type	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type	immediate tag	An expression consisting of tags and/or immediate values separated by operators

Formatting expressions

For each operator used in an expression, one or two operands (tags or immediate values) must be provided. Use the following table to format operators and operands within an expression.

For operators that operate on:	Use this format:	Example
One operand	operator(operand)	ABS(tag)
Two operands	operand_a operator operand_b	tag_b + 5 tag_c AND tag_d (tag_e**2) MOD (tag_f / tag_g)

Determine the order of operation

The instructions performs operations in the expression are in a prescribed order, not necessarily the order they appear. The order of operation can be specified by grouping terms within parentheses, forcing the instruction to perform an operation within the parentheses ahead of their operations.

Operations of equal order are performed from left to right.

Order	Operation
1	( )
2	ABS, ACOS, ASIN, ATAN, COS, DEG, BCD_TO, IsINF, IsNAN, LN, LOG, RAD, SIN, SQRT, TAN, TO_BCD, TRUNC
3	**
4	- (negate), NOT, !
5	*, /, MOD

Order	Operation
6	- (subtract), +
7	AND
8	XOR
9	OR
10	<, <=, >, >=, =, <>
11	&&
12	^^
13	||

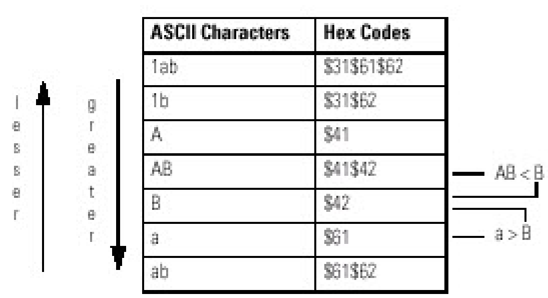
Using strings in an expression

To use strings of ASCII characters in an expression, follow these guidelines:

- An expression can compare two string tags.
- ASCII characters cannot be entered directly into the expression.
- The following operands are permitted:

Operator	Description
=	Equal
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
<>	Not equal

- Strings are equal if their characters match.
- ASCII characters are case-sensitive. Uppercase A ($41) is not equal to lowercase a ($61).
- The hexadecimal values of the characters determine if one string is less than or greater than another string.
- When the two strings are sorted as in a telephone directory, the order of the strings determine which one is greater.



ASCII Characters	Hex Codes
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

Affects Math Status Flags

Controllers	Affects Math Status Flags

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	The CMP instruction affects the math status flags if the expression contains an operator (for example, +, −, *, /) that affects the math status flags.

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

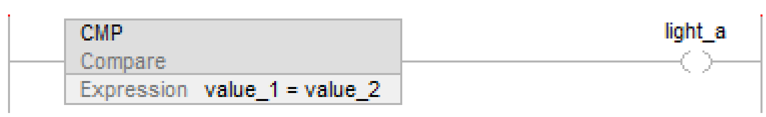
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in if expression evaluates to false Rung-condition-out is cleared to false
Postscan	N/A

Example

Ladder Diagram



If value_1 is equal to value_2, light_a is set to true. If value_1 is not equal to value_2, light_a is cleared to false.

Greater Than (GT)

This table lists the controllers and applications that support this instruction.

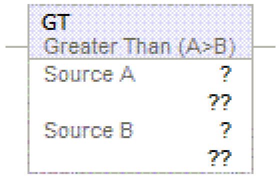
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Greater Than (GT) instruction and the operator > tests whether Source A is greater than Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from GRT to GT.

Available Languages

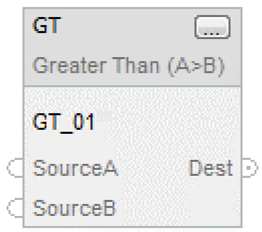
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator > with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source B
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source A

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value tag	String to test against Source B
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
GT	FBD_COMPARE	tag	GT structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to test against SourceB
SourceB	REAL	Value to test against SourceA

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is greater than SourceB. Cleared to false when SourceA is not greater than SourceB.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceB
SourceB (bottom)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceA

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	BOOL	Set to true when SourceA is greater than SourceB. Cleared to false when SourceA is not greater than SourceB.

See FBD Functions on page 862.

Affects Math Status Flags

No

Major/Minor Faults

See GT String Compare Flow Chart on page 338 for faults.

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A and Source B are not NANs and Source A is greater than Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See GT String Compare Flow Chart on page 338 If output is false Clear Rung-condition-out to false else Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** Set EnableOut to EnableIn If SourceA and SourceB are not NANs and SourceA is greater than SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

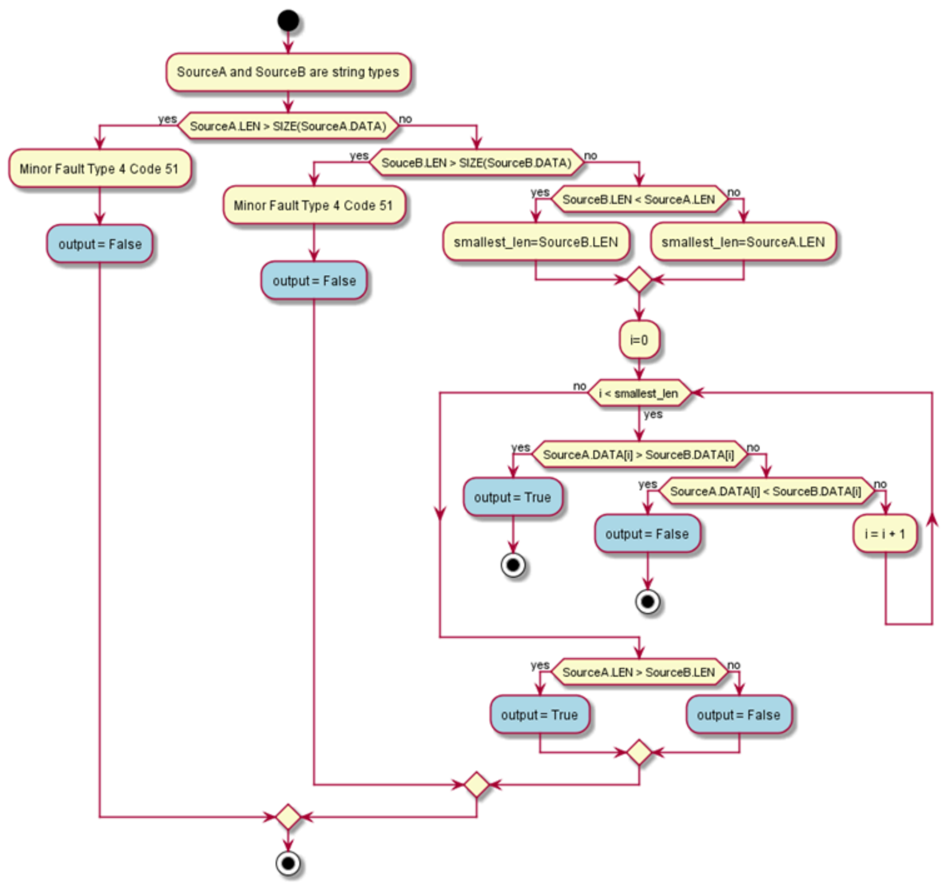
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** If SourceA and SourceB are not NANs and SourceA is greater than SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

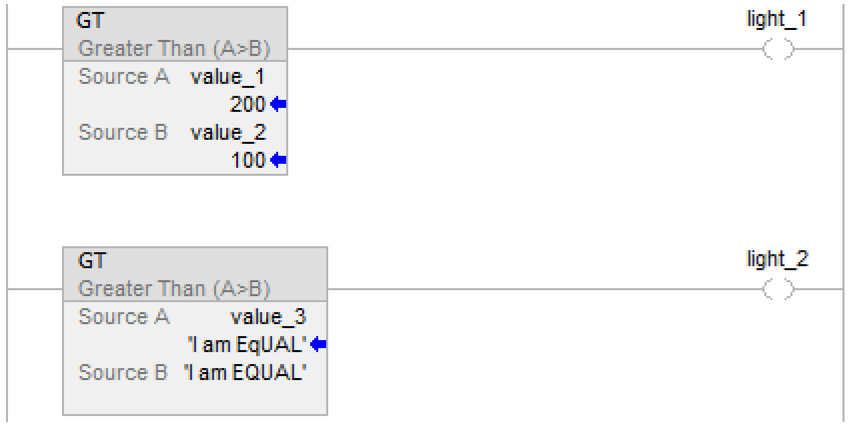
GT String Compare Flow Chart

SourceA.LEN and SourceB.LEN are handled as unsigned values.



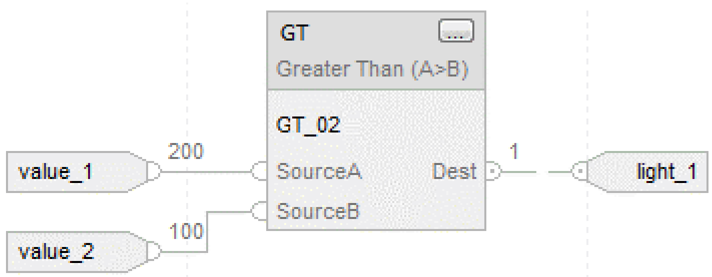
Example

Ladder Diagram

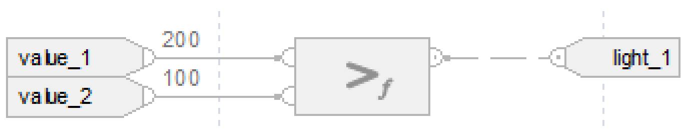


Function Block Diagram

FBD Block



FBD Function



Structured Text

if value_1 > value_2 then

light_1 := 1;

else

light_1 := 0;

end_if;

if value_3 > 'I am EQUAL' then

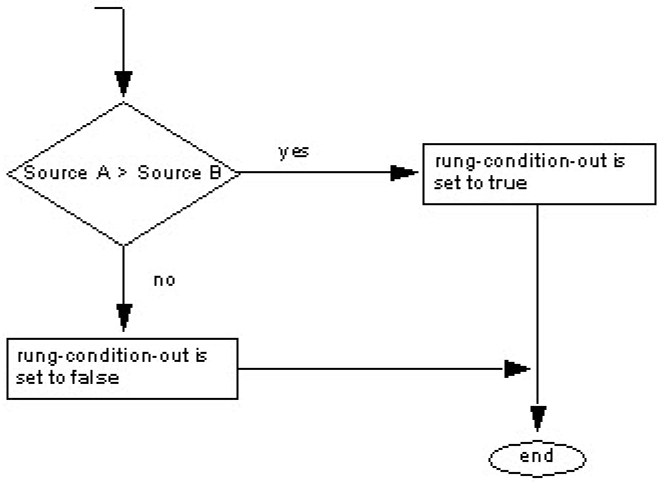
light_2 := 1;

else

light_2 := 0;

end_if;

GT Flow Chart (True)



Greater Than or Equal To (GE)

This table lists the controllers and applications that support this instruction.

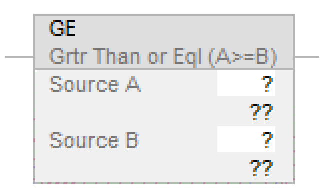
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Greater Than or Equal To (GE) instruction and the operator $\geq$ test whether Source A is greater than or equal to Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from GEQ to GE.

Available Languages

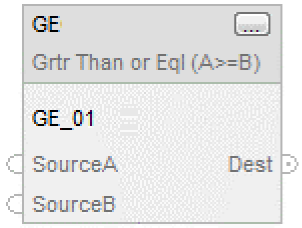
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator ≥ with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions. on page 851.

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT	immediate tag	Value to test against Source B

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT		
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source A

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value	String to test against Source B

Operand	Data Type	Format	Description
		tag	
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
GE	FBD_COMPARE	tag	GE structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to test against SourceB
SourceB	REAL	Value to test against SourceA

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is greater than or equal to SourceB. Cleared to false when SourceA is less than SourceB.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description
SourceA (top)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceB.

Input Operands (Left Pins)	Data Type	Description
SourceB (bottom)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceA.

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when SourceA is greater than or equal to SourceB. Cleared to false when SourceA is less than SourceB.

See *FBD Functions* on page 862 .

Affects Math Status Flags

No

Major/Minor Faults

See *GE String Compare Flow Chart* below for faults.

See *Index Through Arrays* on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A and Source B are not NANs and Source A is greater than or equal to Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See GEQ String Compare Flow Chart. If output is false Clear Rung-condition-out to false

Condition/State	Action Taken
	else Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** Set EnableOut to EnableIn If SourceA and SourceB are not NANs and SourceA is greater than or equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

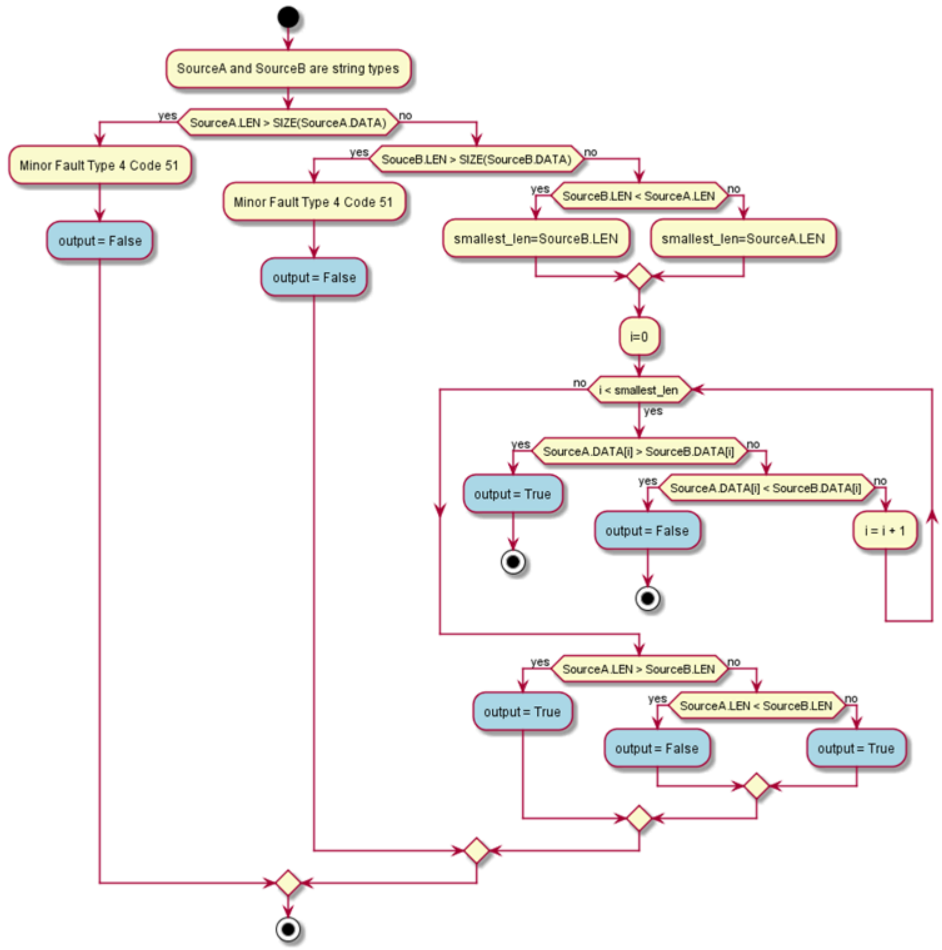
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** If SourceA and SourceB are not NANs and SourceA is greater than or equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

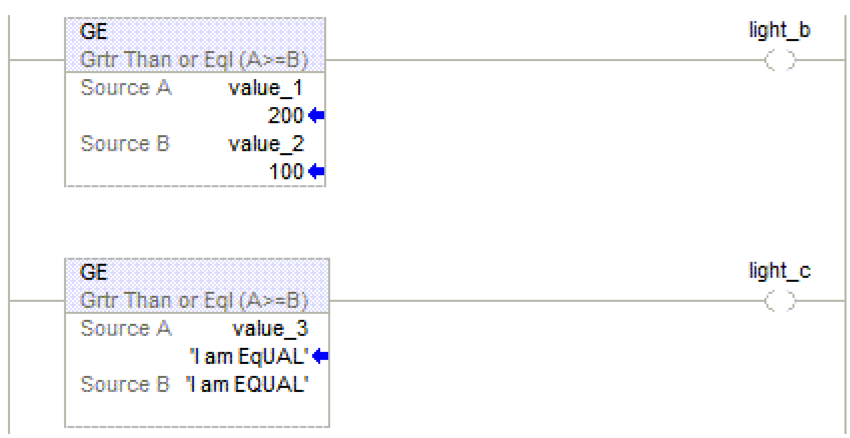
GE String Compare Flow Chart

SourceA.LEN and SourceB.LEN are handled as unsigned values.



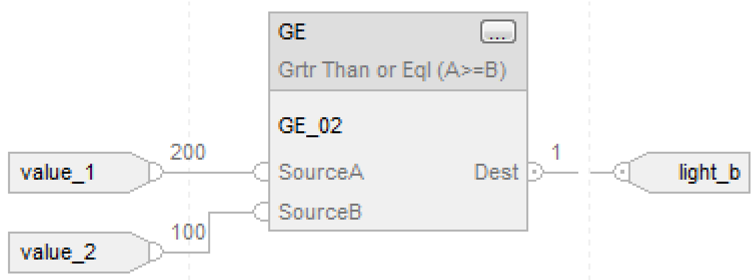
Example

Ladder Diagram

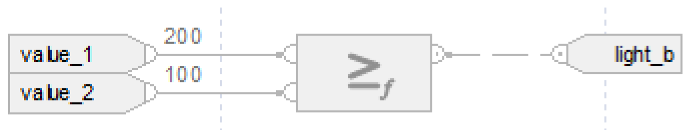


Function Block Diagram

FBD Block



FBD Function



Structured Text

```
if value_1 >= value_2 then

        light_b := 1;

else

        light_b := 0;

end_if;

if value_3 >= 'I am EQUAL' then

        light_c := 1;

else

        light_c := 0;

end_if;
```

Is Infinity (IsINF)

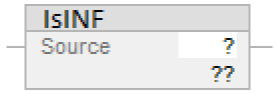
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Is Infinity (IsINF) instruction tests whether the Source is infinity.

Available Languages

Ladder Diagram



Function Block Diagram

This instruction is not available in Function Block Diagram.

Structured Text

This instruction is not available in structured text.

Operands

Ladder Diagram

Operand	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Format	Description
Source	REAL LREAL	tag	Value to test against Infinity.

Affects Math Status Flags

No

Major/Minor Faults

This instruction does not generate any major/minor faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	If Source is +INF or -INF. Set Rung-condition-out to true else Clear Rung-condition-out to false.
Postscan	N/A

Examples

Ladder Diagram

Is Not a Number (IsNAN)

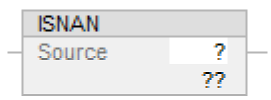
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Is Not a Number (IsNAN) instruction tests whether the source is not a number.

Available Languages

Ladder Diagram



Function Block Diagram

This instruction is not available in Function Block Diagram.

Structured Text

This instruction is not available in structured text.

Operands

Ladder Diagram

Operand	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Format	Description
Source	REAL LREAL	tag	Value to test against Infinity

Affects Math Status Flags

No

Major/Minor Faults

This instruction does not generate any major/minor faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	If Source is NAN. Set Rung-condition-out to true else Clear Rung-condition-out to false.
Postscan	N/A

Examples

Ladder Diagram

Less Than (LT)

This table lists the controllers and applications that support this instruction.

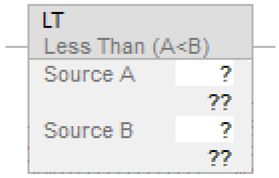
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Less Than (LT) instruction and the operator < tests Source A is less than Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from LES to LT.

Available Languages

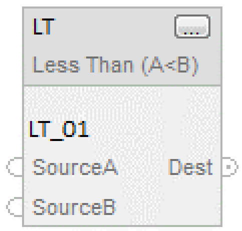
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator < with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions. on page 851

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source B
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source A

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value tag	String to test against Source B
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
LT	FBD_COMPARE	tag	LT structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to test against SourceB
SourceB	REAL	Value to test against SourceA

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is less than SourceB. Cleared to false when SourceA is not less than SourceB.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT	Value to test against SourceB.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
	INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	
SourceB (bottom)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceA.

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when SourceA is less than SourceB. Cleared to false when SourceA is not less than SourceB.

See FBD Functions on page 862.

Affects Math Status Flags

No

Major/Minor Faults

See LES String Compare Flow Chart below for faults.

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A and Source B are not NANs and Source A is less than Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See LES String Compare Flow Chart. If output is false Clear Rung-condition-out to false else Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** If SourceA and SourceB are not NANs and SourceA is less than SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

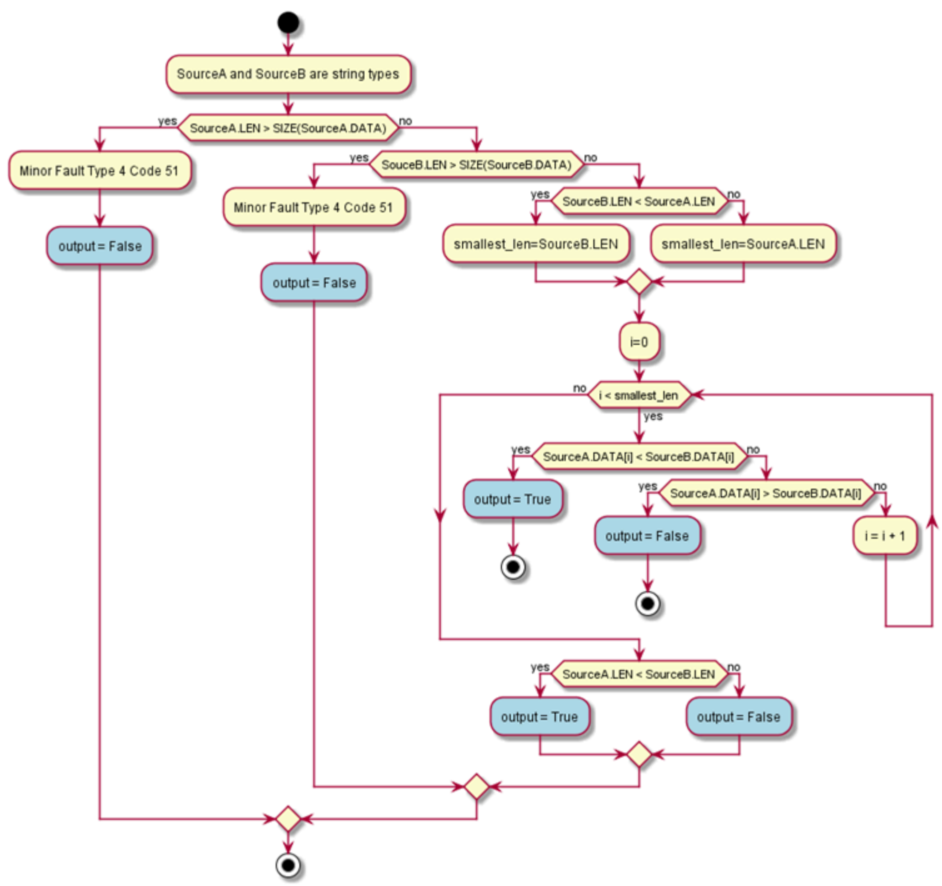
Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** Set EnableOut to EnableIn

Condition/State	Action Taken
	If SourceA and SourceB are not NANs and SourceA is less than SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

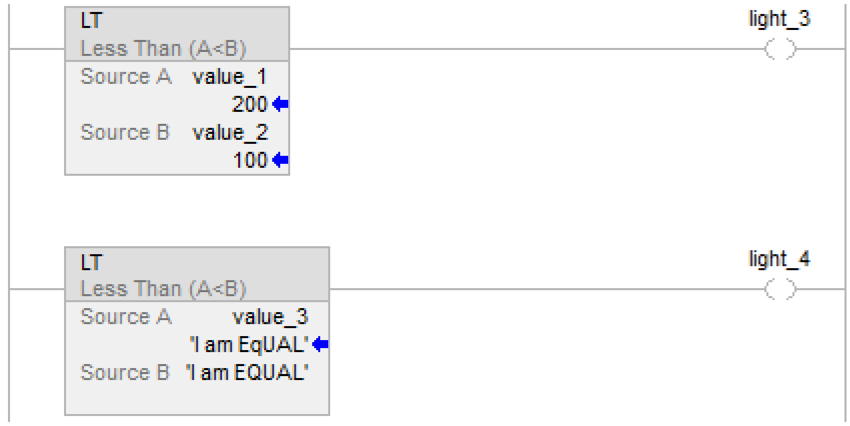
LT String Compare Flow Chart

SourceA.LEN and SourceB.LEN are handled as unsigned values.



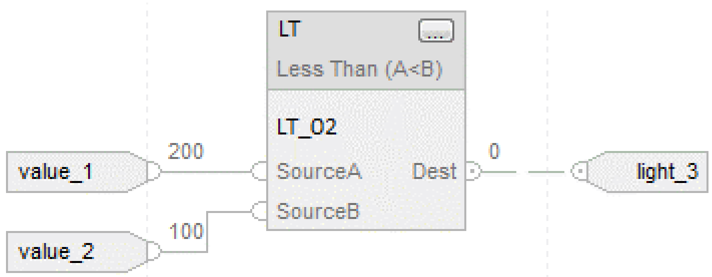
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

```
if value_1 < value_2 then

    light_3 := 1;

else

    light_3 := 0;

end_if;

if value_3 < 'I am EQUAL' then

    light_4 := 1;

else

    light_4 := 0;

end_if;
```

Less Than or Equal To (LE)

This table lists the controllers and applications that support this instruction.

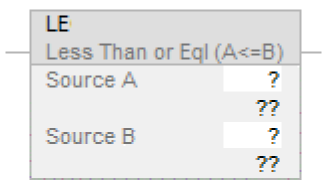
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Less Than or Equal To (LE) instruction and the operator <= tests whether Source A is less than or equal to Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from LES to LE.

Available Languages

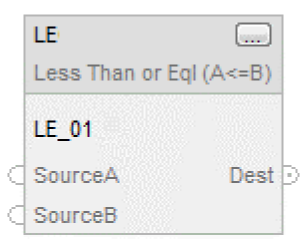
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator <= with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source B
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT	immediate tag	Value to test against Source A

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		REAL LREAL TIME TIME32 LTIME DT LDT		

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value tag	String to test against Source B
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
LE	FBD_COMPARE	tag	LE structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.

Input Members	Data Type	Description
SourceA	REAL	Value to test against SourceB
SourceB	REAL	Value to test against SourceA

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is less than or equal to SourceB. Cleared to false when SourceA is greater than SourceB.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceB.
SourceB (bottom)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceA.

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when SourceA is less than or equal to SourceB. Cleared to false when SourceA is greater than SourceB.

See FBD Functions on page 862.

Affects Math Status Flags

No

Major/Minor Faults

See *LE String Compare Flow Chart* for faults.

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A and Source B are not NANs and Source A is less than or equal to Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See *LE String Compare Flow Chart.* If output is false Clear Rung-condition-out to false else Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** Set EnableOut to EnableIn

Condition/State	Action Taken
	If SourceA and SourceB are not NANs and SourceA is less than or equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

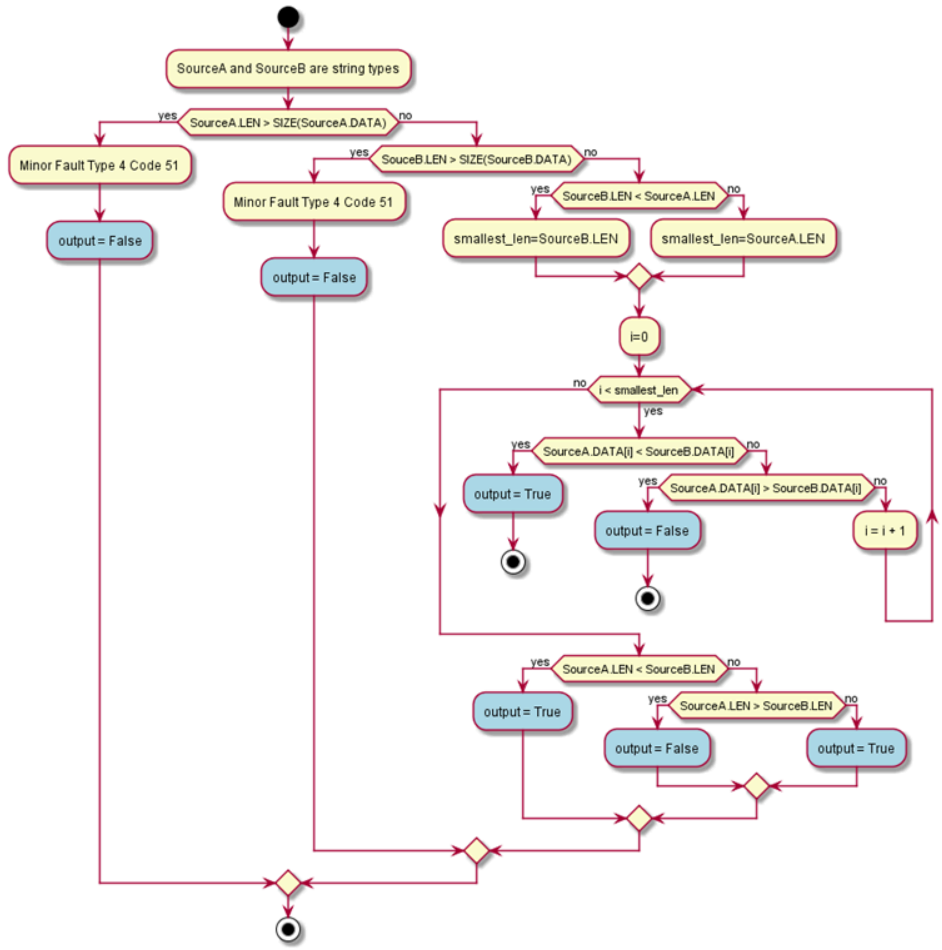
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** If SourceA and SourceB are not NANs and SourceA is less than or equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

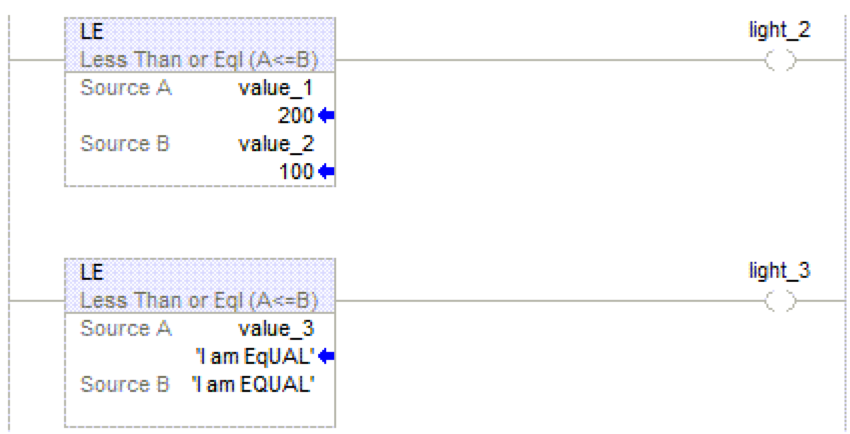
LE String Compare Flow Chart

SourceA.LEN and SourceB.LEN are handled as unsigned values.



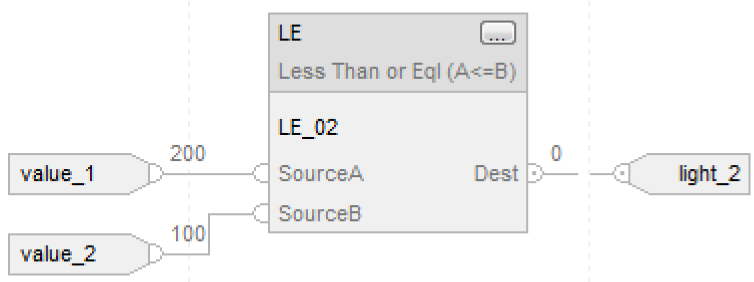
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

```
if value_1 <= value_2 then

        light_2 := 1;

else

        light_2 := 0;

end_if;

if value_3 <= 'I am EQUAL' then

        light_3 := 1;

else

        light_3 := 0;

end_if;
```

Limit (LIMIT)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

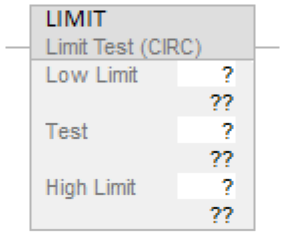
The LIMIT instruction tests if the Test value is within the range of the Low and High Limits as indicated in the LIMIT Flow Chart (True).

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from LIM to LIMIT.

If any operand is Not A Number (NAN), the .EnableOut is cleared to false.

Available Languages

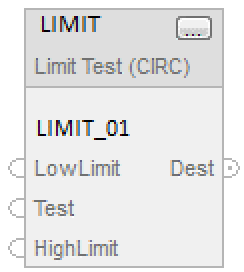
Ladder Diagram



Function Block Diagram

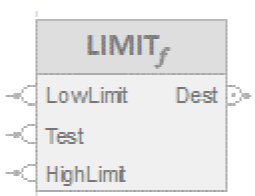
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Low Limit	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of lower limit.
Test	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to test against limits.
High Limit	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of upper limit.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
LIMIT	FBD_LIMIT	tag	LIMIT structure

FBD_LIMIT Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
LowLimit	REAL	Value of lower limit.
Test	REAL	Value to test against limits.
HighLimit	REAL	Value of upper limit.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true if Limit test is true. Cleared to false if Limit test is false.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Low Limit	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value of lower limit
Test	SINT INT DINT LINT	Value to test against limits.

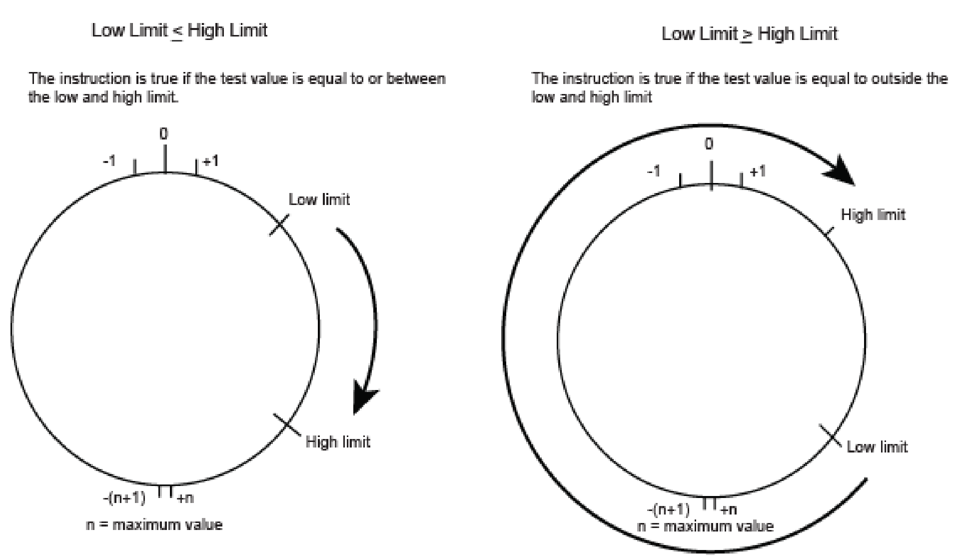
Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
	USINT UINT UDINT ULINT REAL LREAL	
High Limit	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value of upper limit.

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true if Limit test is true. Cleared to false if Limit test is false.

See FBD Functions on page 862.

Operation

This section illustrates the operation for the LIMIT instruction.



If Low Limit:	And if the test value is:	Then the EnableOut is:
< or = to High Limit	equal to or between limits	true
	not equal to or outside limits	false
> High Limit	equal to or outside limits	true
	not equal to or inside limits	false

Signed integers transition from the maximum positive number to the maximum negative number when the most significant bit is true. For example, in 16-bit integers (INT type), the maximum positive integer is 32,767, which is represented in hexadecimal as 16#7FFF (bits 0 through 14 are all true). If that number increments by one, the result is 16#8000 (bit 15 is true). For signed integers, hexadecimal 16#8000 is equal to -32,768 decimal. Incrementing from this point on until all 16 bits are set ends up at 16#FFFF, which is equal to -1 decimal.

This can be shown as a circular number line. The LIMIT instruction starts at the Low Limit and increments clockwise until it reaches the High Limit. Any Test value in the clockwise range from the Low Limit to the High Limit sets the EnableOut to true. Any Test value in the clockwise range from the High Limit to the Low Limit clears the EnableOut to false.

If any operand is Not A Number (NAN), the .EnableOut is cleared to false.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	See *LIMIT Flow Chart (True)* If output is true Set Rung-condition-out to true. else Clear Rung-condition-out to false.
Postscan	N/A

Function Block Diagram

FBD Block

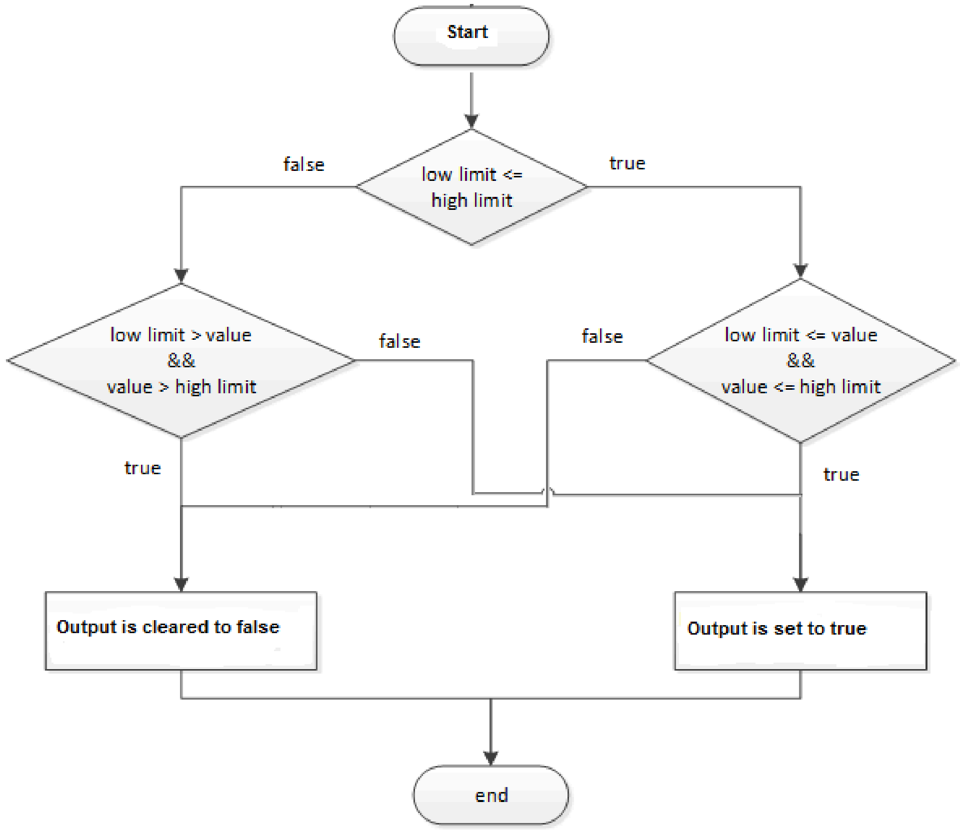
Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Set EnableOut to EnableIn. See LIM Flow Chart (True) Dest = output
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	See LIMIT Flow Chart (True) Dest = output
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

LIMIT Flow Chart (True)

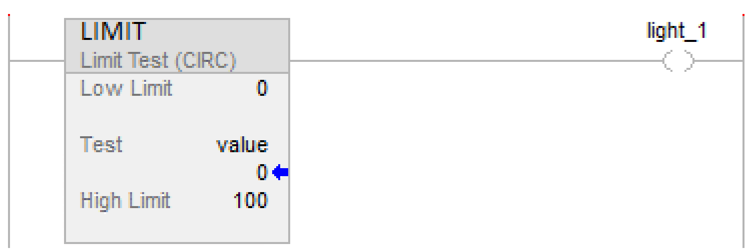


Examples

Example 1: Low Limit <= High Limit

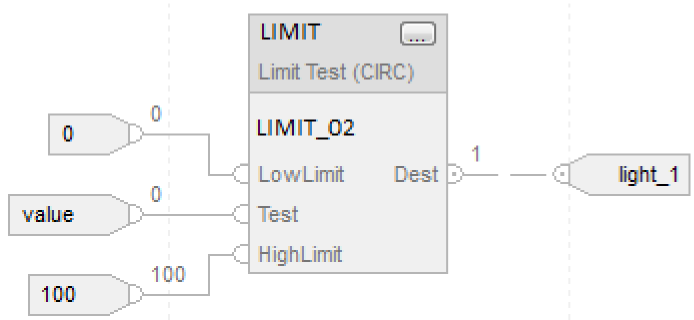
When Test value is equal to or greater than Low Limit, and Test value is less than or equal to High Limit, light_1 is set.

Ladder Diagram

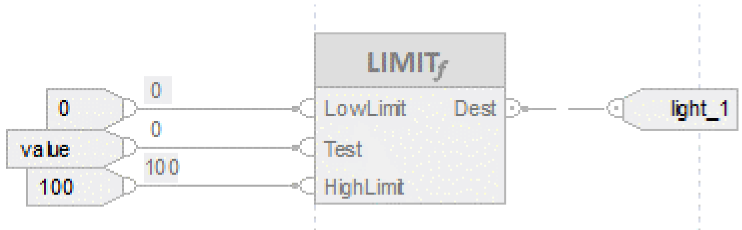


Function Block Diagram

FBD Block



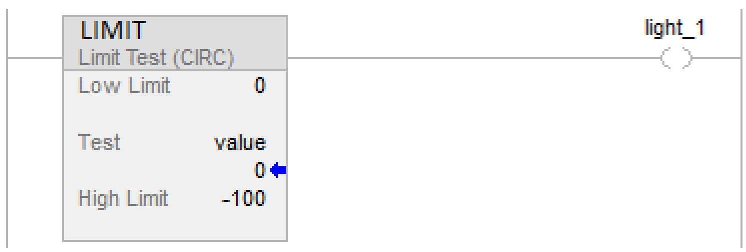
FBD Function



Example 2: Low Limit > High Limit

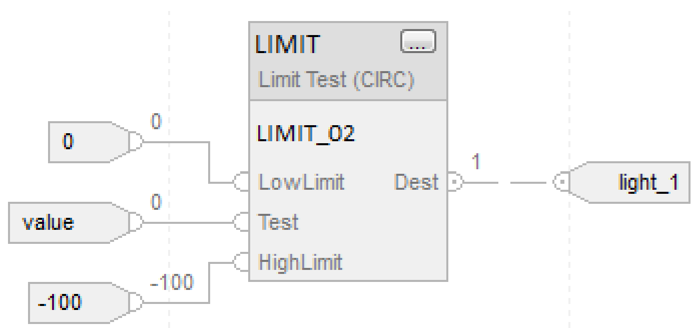
When value > or = to 0 or value < or = to -100, set light_1 to true. If value < 0 and value > -100, clear light_1 to false.

Ladder Diagram

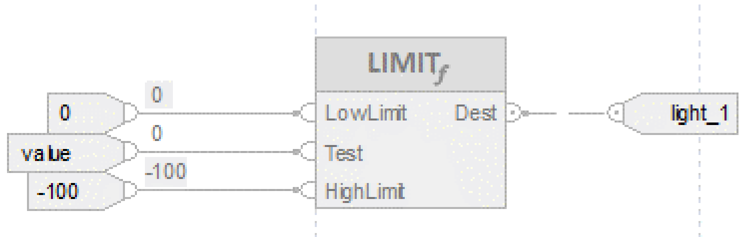


Function Block Diagram

FBD Block



FBD Function



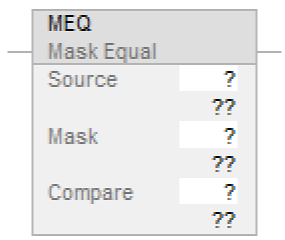
Mask Equal To (MEQ)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The MEQ instruction passes the Source and Compare values through a Mask and compares the results.

Available Languages

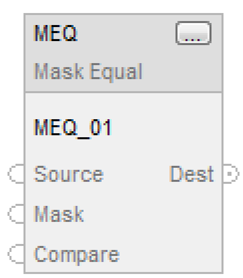
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	immediate tag	Value to test against Compare.
Mask	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	immediate tag	Which bits to block or pass.
Compare	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	immediate tag	Value to test against Source.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
MEQ	FBD_MASK_EQUAL	tag	MEQ structure

FBD_MASK_EQUAL Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	DINT	Value to test against Compare.
Mask	DINT	Defines which bits to block, such as mask.
Compare	DINT	Value to test against Source.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	BOOL	Set to true when result is true. Cleared to false when result is false.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT UINT UDINT ULINT	Value to test against Compare.
Mask	SINT INT DINT LINT	Which bits to block or pass.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
	USINT UINT UDINT ULINT	
Compare	SINT INT DINT LINT USINT UINT UDINT ULINT	Value to test against Source.
	A SINT or INT tag is converted to a DINT value by zero-fill.	

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when result is true. Cleared to false when result is false.

See FBD Functions on page 862.

Operation

A "1" in the mask means the data bit is passed. A "0" in the mask means the data bit is blocked. Typically, the Source, Mask, and Compare values are all the same data type.

If using SINT or INT data type, the instruction fills the upper bits of that value with 0s so that it is the same size as the DINT data type.

Enter an immediate mask value

When entering a mask, the programming software defaults to decimal values. To enter a mask using another format, precede the value with the correct prefix.

Prefix	Description
16#	hexadecimal, such as 16#0F0F
8#	octal, such as 8#16
2#	binary, such as 2#00110011

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Refer to MEQ Flow Chart (True). If output is true Set Rung-condition-out to true else Clear Rung-condition-out to false
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Set EnableOut to EnableIn. Refer to *MEQ Flow Chart (True).* If output is true Set Dest to true else Clear Dest to false
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

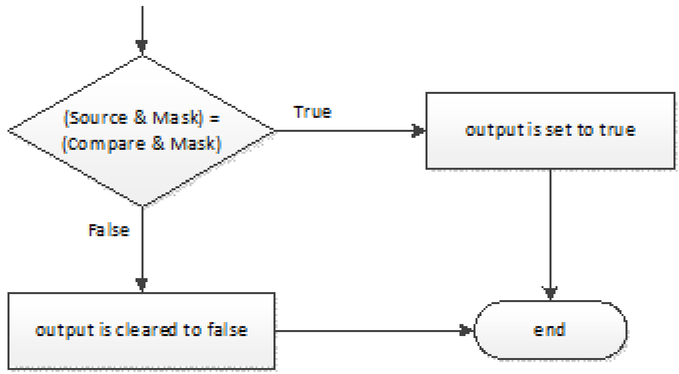
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Refer to *MEQ Flow Chart (True).* If output is true

Condition/State	Action Taken
	Set Dest to true else Clear Dest to false
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

MEQ Flow Chart (True)



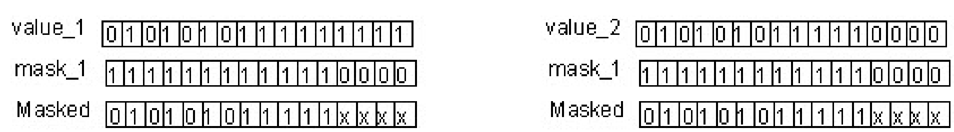
Examples

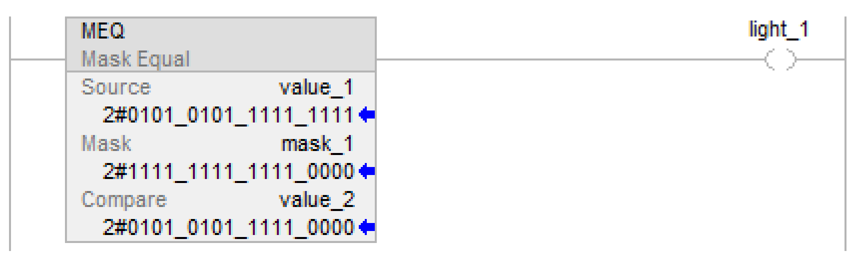
Example 1

If the masked value_1 is equal to the masked value_2, set light_1 to true. If the masked value_1 is not equal to the masked value_2, clear light_1 to false.

This example shows that the masked values are equal. A 0 in the mask restrains the instruction from comparing that bit (indicated by an x in the example).

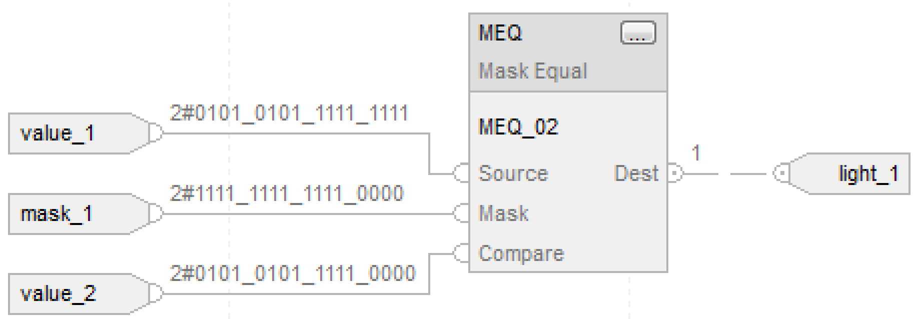
Ladder Diagram



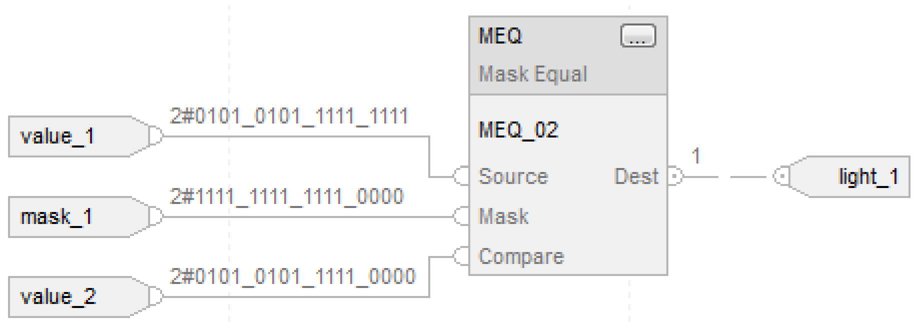


Function Block Diagram

FBD Block



FBD Function

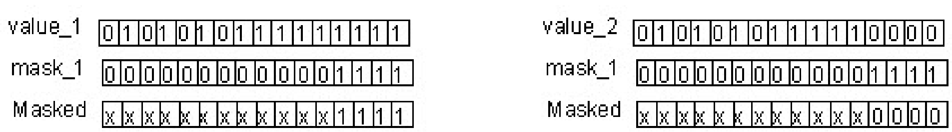


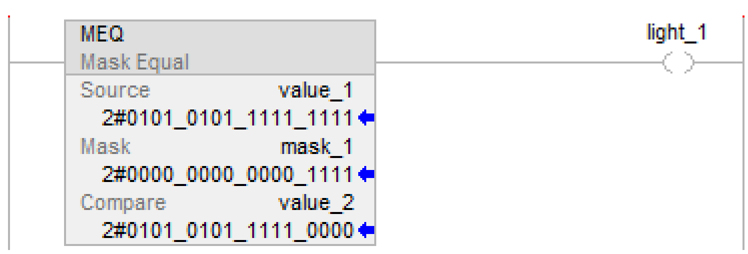
Example 2

If the masked value_1 is equal to the masked value_2, set light_1 to true. If the masked value_1 is not equal to the masked value_2, clear light_1 to false.

This example shows that the masked values are not equal. A 0 in the mask restrains the instruction from comparing that bit (indicated by an x in the example).

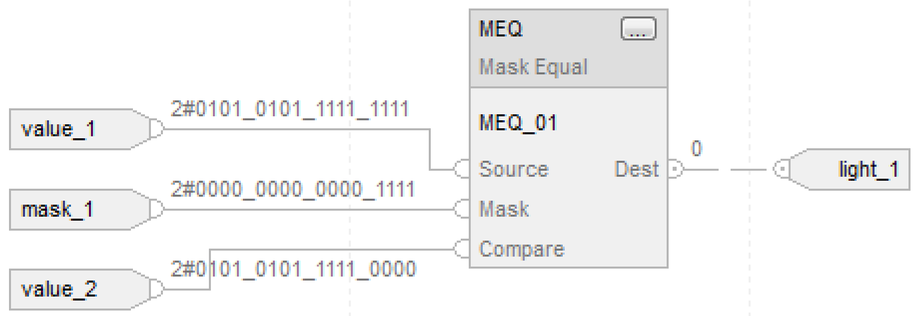
Ladder Diagram



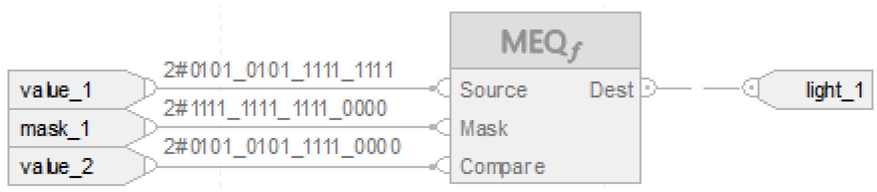


Function Block Diagram

FBD Block



FBD Function



Not Equal To (NE)

This table lists the controllers and applications that support this instruction.

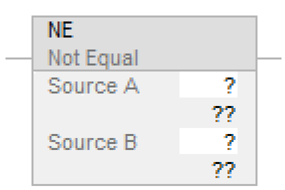
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Not Equal To (NE) instruction and the <> operator tests whether Source A is not equal to Source B.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from NEQ to NE.

Available Languages

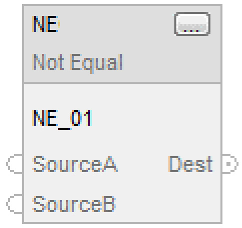
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the <> operator with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Ladder Diagram

Numeric Comparison

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT	immediate tag	Value to test against Source B

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		ULINT REAL LREAL TIME TIME32 LTIME DT LDT		
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	immediate tag	Value to test against Source A

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

String Comparison

Tip: Immediate string literals are only applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.

Operand	Data Type	Format	Description
Source A	String type	immediate literal value tag	String to test against Source B

Operand	Data Type	Format	Description
Source B	String type	immediate literal value tag	String to test against Source A

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
NEQ	FBD_COMPARE	tag	NE structure

FBD_COMPARE Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to test against SourceB.
SourceB	REAL	Value to test against SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction is enabled.
Dest	BOOL	Set to true when SourceA is not equal to SourceB. Cleared to false when SourceA is equal to SourceB.

FBD Function

Tip: FBD Function is applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description
SourceA (top)	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to test against SourceB
SourceB (bottom)	SINT	Value to test against SourceA.

Input Operands (Left Pins)	Data Type	Description
	INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	

Output Operand (Right Pin)	Data Type	Description
Dest	BOOL	Set to true when SourceA is not equal to SourceB. Cleared to false when SourceA is equal to SourceB.

SeeFBD Functions on page 862FBD Functions

Affects Math Status Flags

No

Major/Minor Faults

See *NE String Compare Flow Chart* for faults.

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	**Numeric compare:** If Source A or Source B is NAN or Source A is not equal to Source B. Set Rung-condition-out to true else Clear Rung-condition-out to false.
	String compare: See *NE String Compare Flow Chart.* If output is false Clear Rung-condition-out to false else

Condition/State	Action Taken
	Set Rung-condition-out to true
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	**Numeric compare:** Set EnableOut to EnableIn If SourceA or SourceB is NAN or SourceA is not equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

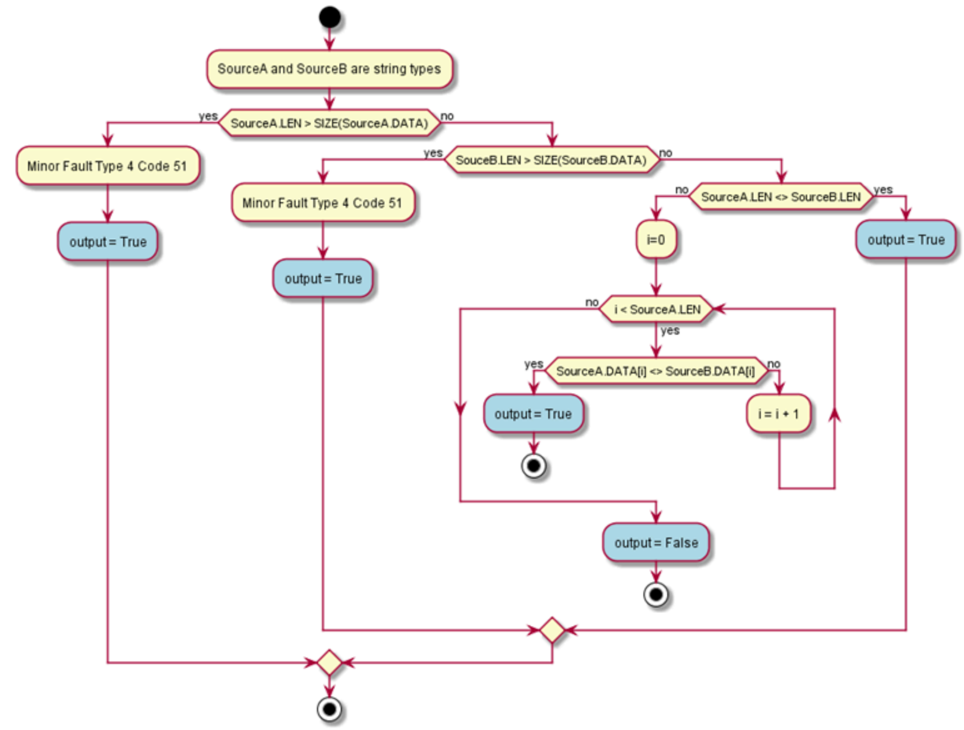
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	**Numeric compare:** If SourceA or SourceB is NAN or SourceA is not equal to SourceB. Set Dest to true else Clear Dest to false.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

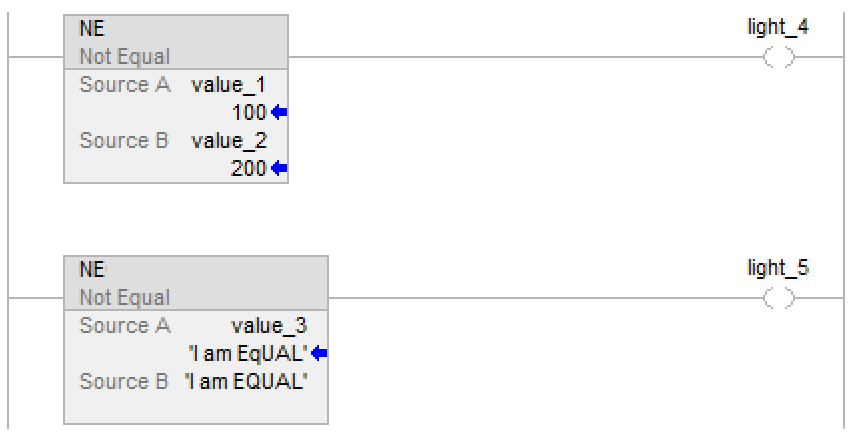
NE String Compare Flow Chart

SourceA.LEN and SourceB.LEN are handled as unsigned values.



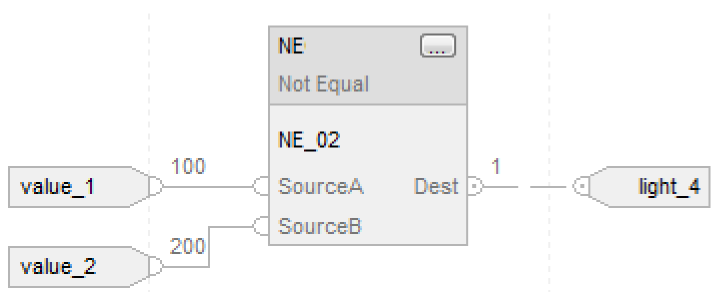
Examples

Ladder Diagram

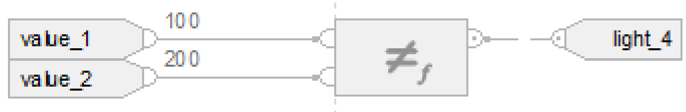


Function Block Diagram

FBD Block



FBD Function



Structured Text

```
if value_1 <> value_2 then

  light_4 := 1;

else

  light_4 := 0;

end_if;

if value_3 <> 'I am EQUAL' then

  light_5 := 1;

else

  light_5 := 0;

end_if;
```

Valid operators

This table lists the valid operators.

Operator	Description	Allowed in					
		Array Index	FSC	CMP	CMP	CPT	Safety
+	add	X	X	X	X	X	X
-	subtract/neg ate	X	X	X	X	X	X
*	multiply	X	X	X	X	X	X
/	divide	X	X	X	X	X	X
=	equal		X	X			X
<	less than		X	X			X
<=	less than or equal		X	X			X
>	greater than		X	X			X
>=	greater than or equal		X	X			X
<>	not equal		X	X			X

Operator	Description	Allowed in					
		Array Index	FSC	CMP	CMP	CPT	Safety
**	exponent (x to y)		X	X	X	X	
&&	Logical AND		X	X			X
||	Logical OR		X	X			X
^^	Logical XOR		X	X			X
!	Logical NOT		X	X			X
ABS	absolute value		X	X	X	X	X
ACOS	arc cosine		X	X	X	X	X
AND	bitwise AND	X	X	X	X	X	X
ASIN	arc sine		X	X	X	X	X
ATAN	arc tangent		X	X	X	X	X
ATAN2	two-argument arctangent		X	X	X	X	X
COS	cosine		X	X	X	X	X
DEG	radians to degrees		X	X	X	X	
BCD_TO	BCD to integer	X	X	X	X	X	
IsINF	Is infinity		X	X			X
IsNAN	Is not a number		X	X			X
LN	natural log		X	X	X	X	
LOG	log base 10		X	X	X	X	
MOD	modulo-divide		X	X	X	X	X
NOT	bitwise NOT	X	X	X	X	X	X
OR	bitwise OR	X	X	X	X	X	X
RAD	degrees to radians		X	X	X	X	
SIN	sine		X	X	X	X	X
SQRT	square root	X	X	X	X	X	
TAN	tangent		X	X	X	X	X
TOD	integer to BCD	X	X	X	X	X	
TRUNC	truncate		X	X	X	X	
XOR	bitwise exclusive OR	X	X	X	X	X	X

Expressions

Expressions are implemented in the Logix Designer application to be passed in as an operand expression to an instruction, or to specify a variable index, as a subscript expression, in an array. These sections describe the differences between the two.

The maximum length for an operand expression is 4096 characters.

Operand Expressions

Operand expressions are provided as an operand to the following instructions: CPT, FAL, FSC, and CMP. Each of these instructions documents which operators are allowed in the expression and their precedence. CPT and FAL have identical operators and precedence. CMP and FSC have a slightly expanded operator list and therefore a different precedence list of operators.

Subscript Expressions

You can also use a subscript expression to compute an array subscript. Subscripts are processed differently than operands. See the Array Index column in the table above for a list of operators that function as subscript expressions.

What is zero fill?

There are two ways a smaller integer type can be converted to a larger one:

- Zero fill
- Sign extension

The method employed depends on the instruction that is using the operand.

For zero-fill, all bits above the range of the smaller type are filled with 0.

For example, SINT: 16#87 = -121 converted to a DINT yields 16#00000087 = 135

For sign-extension, all bits above the range of the smaller type are filled with the sign bit of the smaller type.

For example:SINT: 16#87 = -121 converted to a DINT yields 16#FFFFFF87 = -121

Compute/Math Instructions

The compute/math instructions evaluate arithmetic operations using an expression or a specific arithmetic instruction.

Available Instructions

Ladder Diagram

CPT on page 401	ADD on page 395	SUB on page 433	MUL on page 417	DIV on page 405	MOD on page 411	SQRT on page 427	NEG on page 422	ABS on page 390

Function Block Diagram

FBD Block

ADD on page 395	SUB on page 433	MUL on page 417	DIV on page 405	MOD on page 411	SQRT on page 427	NEG on page 422	ABS on page 390

FBD Function

ADD on page 395	SUB on page 433	DIV on page 405	MOD on page 411	SQRT on page 427	NEG on page 422	ABS on page 390

Structured Text

SQRT on page 427	ABS on page 390

If you want to:	Use this instruction:
evaluate an expression	CPT
add two values	ADD
subtract two values	SUB
multiply two values	MUL
divide two values	DIV
determine the remainder after one value is divided by another	MOD
calculate the square root of a value	SQRT
take the opposite sign of a value	NEG
take the absolute value of a value	ABS

You can mix data types, but loss of accuracy and rounding error might occur and the instruction takes more time to execute. Check the S:V bit to see whether the result was truncated.

The bold data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

A compute/math instruction executes once each time the instruction is scanned as long as the rung-condition-in is true. If you want the expression evaluated only once, use any one-shot instruction to trigger the instruction.

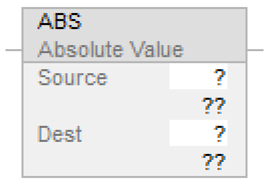
Absolute Value (ABS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the ABS instruction and operator take the absolute value of Source. The instruction stores the result in Dest while the operator simply returns the result. An overflow is indicated if the result is the maximum negative integer value, e.g. -128 for SINT, -32,768 for INT and -2,147,483,648 for DINT.

Available Languages

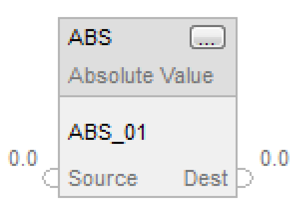
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use ABS as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of which to take the absolute value.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
ABS	FBD_MATH_ADVANCED	tag	ABS structure

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Value of which to take the absolute value.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operand (Left Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of which to take the absolute value.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Result of the function.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = absolute value of Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false.	Set EnableOut to EnableIn.
EnableIn is true	Dest = absolute value of Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

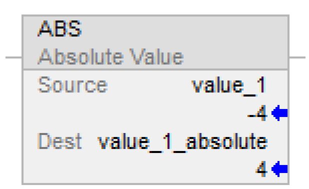
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = absolute value of Source
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

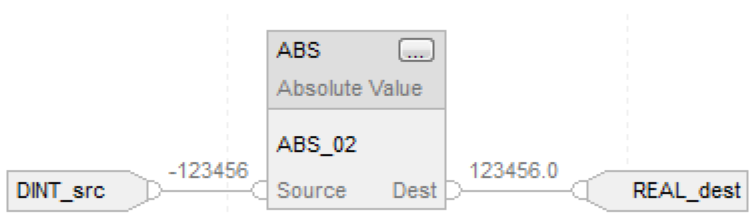
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

DINT_dest := ABS(DINT_src);

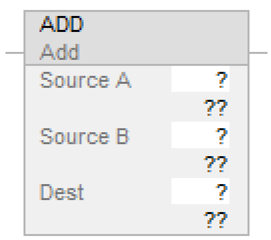
Add (ADD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the ADD instruction and the operator '+' adds Source A to Source B.

Available Languages

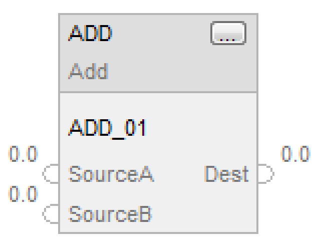
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator '+' in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
SourceA	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME* TIME* TIME32* LDT* DT*	immediate tag	Value to add to Source B

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
SourceB	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME* TIME* TIME32* LDT* DT*	immediate tag	Value to add to Source A
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME* TIME* TIME32* LDT* DT*	tag	Tag to store result of the instruction

NOTE:

*Keep these considerations in mind when using relative time (LTIME, TIME32, TIME) and absolute time (LDT, DT) data types in ADD instructions:

- If both Source A and Source B are relative time, the Dest must be relative time.
- If Source A is relative time and Source B is absolute time or vice versa, the Dest must be absolute time.
- In ADD instructions, Source A and Source B cannot both be absolute time.

See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
ADD	FBD_MATH	tag	ADD structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value to add to SourceB.
SourceB	REAL	Value to add to SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to add to SourceB.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to add to SourceA.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

See *FBD Functions* on page 440.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional

Controllers	Affects Math Status Flags
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

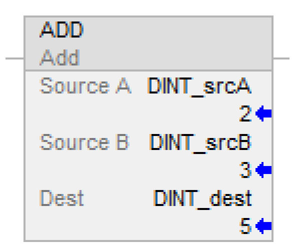
Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest = Source A + Source B
Postscan	N/A

Function Block Diagram

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Dest = SourceA + SourceB If overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

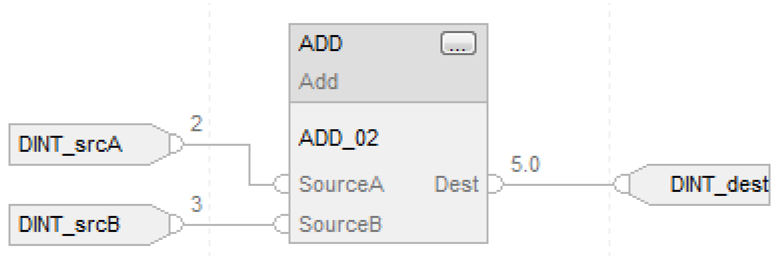
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

DINT_dest := DINT_srcA + DINT_srcB;

Compute (CPT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the CPT instruction evaluates the expression and places the result in the Dest.

The CPT instruction enables complex expressions in one instruction.

When evaluating the expression, all non-LREAL operands convert to LREAL before performing calculations if either of these conditions are true:

- Any operand in the expression is LREAL.
- The Dest is LREAL.

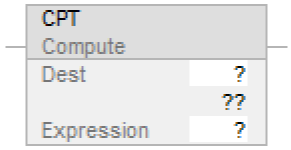
When evaluating the expression, all non-REAL operands will be converted to REAL before performing calculations if any operand or Dest in the expression is NOT LREAL, and any of these conditions are true:

- Any operand in the expression is REAL.
- The expression contains SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, DEG or RAD.
- The Dest is REAL.

There are rules for allowable operators in safety applications. See *Valid Operators*.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result.
Expression	SINT INT DINT	SINT INT DINT	immediate tag	An expression consisting of tags and/or immediate

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
	REAL	LINT USINT UINT UDINT ULINT REAL LREAL		values separated by operators.

Formatting expressions

For each operator used in an expression, one or two operands (tags or immediate values) must be provided. Use the following table to format operators and operands within an expression.

For operators that operate on:	Use this format:	Example
One operand	operator(operand)	ABS(tag)
Two operands	operand_a operator operand_b	tag_b + 5 tag_c AND tag_d (tag_e**2) MOD (tag_f / tag_g)

Determine the order of operation

The instruction performs the operations in the expressions in a prescribed order. Specify the order of operation by grouping terms within parentheses. This forces the instruction to perform an operation within the parentheses ahead of the other operations.

Operations of equal order are performed from left to right.

Order	Operation
1	( )
2	ABS, ACOS, ASIN, ATAN, COS, DEG, BCD_TO, LN, LOG, RAD, SIN, SQRT, TAN, TO_BCD, TRUNC
3	**
4	- (negate), NOT
5	*, /, MOD
6	- (subtract), +
7	AND
8	XOR
9	OR

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The instruction evaluates the expression and places the result in the Dest.
Postscan	N/A

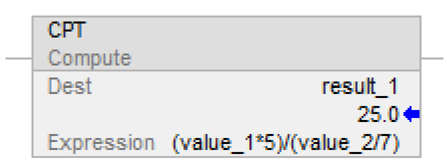
Examples

Ladder Diagram

Example 1

When enabled, the CPT instruction evaluates value_1 multiplied by 5 and divides that result by the result of value_2 divided by 7 and places the final result in result_1.

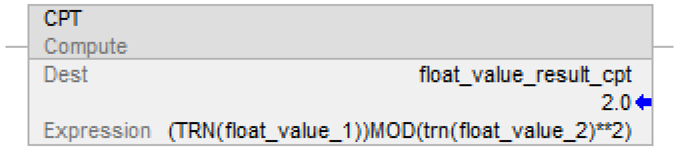
result_1	25.0		Float	REAL
+ value_1	10		Decimal	DINT
+ value_2	14		Decimal	DINT



Example 2

When enabled, the CPT instruction truncates float_value_1 and float_value_2 to the power of two and divides the truncated float_value_1 by that result, and then stores the remainder after the division in float_value_result_cpt.

Ladder Diagram



	float_value_result_cpt	2.0		Float	REAL
	float_value_1	10.5		Float	REAL
	float_value_2	2.5		Float	REAL

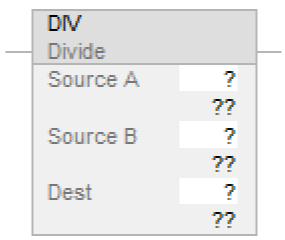
Divide (DIV)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the DIV instruction and the operator '/' divides Source A by Source B.

Available Languages

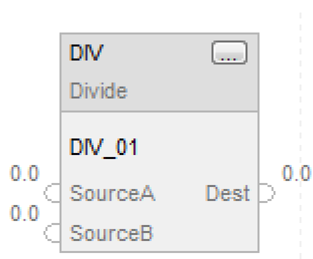
Ladder Diagram



Function Block Diagram

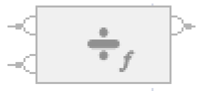
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Tip: Use the operator '/' in an expression to compute the same result. Refer to *Structured Text Syntax* for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Ladder Diagram

Operand	**Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers**	**Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers**	**Format**	**Description**
SourceA	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of the dividend
SourceB	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT	immediate tag	Value of the divisor

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		ULINT REAL LREAL		
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
DIV	FBD_MATH	tag	DIV structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source A	REAL	Value of the dividend.
Source B	REAL	Value of the divisor.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the dividend.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the divisor

Output Operands (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function

See *FBD Functions*.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
Source_B = 0	4	4

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest = Source A / Source B [1,2]
Postscan	N/A

Function Block Diagram

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Dest = SourceA / SourceB [1,2] If overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

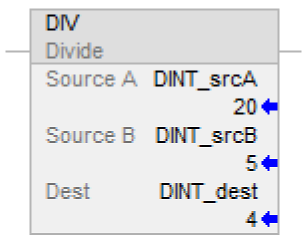
Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = SourceA / SourceB[1,2]
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

[1] If SourceB is zero the result will be SourceA for integer divides and 1.$ for floating point divides. This condition can also cause minor overflow faults on page 145.

[2] For integer destination and source operands the result is truncated.

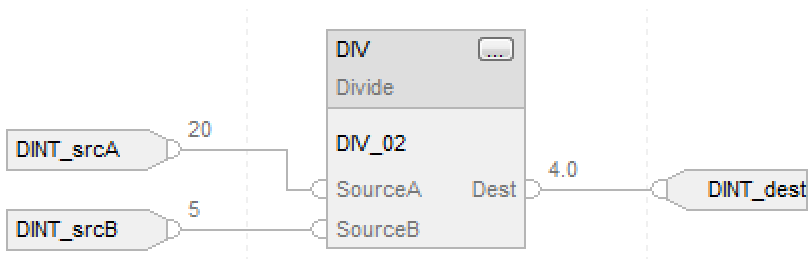
Examples

Ladder Diagram

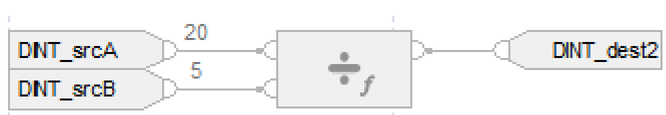


Function Block Diagram

FBD Block



FBD Function



Structured Text

DINT_dst := DINT_srcA / DINT_srcB;

Modulo (MOD)

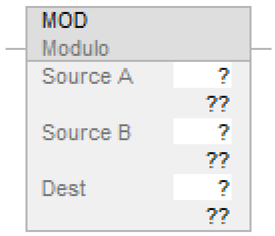
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the MOD instruction and the operator divides Source A by Source B and places the remainder in Dest. This is done using the algorithm:

Dest = Source A – (truncate (Source A / Source B) * Source B)

Available Languages

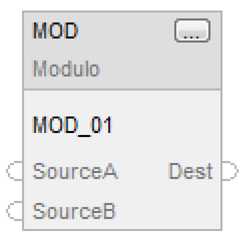
Ladder Diagram



Function Block Diagram

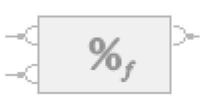
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Tip: Use MOD as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

These are the operands for Ladder Diagram.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of the dividend.
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of the divisor.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
MOD	FBD_MATH	tag	MOD structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value of the dividend.
SourceB	REAL	Value of the divisor.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the dividend.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the divisor

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

See FBD Functions on page 862.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional

Controllers	Affects Math Status Flags
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
Source B = 0	4	4

See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest is set (to the remainder) as described in the Description section.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Dest is set (to the remainder) as described in the Description section. If an overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

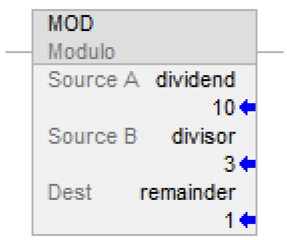
Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest is set (to the remainder) as described in the Description section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Tip: If Source B is 0, the result is 0 and a minor fault is generated.

Examples

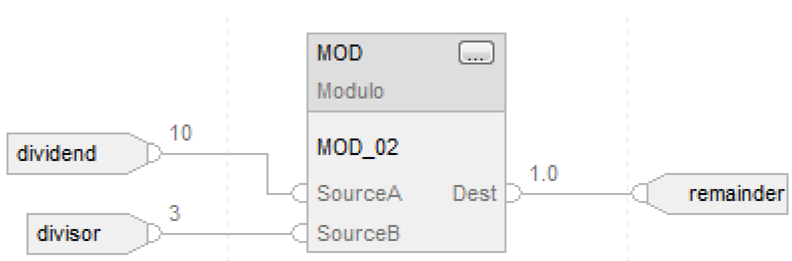
Ladder Diagram



Divide dividend by divisor and place the remainder in remainder. In this example, 3 goes into 10, three times, with a remainder of 1.

Function Block Diagram

FBD Block



FBD Function

Structured Text

remainder := dividend MOD divisor;

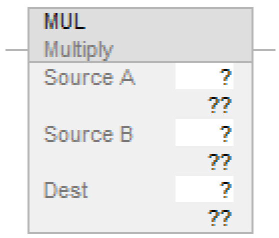
Multiply (MUL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the MUL instruction and the operator "*" multiplies Source A with Source B.

Available Languages

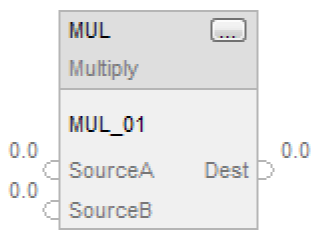
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator '*' in an expression to compute the same result. Refer to *Structured Text Syntax* for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of the multiplicand.
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value of the multiplier.
Dest	SINT INT	SINT INT	tag	Tag to store the result of the instruction.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
	DINT REAL	DINT LINT USINT UINT UDINT ULINT REAL LREAL		

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
MUL	FBD_MATH	tag	MUL structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value of the multiplicand.
SourceB	REAL	Value of the multiplier.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the multiplicand.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of the multiplier.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

See *FBD Functions*.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional

Controllers	Affects Math Status Flag
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest = Source A x Source B
Postscan	N/A

Function Block Diagram

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Dest = SourceA x SourceB If overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

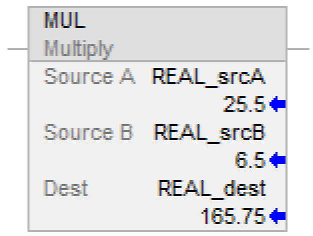
Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = Source A x Source B
Instruction first run	N/A

Condition/State	Action Taken
Instruction first scan	N/A
Postscan	N/A

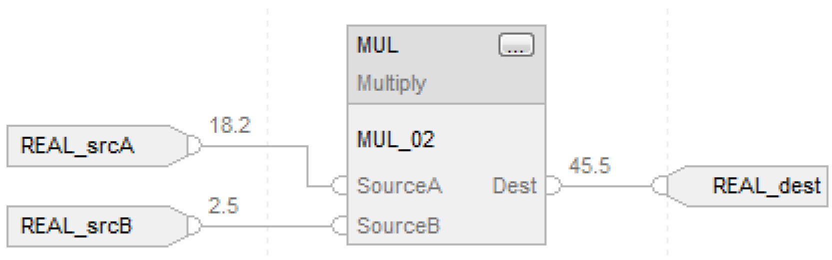
Examples

Ladder Diagram

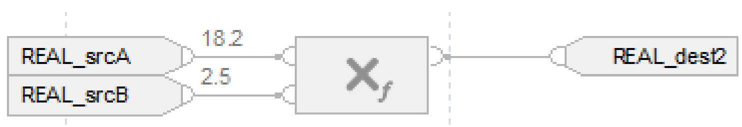


Function Block Diagram

FBD Block



FBD Function



Structured Text

REAL_dest := REAL_srcA * REAL_srcB;

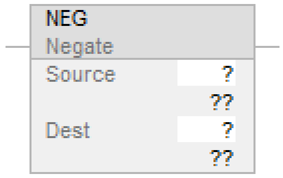
Negate (NEG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the NEG instruction and operator subtract the Source value from zero.

Available Languages

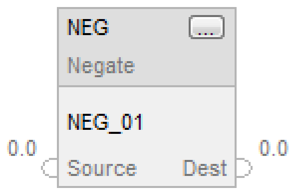
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use operator '-' in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to negate
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
NEG	FBD_MATH_ADVANCED	tag	NEG structure

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Value to negate.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operand (Left Pin)	Data Type	Description
Source	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to negate.

Output Operand (Right Pin)	Data Type	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

See FBD Functions on page 862.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = 0 - Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = 0 - Source. If overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

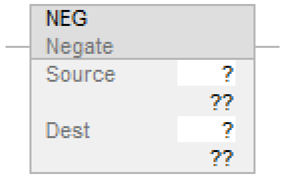
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = 0 - Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

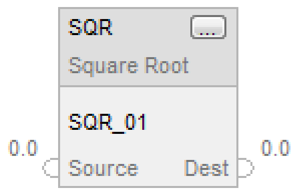
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

DINT_dest := -DINT_src;

Square Root (SQRT)

This table lists the controllers and applications that support this instruction.

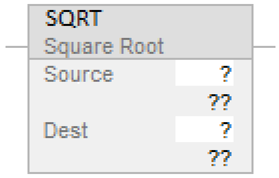
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The Square Root (SQRT) instruction and operator computes the square root of the Source and places the result in Dest.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from SQR to SQRT.

Available Languages

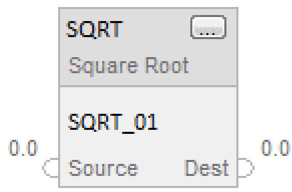
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use SQRT as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Computes the square root of this value.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
SQRT	FBD_MATH_ADVANCED	tag	SQRT structure

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Find the square root of this value.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operand (Left Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Computes the square root of this value.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

See FBD Functions on page 862.

Description

If the Dest is not an LREAL/REAL, the instruction handles the fractional portion of the result as follows:

If the Source is:	(For CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers) The fractional portion of the result:	Example			(For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers) The fractional portion of the result:	Example		
any elementary integer tag/value	Truncates	Source	DINT	3	Rounds	Source	DINT	3
		Dest	DINT	1		Dest	DINT	2
any floating point tag/value	Rounds	Source	REAL	3.0	Rounds	Source	REAL	3.0
		Dest	DINT	2		Dest	DINT	2

If the Source is negative, the instruction takes the absolute value of the Source before calculating the square root.

For the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers, if the Source is an integer data type and the Dest is an integer data type, the instruction truncates the result. For example, if the integer Source value is 3, the result is 1.732, and the Dest value becomes 1.

If the Source is a real data type and the Dest is an integer type, the instruction rounds the result. For example, if the real Source value is 3.0, the result is 1.732, and the Dest value becomes 2.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = square root of Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest. = square root of Source. If overflow occurs Clear EnableOut to false else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

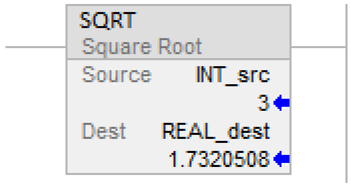
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = square root of Source
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

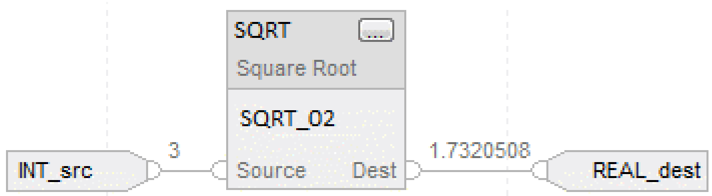
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := SQRT(INT_src);

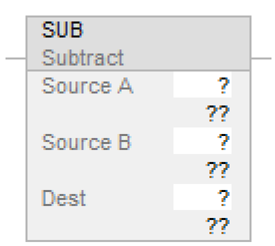
Subtract (SUB)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

When enabled, the SUB instruction and the operator '-' subtracts Source B from Source A.

Available Languages

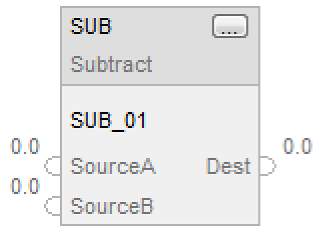
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator '-' in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME* TIME* TIME32* LDT* DT*	immediate tag	Value from which to subtract Source B.
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME* TIME* TIME32* LDT* DT*	immediate tag	Value to subtract from Source A.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT	tag	Tag to store result of the instruction.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		REAL LREAL LTIME* TIME* TIME32* LDT* DT*		

Tip:

*Keep these considerations in mind when using relative time (LTIME, TIME32, TIME) and absolute time (LDT, DT) data types in SUB instructions:

- If both Source A and Source B are relative time, the Dest must be relative time.
- If Source A is relative time and Source B is absolute time or vice versa, the Dest must be absolute time.
- In ADD instructions, Source A and Source B cannot both be absolute time.

See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
SUB	FBD_MATH	tag	SUB structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value from which to subtract SourceB.
SourceB	REAL	Value to subtract from SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value from which to subtract SourceB.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to subtract from SourceA.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT	Result of the function.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
	LINT ULINT REAL LREAL	

See FBD Functions on page 862.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest = Source A - Source B
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Dest = SourceA - SourceB If overflow occurs Clear EnableOut to false

Condition/State	Action Taken
	else Set EnableOut to true
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

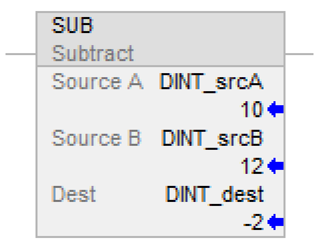
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = SourceA - SourceB
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

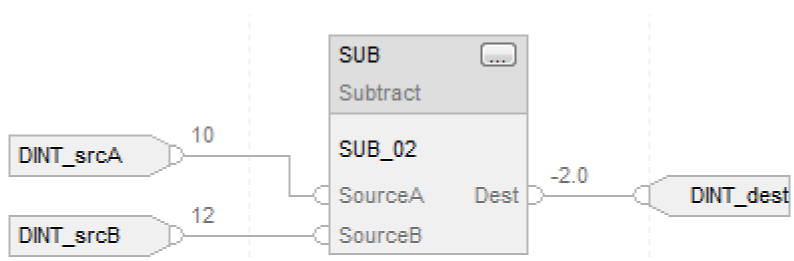
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

DINT_dest := DINT_srcA - DINT_srcB;

FBD Functions

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

FBD Functions are implemented in accordance with IEC 61131-3 Edition 3. Arithmetic and Numeric functions are provided in the Function Block Diagram language. Ladder Diagram and Structured Text languages include Arithmetic and Numeric as operators and functions.

FBD Functions have one or more inputs and one output. FBD Functions are implemented for efficiency, have smaller footprints and use less system resources to operate than FBD Function Blocks.

FBD Functions

- Require all inputs and outputs. All inputs must be of a supported data type.
- Do not have backing tags or predefined data types. Connected input values do not convert to predefined data types.
- Do not have EnableIn bits and are always executed.

Example: Add Function

Function Overloading

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Function overloading defines two or more functions with the same name but different signature, such as argument or return type. FBD Functions that support overloading take a range of input data types. The output data types depend on the input data types.

FBD Functions follow these rules:

- Input type promotion
    - Input type promotion
        - Data types rankings from highest to lowest priority:
          LREAL, REAL, ULINT, LINT, UDINT, DINT, UINT, INT, USINT, SINT
        - All inputs promote to the data type of the input with the highest rank before execution
        - If all inputs have a rnak value of DINT or lower, all inputs promote to DINT type before execution
    - Output type depends on the input type
      The function's output type is the promoted input type

For example, Add function,

- SINT + UINT inputs promote to DINT + DINT inputs. Outputs are DINT
- USINT + LINT inputs promote to LInt + LINT inputs. Outputs are LINT
- UNIT + LREAL inputs promote to LREAL + LREAL inputs. Outputs are LREAL

Move/Logical Instructions

The Move instructions modify and move bits.

Available Instructions

Ladder Diagram

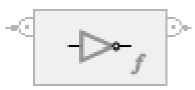
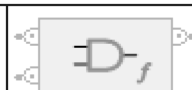
MOVE on page 482	MVM on page 476	AND on page 451	OR on page 467	XOR on page 457	NOT on page 462	SWPB on page 486	CLR on page 473	BTD on page 444

Function Block Diagram

FBD Block

MVMT on page 478	AND on page 451	OR on page 467	XOR on page 457	NOT on page 462	BTDT on page 447	BAND on page 489	BXOR on page 493
BOR on page 500	BNOT on page 497						

FBD Function

 BNOT on page 497	 BOR on page 500	 BAND on page 489	 BXOR on page 493
 AND on page 451	 NOT on page 462	 OR on page 467	 XOR on page 457

Structured Text

MVMT on page 478	SWPB on page 486	BTDT on page 447

If you want to:	Use this instruction:
Copy a value or move strings	MOVE
Copy a specific part of an integer	MVM
Copy a specific part of an integer in a function block	MVMT
Move bits within an integer or between integers	BTD
Move bits within an integer or between integers in a function block	BTDT
Clear a value	CLR
Rearrange the bytes of an INT, DINT, or REAL tag	SWPB

The logical instructions perform logical operations on bits.

If you want to:	Use this instruction:
Perform a bitwise AND operation	AND
Perform a bitwise OR operation	OR
Perform a bitwise, exclusive OR operation	XOR
Perform a bitwise NOT operation	NOT

You can mix data types, but loss of accuracy and rounding error might occur and the instruction takes more time to execute. Check the S:V bit to see whether the result was truncated.

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

A move/logic instruction executes once each time the instruction is scanned as long as the rung-condition-in is true. If you want the expression evaluated only once, use any one-shot instruction to trigger the move/logic instruction.

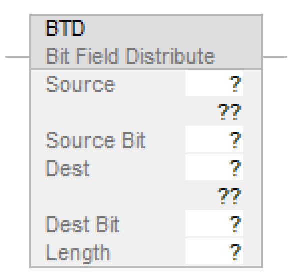
Bit Field Distribute (BTD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The BTD instruction copies the specified bits from the Source, shifts the bits to the appropriate position, and writes the bits into the Destination.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram

Operand	Type	Format	Description
Source	SINT INT DINT	immediate tag	Tag that contains the bits to move
Source bit	DINT	immediate (0-31)	Number of the bit (lowest bit number) from where to start the move Must be within the valid range for the Source data type
Destination	SINT INT DINT	tag	Tag where to move the bits
Destination bit	DINT	immediate (0-31)	The number of the bit to which the data should be moved must be within the valid range for the Destination data type.
Length	DINT	immediate (1-32)	Number of bits to move

Description

When enabled, the BTD instruction copies a group of bits from the Source to the Destination. The group of bits is identified by the Source bit (lowest bit number of the Source) and the Length (number of bits to copy). The Destination bit identifies the lowest bit number to start with in the Destination. The Source remains unchanged.

If the length of the bit field extends beyond the Destination, the instruction does not save the extra bits. Any extra bits do not wrap to the next word.

A SINT or INT tag is converted to a DINT value by zero-fill.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

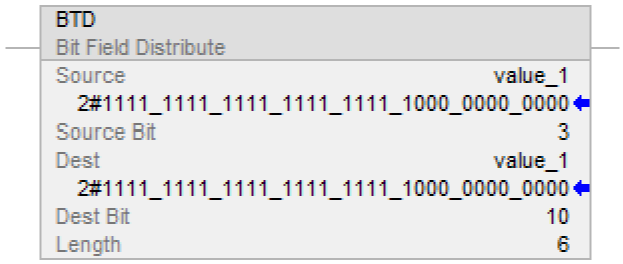
Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false.	N/A

Condition/State	Action Taken
Rung-condition-in is true.	The instruction copies and shifts the Source bits to the Destination.
Postscan	N/A

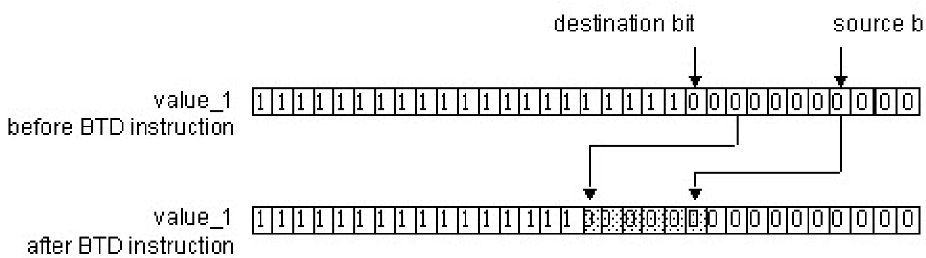
Examples

Example 1

Ladder Diagram

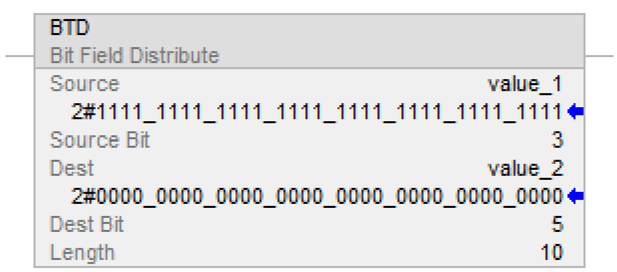


When enabled, the BTD instruction moves bits within value_1.

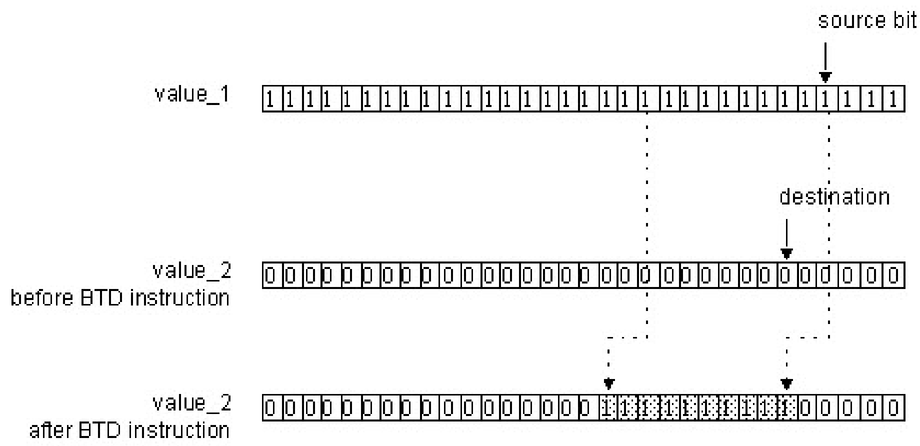


The shaded boxes show the bits that changed in value_1.

Example 2



When enabled, the BTD instruction moves 10 bits from value_1 to value_2.



The shaded boxes show the bits that changed in value_2.

Bit Field Distribute with Target (BTDT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

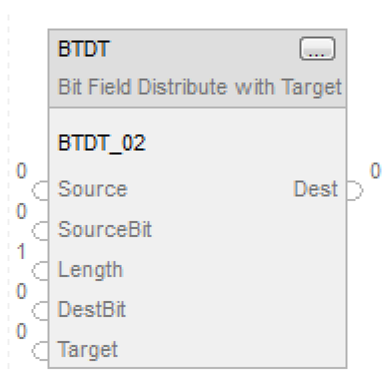
The BTDT instruction first copies the Target to the Destination. Then the instruction copies the specified bits from the Source, shifts the bits to the appropriate position, and writes the bits into the Destination. The Target and Source remain unchanged.

Available Languages

Ladder Diagram

This instruction is not available in a ladder diagram.

Function Block



Structured Text

BTDT(BTDT_tag);

Operands

Function Block

Operand	Type	Format	Description
BTDT tag	FBD_BIT_FIELD_DISTRIBUTE	structure	BTDT structure

Structured Text

Input Parameter	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
Source	DINT	Input value containing the bits to move to Destination. Valid = any integer
SourceBit	DINT	The bit position in Source (lowest bit number from where to start the move). Valid = 0-31
Length	DINT	Number of bits to move. Valid = 1-32
DestBit	DINT	The bit position in Dest (lowest bit number to start copying bits into). Valid = 0-31
Target	DINT	Input value to move to Dest prior to moving bits from the Source. Valid = any integer

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Dest	DINT	Result of the bit move operation.

See *Structured Text Syntax* for information on the syntax of expressions within structured text.

Description

When true, the BTDT instruction first copies the Target to the Destination, and copies a group of bits from the Source to the Destination. The group of bits is identified by the Source bit (lowest bit number of the group) and the Length (number of bits to copy). The Destination bit identifies the lowest bit number bit to start with in the Destination. The Source and Target remains unchanged.

If the length of the bit field extends beyond the Destination, the instruction does not save the extra bits. Any extra bits do not wrap to the next word.

Affects Math Status Flags

Controllers	Affected Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	No

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

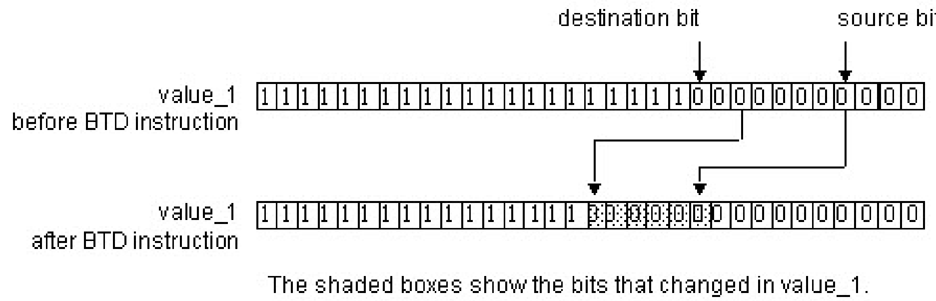
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal Execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Example

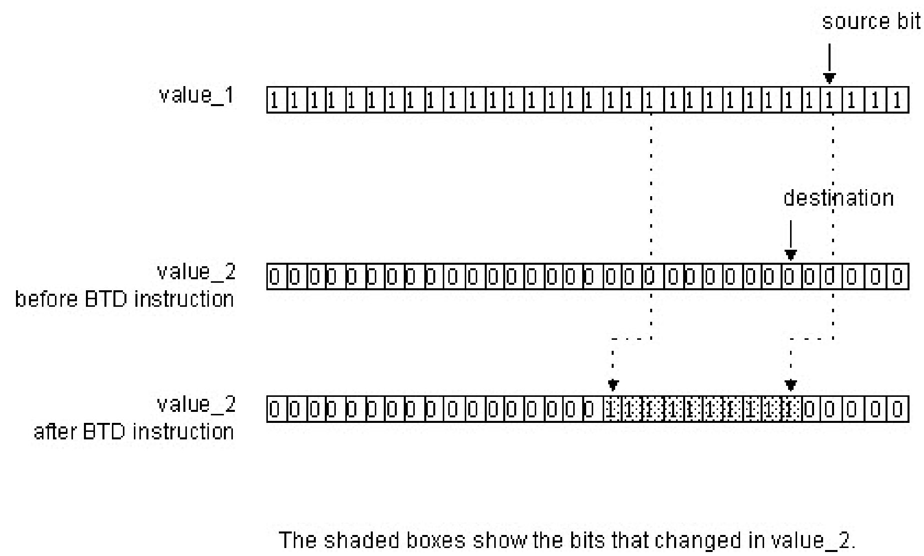
Step 1

The controller copies Target into Dest.

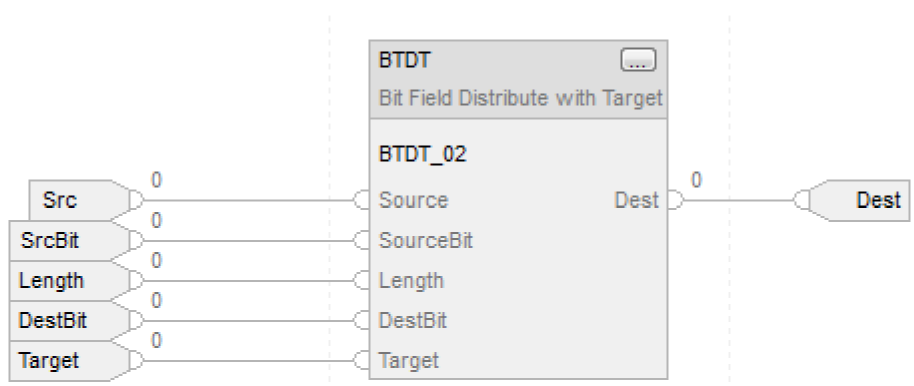


Step 2

The SourceBit and the Length specify which bits in Source to copy into Dest. Starting at DestBit, Source and Target remain unchanged.



Function Block



Structured Text

BTDT_01.Source := sourceSTX;

BTDT_01.SourceBit := source_bitSTX;

```
BTDT_01.Length := LengthSTX;

BTDT_01.DestBit := dest_bitSTX;

BTDT_01.Target := TargetSTX;

BTDT(BTDT_01);

distributed_value := BTDT_01.Dest;
```

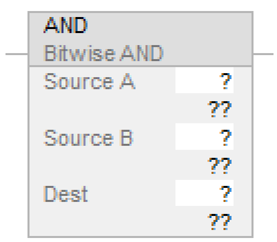
Bitwise And (AND)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The AND instruction performs a bitwise AND operation using the bits in Source A and Source B and places the result in Dest.

Available Languages

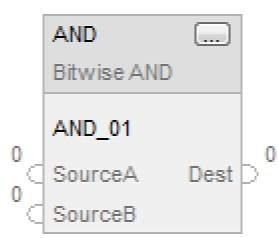
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

Function Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use the operator AND (or &) in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to AND with Source B. //Float includes REAL and LREAL data types. **Tip:** Float inputs are converted to integer which may cause an overflow.
Source B	SINT INT DINT REAL	SINT INT DINT LINT	immediate tag	Value to AND with Source A. **Tip:** Float inputs are converted to integer

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		USINT UINT UDINT ULINT REAL LREAL		which may cause an overflow.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction. **Tip:** If the destination type is Float, the resultant value will be converted to Float.

Tip: When integer promotion is required for the inputs, the smaller type is converted to the larger type using zero extension.

Function Block

Operand	Data Type	Format	Description
AND	FBD_LOGICAL	tag	AND structure

FBD_LOGICAL Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	DINT	Value to AND with SourceB.
SourceB	DINT	Value to AND with SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fail when it was enabled.
Dest	DINT	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to AND with Source B.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to AND with Source A.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT	Result of the function.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
	LINT ULINT	

See FBD Functions on page 862.

Description

When enabled, the instruction evaluates the bitwise AND operation: Dest = A AND B

If the bit in Source A is:	And the bit in Source B is:	The bit in the Dest is:
0	0	0
0	1	0
1	0	0
1	1	1

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest is set as described in the Description section.
Postscan	N/A

Function Block

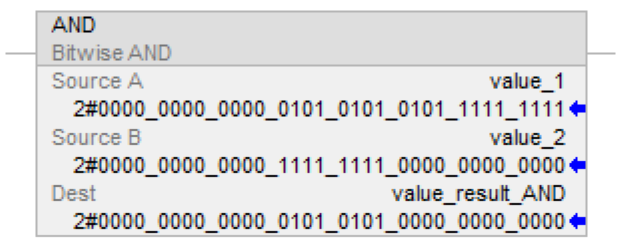
Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Set EnableOut to EnableIn Dest is set as described in the Description section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

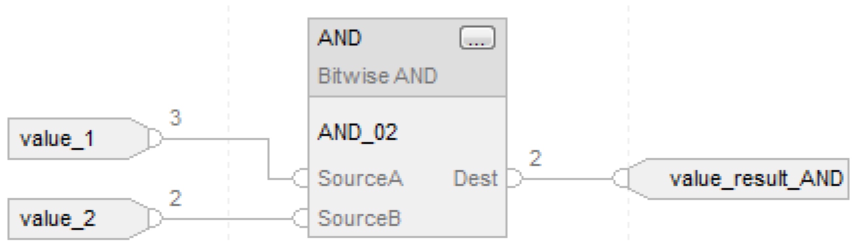
Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = SourceA AND SourceB
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Examples

Ladder Diagram



Function Block



FBD Function

Structured Text

value_result_and := value_1 AND value_2;

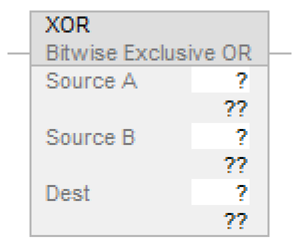
Bitwise Exclusive Or (XOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The XOR instruction performs a bitwise XOR operation using the bits in Source A and Source B and places the result in Dest.

Available Languages

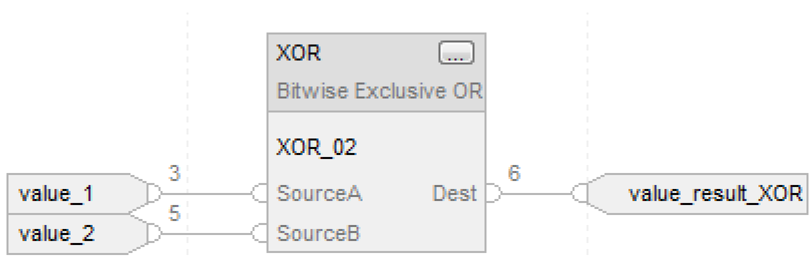
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

Function Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use XOR as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to XOR with Source B. //Float includes REAL and LREAL data types. **Tip:** Float inputs are converted to integer which may cause an overflow.
Source B	SINT INT DINT REAL	SINT INT DINT LINT	immediate tag	Value to XOR with Source A. **Tip:** Float inputs are converted to integer

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		USINT UINT UDINT ULINT REAL LREAL		which may cause an overflow.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction. **Tip:** If the destination type is Float, the resultant value will be converted to Float.

Tip: When integer promotion is required for the inputs, the smaller type is converted to the larger type using zero extension.

Function Block

Operand	Data Type	Format	Description
XOR	FBD_LOGICAL	tag	XOR structure

FBD_LOGICAL Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	DINT	Value to XOR with SourceB.
SourceB	DINT	Value to XOR with SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	DINT	Result of the instruction.

FBD Function

Input Operands (Left Pins)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to OR with Source B.
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to OR with Source A.

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT LINT ULINT	Result of the function.

Description

When enabled, the instruction evaluates the bitwise XOR operation:

Dest = Source A XOR Source B

If the bit in Source A is:	And the bit in Source B is:	The bit in Dest is:
0	0	0
0	1	1
1	0	1
1	1	0

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest is set as described in the Description section.
Postscan	N/A

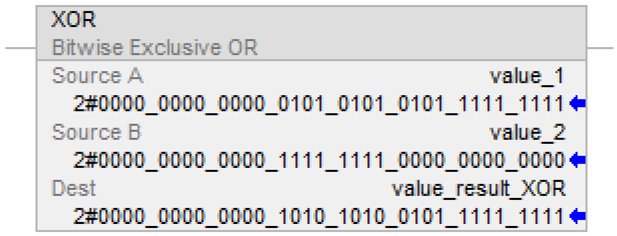
Function Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Set EnableOut to EnableIn Dest is set as described in the Description section.
Instruction first run	N/A
Instruction first scan	N/A

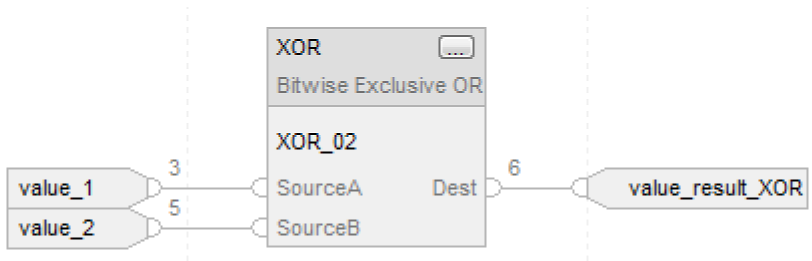
Condition/State	Action Taken
Postscan	N/A

Examples

Ladder Diagram



Function Block



FBD Function

Structured Text

value_result_XOR := value_1 XOR value_2;

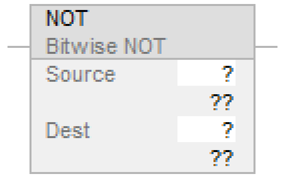
Bitwise Not (NOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The NOT instruction performs a bitwise inversion of the Source and places the result in Dest.

Available Languages

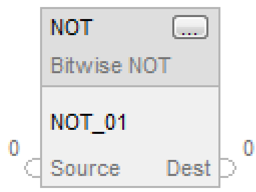
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

Function Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use NOT as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions.*

Ladder Diagram

Operand	CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers Data Type	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to NOT //FLOAT includes REAL and LREAL data types. **Tip:** Floating point inputs are converted to integer which may cause an overflow.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction. **Tip:** If the destination type is FLOAT, the resultant value will be converted to floating point.

Tip: When integer promotion is required for the inputs, the smaller type is converted to the larger type using zero extension.

Function Block

Operand	Data Type	Format	Description
NOT	FBD_CONVERT	tag	NOT structure

FBD_CONVERT Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	DINT	Value to NOT.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	DINT	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to NOT.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	DINT UDINT LINT ULINT	Result of the function.

See FBD Functions on page 862.

Description

When enabled, the instruction evaluates the bitwise NOT operation:

Dest = NOT Source

If the bit in the Source is:	The bit in the Dest is:
0	1
1	0

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest is set as described in the Description section.
Postscan	N/A

Function Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Set EnableOut to EnableIn Dest is set as described in the Description section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

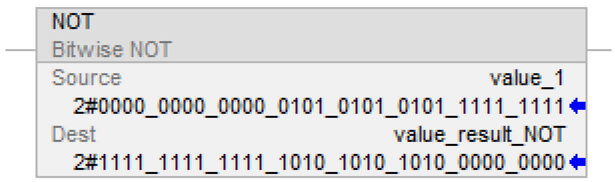
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

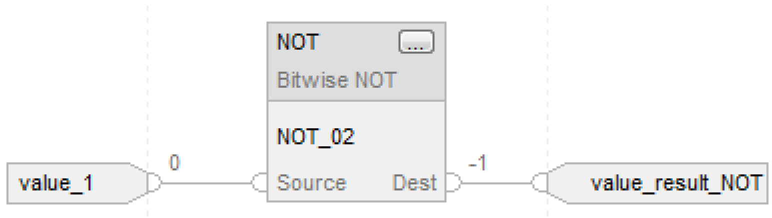
Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = NOT Source
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Examples

Ladder Diagram



Function Block



FBD Function

Structured Text

value_result_NOT := NOT value_1;

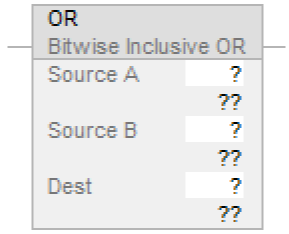
Bitwise Inclusive Or (OR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The OR instruction performs a bitwise OR operation using the bits in Source A and Source B and places the result in Dest.

Available Languages

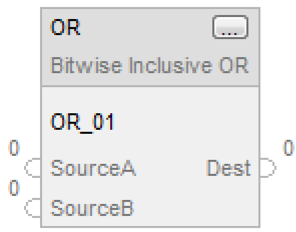
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use OR as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversion on page 851s.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to OR with Source B. //Float includes REAL and LREAL data types. **Tip:** Float inputs are converted to integer which may cause an overflow.
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Value to OR with Source A. **Tip:** Float inputs are converted to integer which may cause an overflow.
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction. **Tip:** If the destination type is Float, the resultant value will be converted to Float.

Tip: When integer promotion is required for the inputs, the smaller type is converted to the larger type using zero extension.

Function Block

Operand	Type	Format	Description
OR	FBD_LOGICAL	tag	OR structure

FBD_LOGICAL Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	DINT	Value to OR with SourceB.
SourceB	DINT	Value to OR with SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed successfully when it was enabled.
Dest	DINT	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceA (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to OR with Source B.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
SourceB (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to OR with Source A.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	DINT UDINT LINT ULINT	Result of the function.

See *FBD Functions*.

Description

When enabled, the instruction evaluates the bitwise OR operation:

Dest = Source A OR Source B

If the bit in Source A is:	And the bit in Source B is:	The bit in Dest is:
0	0	0
0	1	1
1	0	1
1	1	1

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in Dest is set as described in the Description section.
Postscan	N/A

Function Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn
EnableIn is true	Set EnableOut to EnableIn Dest is set as described in the Description section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

FBD Function

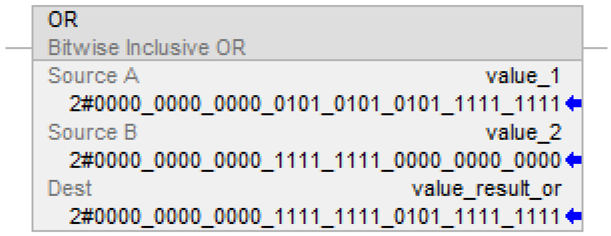
Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = SourceA OR SourceB
Instruction first run	N/A

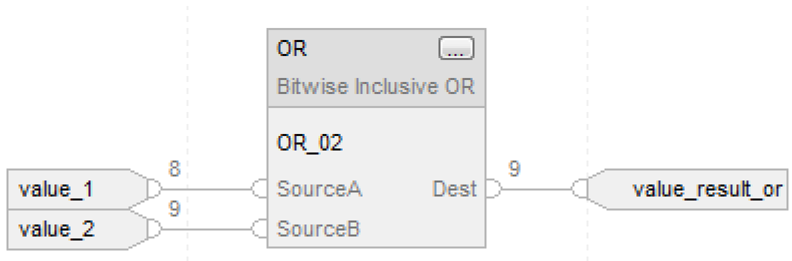
Condition/State	Action Taken
Instruction first scan	N/A
Postscan	N/A

Examples

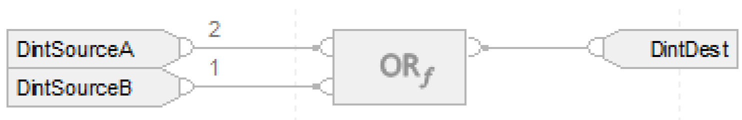
Ladder Diagram



Function Block



FBD Function



Structured Text

value_result_or := value_1 OR value_2;

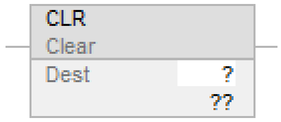
Clear (CLR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The CLR instruction clears all the bits of the Dest.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

The CLR instruction supports elementary data types. See *Elementary Data Types*.

Ladder Diagram

Operand	**Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers**	**Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers**	**Format**	**Description**
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL LTIME TIME TIME32 LDT	tag	Tag to clear.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		DT		

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

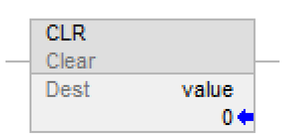
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Clear Dest to 0.
Postscan	N/A

Example

Ladder Diagram



Masked Move (MVM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The MVM instruction copies the Source to a Destination and allows portions of the data to be masked.

The MVM instruction uses a Mask to pass or block Source data bits. A "1" in the mask means the data bit is passed; a "0" in the mask means the data bit is blocked.

If integer data types are mixed, the instruction fills the upper bits of the smaller integer data types with 0s so that they are the same size as the largest data type.

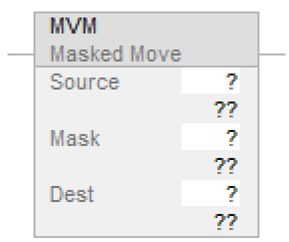
Entering an immediate mask value

When mask is entered, the programming software defaults to decimal values. To enter a mask using another format, precede the value with the correct prefix.

Prefix	Description
16#	Hexadecimal (e.g., 16#0F0F)
8#	Octal (e.g., 8#16)
2#	Binary (e.g., 2#00110011)

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixed data types within an instruction. See *Data Conversions*.

Ladder Diagram

Operand	Data Type	Format	Description
Source	SINT INT DINT	immediate tag	Value to move
Mask	SINT INT DINT	immediate tag	Which bits to block or pass
Dest	SINT INT DINT	tag	Tag to store the result

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

Controllers	A minor fault will occur if:	Fault Type	Fault Code
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	The feature is enabled and overflow is detected	4	4
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	N/A	N/A	N/A

See Index Through Arrays on page 863 for array-indexing faults.

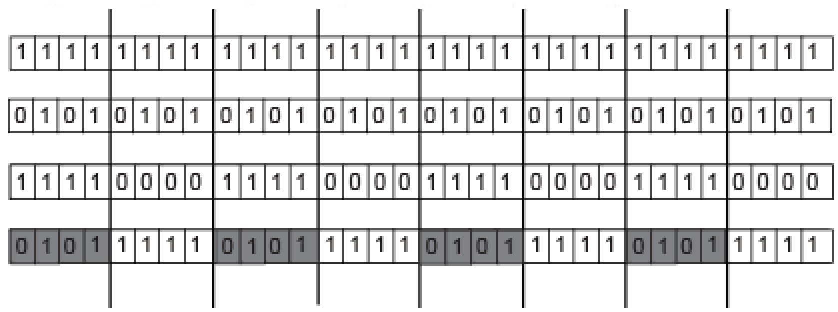
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction passes the Source through the Mask and copies the result into the Destination. Unmasked bits in the Destination remain unchanged.
Postscan	N/A

Example

Ladder Diagram



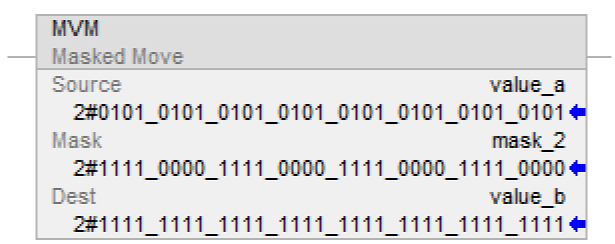
The shaded boxes show the bits that changed in value_b

Row 1: value_b before MVM

Row 2: value_a

Row 3: mask_2

Row 4: value_b after MVM



Copy data from value_a to value_b, while allowing data to be masked (a 0 masks the data in value_a).

Masked Move with Target (MVMT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The MVMT instruction copies the Source to a Destination and allows portions of the data to be masked.

Available Languages

Ladder Diagram

This instruction is not available in Ladder Diagram.

Function Block

Structured Text

MVMT(MVMT_tag);

Operands

Structured Text

Variable	Type	Format	Description
MVMT tag	FBD_MASKED_MOVE	Structure	MVMT structure

See Structured Text Syntax on page 879 for information on the syntax of expressions within structured text.

Function Block

Operand	Type	Format	Description
MVMT tag	FBD_MASKED_MOVE	Structure	MVMT structure

FBD_MASKED_MOVE Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	If cleared, the instruction does not execute and outputs are not updated. If set, the instruction executes. Default is set.
Source	DINT	Input value to move to Destination based on value of Mask. Valid = any integer
Mask	DINT	Mask of bits to move from Source to Dest. All bits set to one cause the corresponding bits to move from Source

Input Parameter	Data Type	Description
		to Dest. All bits that are set to zero cause the corresponding bits not to move from Source to Dest. Valid = any integer
Target	DINT	Input value to move to Dest prior to moving Source bits through the Mask. Valid = any integer

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Dest	DINT	Result of the masked move operation.

Description

When enabled, the MVMT instruction uses a Mask to pass or block Source data bits. A "1" in the mask means the data bit is passed. A "0" in the mask means the data bit is blocked.

If you mix integer data types, the instruction fills the upper bits of the smaller integer data types with 0s so that they are the same size as the largest data type.

Entering an immediate mask value using an Input Reference

When you enter a mask, the programming software defaults to decimal values. If you want to enter a mask using another format, precede the value with the correct prefix.

Prefix	Description
16#	hexadecimal (e.g., 16#0F0F)
8#	octal (e.g., 8#16)
2#	binary (e.g., 2#00110011)

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes for the output

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Function Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

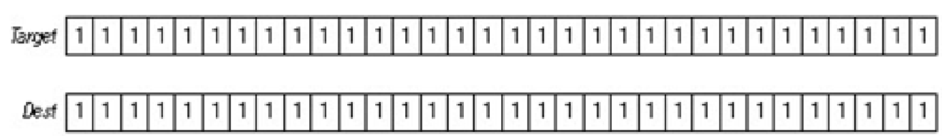
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Function Block table.
Normal execution	See Tag.EnableIn is true in the Function Block table.
Postscan	See Postscan in the Function Block table.

Examples

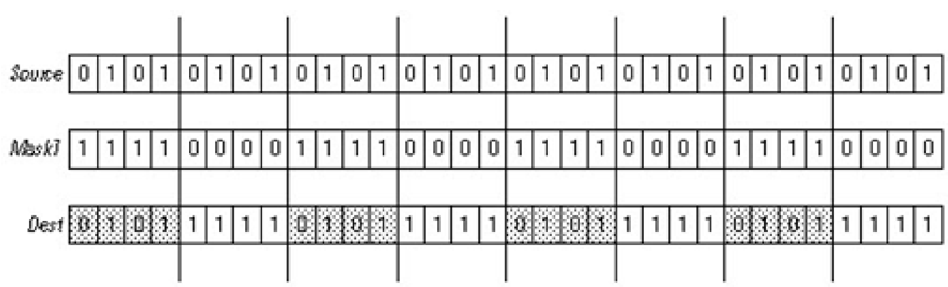
Step 1

The controller copies Target into Dest.



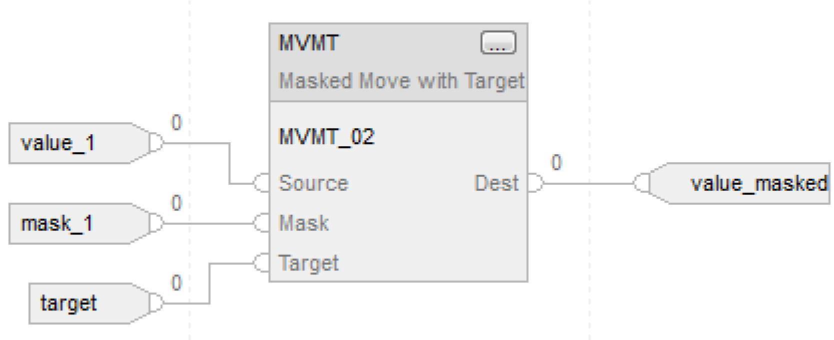
Step 2

The instruction masks Source and compares it to Dest. Any required changes are made in Dest, which becomes and input parameter to value_masked. Source and Target remain unchanged. A 0 in the mask restrains the instruction from comparing that bit.



The shaded boxes show the bits that changed.

Function Block



Structured Text

MVMT_01.Source := value_1;

MVMT_01.Mask := mask_1;

MVMT_01.Target := target;

MVMT(MVMT_01);

value_masked := MVMT_01.Dest;

Move (MOVE)

This table lists the controllers and applications that support this instruction.

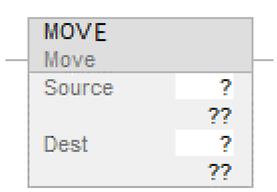
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The MOVE instruction moves a copy of the Source to the Dest. The Source remains unchanged.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from MOV to MOVE.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Tip: Use an assignment ":=" with an expression to achieve the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions on page 851.

Ladder Diagram

Numeric

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME	immediate tag	Value to move

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
		DT LDT		
Dest	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL TIME TIME32 LTIME DT LDT	tag	Tag to store the result

Tip: See Time and date data types on page 858 for a complete description of Relative Time (LTIME, TIME, and TIME32) and Absolute Time (LDT and DT) data types.

Tip: Keep these restrictions in mind when using Relative Time (LTIME, TIME, TIME32) and Absolute Time (LDT, DT) data types:

- A relative time type can move only to or from another relative time type.
- And absolute time type can move only to or from another absolute time type. Additionally, you can program an absolute time type with a LINT to accommodate some legacy timestamp practices.

String (for CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only)

Operand	Data Type	Format	Description
Source	String type	immediate tag	String to move
Dest	String type	tag	Tag to store the result

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math Status Flags on page 849.

Major/Minor Faults

A minor fault will occur if:	Fault type	Fault code
Overflow detection feature is enabled and the Source value is outside the range of Dest type.	4	4

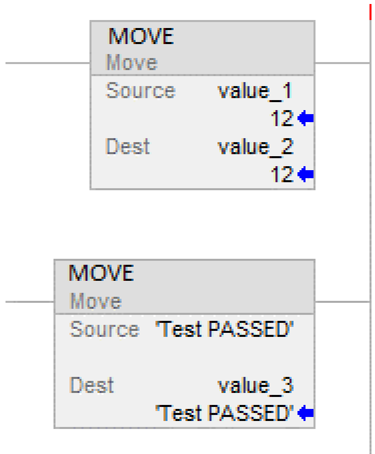
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The instruction copies the Source into the Dest. **String operands:** If Source.LEN > SIZE(Dest.DATA) The string is truncated to what will fit S:V is set.
Postscan	N/A

Examples

Ladder Diagram



Structured Text

value_2 := value_1;

value_3 := 'Test PASSED';

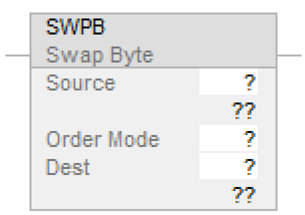
Swap Byte (SWPB)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SWPB instruction rearranges the order of the bytes of the Source. It places the result in the Destination.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

SWPB(Source, Order Mode, Dest);

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversion*.

Ladder Diagram and Structured Text

Operand	Data Type	Format	Description
Source	INT DINT	tag	Tag that contains the bytes to rearrange.
Order Mode		list item	This operand specifies how to reorder. Refer Order Mode table.
Dest	INT DINT	tag	Tag to store the bytes in a new order. Refer Dest table.

If selecting the HIGH/LOW order mode, enter it as HIGHLOW (without the slash). See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Order Mode

If the Source is an	And you want to change the bytes to this pattern(each letter represents a different byte)	Then select
INT	AB => BA	Any option
DINT	ABCD => DCBA	REVERSE
	ABCD =>CDAB	WORD
	ABCD => BADC	HIGH/LOW

Dest

If the Source is an	Then the Destination must be an
INT	INT, DINT If the destination is a DINT, the result is sign extended after bytes swap.
DINT	DINT

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction rearranges the specified bytes.
Postscan	N/A

Structured Text

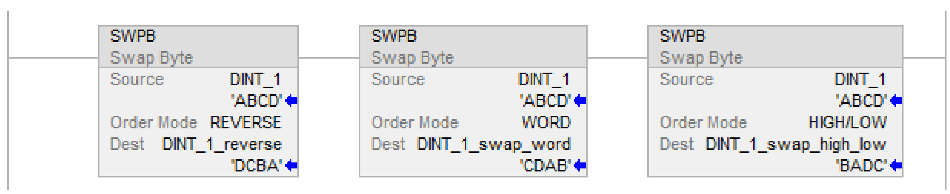
Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal Execution	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Examples

Example 1 - Swap the bytes of a DINT tag

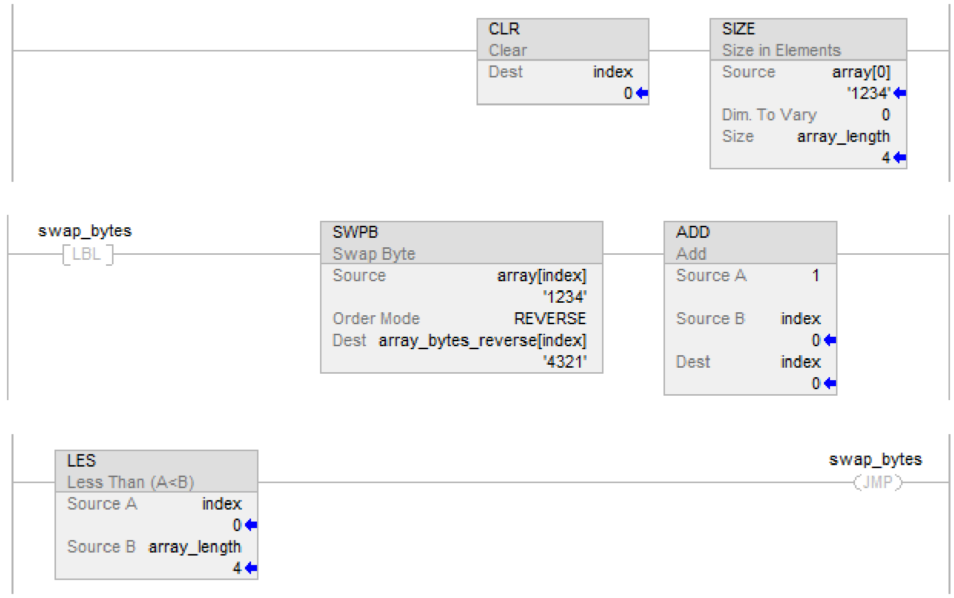
The three SWPB instructions reorder the bytes of DINT_1 according to a different order mode. The display style is ASCII, and each character represents one byte. Every instruction places the bytes, in the new order, in a different Destination.

Ladder Diagram



Example 2 - Swap the bytes in all elements of an array

Ladder Diagram



Example 3: SWPB on Structured Text

Structured Text

index := 0;

SIZE (array[0],0,array_length);

REPEAT

SWPB(array[index],REVERSE,array_bytes_reverse[index]);

index := index + 1;

UNTIL(index >= array_length)END_REPEAT;

Boolean AND (BAND)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The BAND instruction logically ANDs up to eight Boolean inputs. To perform a bitwise AND, refer to *Bitwise And (AND)*.

Available Languages

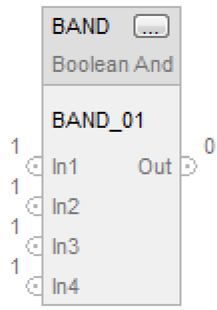
Ladder Diagram

This instruction is not available in ladder diagram.

Function Block Diagram

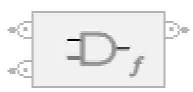
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Operands

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
BAND tag	FBD_BOOLEAN_AND	structure	BAND structure

FBD_BOOLEAN_AND Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In1	BOOL	First Boolean input. Set to 1 on first download.
In2	BOOL	Second Boolean input. Set to 1 on first download.

Input Members	Data Type	Description
In3	BOOL	Third Boolean input. Set to 1 on first download.
In4	BOOL	Forth Boolean input. Set to 1 on first download.
In5	BOOL	Fifth Boolean input. Set to 1 on first download.
In6	BOOL	Sixth Boolean input. Set to 1 on first download.
In7	BOOL	Seventh Boolean input. Set to 1 on first download.
In8	BOOL	Eighth Boolean input. Set to 1 on first download.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description
In1	BOOL	First Boolean input
In2	BOOL	Second Boolean input

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Out	BOOL	The output of the instruction.

See FBD Functions.

Operation

FBD Block

The BAND instruction ANDs up to eight Boolean inputs. If an input is not used, it defaults to set (1).

Out = In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8

IMPORTANT: When removing an input wire from the BAND instruction during an edit, make sure the input is set (1).

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

The FBD Function ANDs two Boolean inputs.

Out = In1 AND In2

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.

Execution

Function Block Diagram

FBD Block

Condition/State	**Action Taken**
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes as described in the Operation section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Out = In1 AND In2
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

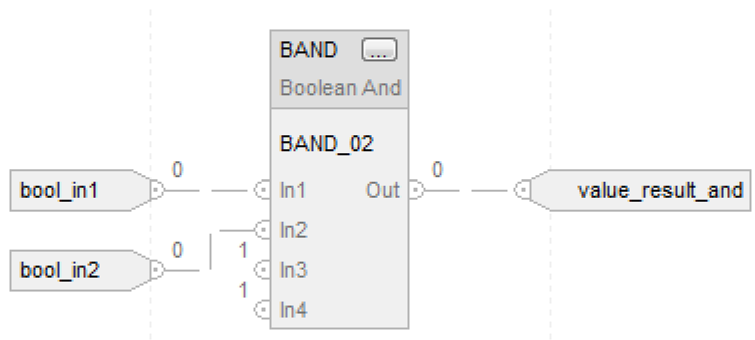
Example

Function Block Diagram

FBD Block

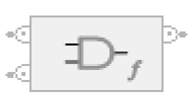
In this example, bool_in1 is copied into BAND_02.In1, bool_in2 is copied into BAND_02.In2, the result of performing AND of all BAND_02 inputs is placed into BAND_02.Out, and BAND_02.Out is then copied into value_result_and.

If bool_in1 is:	If bool_in2 is:	Then value_result_and is:
0	0	0
0	1	0
1	0	0
1	1	1



FBD Function

This example illustrates performing an AND on bool_in1 and bool_in2 and places the result in value_result_and.



Boolean Exclusive OR (BXOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The BXOR instruction performs an exclusive OR on two Boolean inputs.

Available Languages

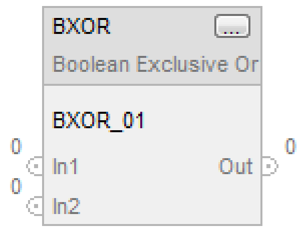
Ladder Diagram

This instruction is not available in ladder diagram.

Function Block Diagram

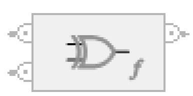
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Operands

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
BXOR tag	FBD_BOOLEAN_XOR	Structure	BXOR structure

FBD_BOOLEAN_XOR Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated.

Input Members	Data Type	Description
		Default is set.
In1	BOOL	First Boolean input. Default is cleared.
In2	BOOL	Second Boolean input. Default is cleared.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
In1	BOOL	First Boolean input.
In2	BOOL	Second Boolean input.

Output Operands (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Out	BOOL	The output of the instruction.

SeeFBD Functions on page 862FBD Functions.

Operation

The BXOR instruction performs an exclusive OR on two Boolean inputs.

Out = In1 XOR In2

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.

Execution

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes as described in the Operation section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Out = In1 XOR In2
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Example

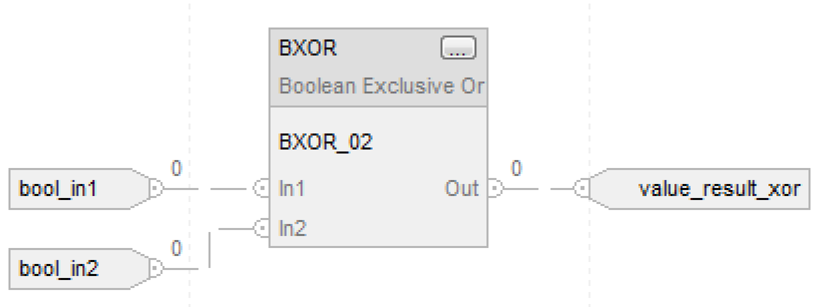
Function Block Diagram

In this example, bool_in1 is copied into BXOR_02.In1, bool_in2 is copied into BXOR_02.In2, the result of performing an exclusive OR on BXOR_02.In1 and BXOR_02.In2 is placed into BXOR_02.Out, and BXOR_02.Out is then copied into value_result_xor.

If bool_in1 is:	If bool_in2 is:	Then value_result_xor is:
0	0	0
0	1	1
1	0	1
1	1	0

FBD Block

This example illustrates performing an exclusive OR on bool_in1 and bool_in2 and places the result in value_result_xor.



FBD Function

Boolean NOT (BNOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The BNOT instruction complements a Boolean input. To perform a bitwise NOT, refer to *Bitwise Not (NOT).*

Available Languages

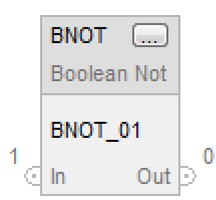
Ladder Diagram

This instruction is not available in ladder diagram.

Function Block Diagram

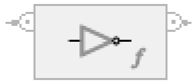
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Operands

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
BNOT tag	FBD_BOOLEAN_NOT	structure	BNOT structure

FBD_BOOLEAN_NOT Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
In	BOOL	Input to the instruction. Set to 1 on first download

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
In	BOOL	Input to the instruction.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Out	BOOL	The output of the instruction.

See FBD Functions on page 862.

Operation

The BNOT instruction complements a Boolean input.

Out = NOT In

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.

Execution

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes as described in the Operation section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

FBD Functions

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	The instruction executes as described in the Operation section.

Condition/State	Action Taken
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

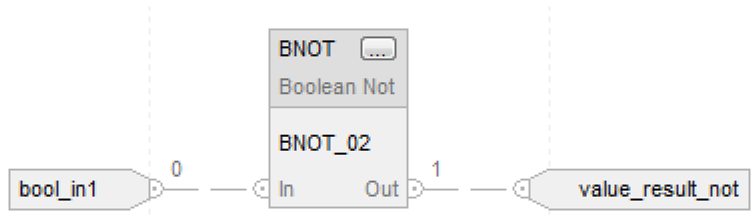
Example

Function Block Diagram

FBD Block

In this example, bool_in1 is copied into BNOT_02.In, the result of the complement of BNOT_02.In is placed into BNOT_02.Out and BNOT_02.Out is copied into value_result_not.

If bool_in1 is:	Then value_result_not is:
0	1
1	0



FBD Function

In this example, the result of the complement of bool_in1 is placed in value_result_not.

Boolean OR (BOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The BOR instruction logically ORs up to eight Boolean inputs. To perform a bitwise OR, refer to *Bitwise Or (OR)*.

Available Languages

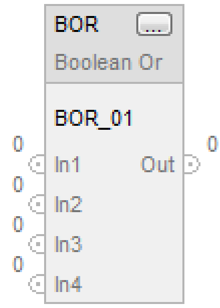
Ladder Diagram

This instruction is not available in ladder diagram.

Function Block Diagram

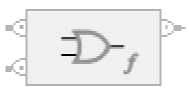
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function supports only two inputs and is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Operands

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
BOR tag	FBD_BOOLEAN_OR	structure	BOR structure

FBD_BOOLEAN_OR Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Set to 0 on first download.
In1	BOOL	First Boolean input. Set to 0 on first download.
In2	BOOL	Second Boolean input. Set to 0 on first download.
In3	BOOL	Third Boolean input.

Input Members	Data Type	Description
		Set to 0 on first download.
In4	BOOL	Forth Boolean input. Set to 0 on first download.
In5	BOOL	Fifth Boolean input. Set to 0 on first download.
In6	BOOL	Sixth Boolean input. Set to 0 on first download.
In7	BOOL	Seventh Boolean input. Set to 0 on first download.
In8	BOOL	Eighth Boolean input. Set to 0 on first download.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Out	BOOL	The output of the instruction.

FBD Function

Tip: FBD Function supports only two inputs and is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
In1	BOOL	First Boolean input.
In2	BOOL	Second Boolean input.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Out	BOOL	The output of the instruction.

See FBD Functions on page 862.

Operation

FBD Block

The BOR instruction ORs up to eight Boolean inputs. If an input is not used, it defaults to cleared (0).

Out = In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8

IMPORTANT: When removing an input wire from the BOR instruction during an edit, make sure the input is cleared (0).

FBD Function

Tip: FBD Function supports only two inputs and is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

The FBD Function ORs two Boolean inputs.

Out = In1 OR In2

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.

Execution

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is false	EnableIn and EnableOut bits are cleared to false.
Tag.EnableIn is true	EnableIn and EnableOut bits are set to true. The instruction executes as described in the Operation section.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	EnableIn and EnableOut bits are cleared to false.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Out = In1 OR In2
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

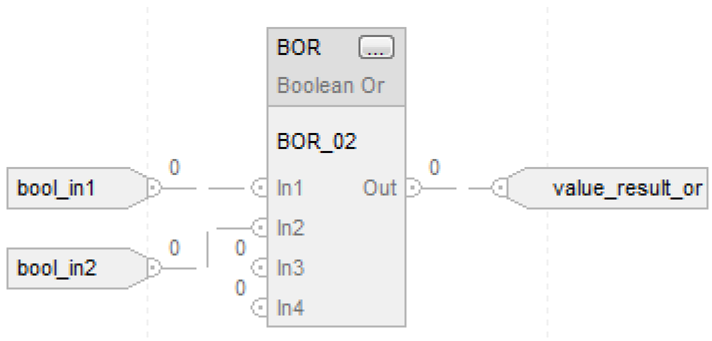
Example

Function Block Diagram

FBD Block

In this example, bool_in1 is copied into BOR_02.In1, bool_in2 is copied into BOR_02.In2, the result of performing OR of all BOR_02 inputs is placed into BOR_02.Out, and BOR_02.Out is then copied into value_result_or.

If bool_in1 is:	If bool_in2 is:	Then value_result_or is:
0	0	0
0	1	1
1	0	1
1	1	1



FBD Function

Array File-Misc Instructions

The file/miscellaneous instructions operate on arrays of data.

Available Instructions

Ladder Diagram

AVE on page 529	COP on page 506	CPS on page 506	FAL on page 514	FLL on page 533	FSC on page 536	SIZE on page 557

NOTE: STD on page 553

Function Block

Not available

Structured Text

SIZE on page 557	COP on page 506	CPS on page 506

If you want to:	Use this instruction:
Perform arithmetic, logic, shift, and function operations on values in arrays	FAL
Search for and compare values in arrays	FSC
Copy the contents of one array into another array	COP
Copy the value(s) in the Source to the Destination	CPS
Fill an array with specific data	FLL
Calculate the average of an array of values	AVE
Sort one dimension of array data into ascending order	SRT
Calculate the standard deviation of an array of values	STD
Find the size of a dimension of an array	SIZE

You can mix data types, but loss of accuracy and rounding error might occur and the instruction takes more time to execute. Check the S:V bit to see whether the result was truncated.

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

Selecting Mode of Operation

For FAL and FSC instructions, the mode tells the controller how to distribute the array operation.

If you want to:	Select this mode:

operate on all of the specified elements in an array before continuing on to the next instruction	All Mode
distribute array operation over a number of scans enter the number of elements to operate on per scan (1-2147483647)	Numerical Mode
manipulate one element of the array each time the rung-condition-in goes from false to true	Incremental Mode

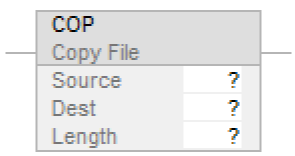
Copy (COP) - Synchronous Copy (CPS)

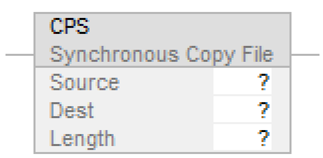
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, CompactLogix 5380, CompactLogix 5480, and ControlLogix 5580 controllers. Controller differences are noted where applicable.

The COP and CPS instructions copy the value(s) in the Source to the values in the Dest. The Source remains unchanged.

Available Languages

Ladder Diagram





Function Block

This instruction is not available in function block.

Structured Text

COP(Source,Dest,Length);

CPS(Source,Dest,Length);

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers Data Type	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Format	Description
Source	SINT INT DINT LINT REAL String type structure	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type structure	tag	Initial element to copy. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type. The Reference (REF) Instruction associates a reference with an axis or coordinate system concrete tag.
Dest	SINT INT DINT LINT REAL String type structure	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type structure	tag	Initial element to be overwritten by the Source
Length	SINT INT DINT	SINT INT DINT	immediate tag	Number of Destination elements to copy

Structured Text

Operand	CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers Data Type	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Format	Description
Source	SINT INT DINT LINT REAL String type structure	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type structure REF_TO_AXIS_VIRTUAL REF_TO_AXIS_CONSUMED REF_TO_AXIS_GENERIC_DRIVE REF_TO_AXIS_SERVO REF_TO_AXIS_SERVO_DRIVE REF_TO_AXIS_CIP_DRIVE	tag	Initial element to copy
Dest	SINT INT DINT LINT REAL String type structure	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type structure	tag	Initial element to be overwritten by the Source
Length	SINT INT DINT	SINT INT DINT	immediate tag	Number of Destination elements to copy

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Index Through Arrays* for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The instruction copies the data.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is true in Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

During execution of the COP and CPS instructions, other controller actions may try to interrupt the copy operation and change the source:

If the source or destination is:	And need to:	Then select:	Notes
• produced tag • consumed tag • I/O data • data that another task can overwrite • non-atomic tag that is written to by a remote device	Prevent the source data from changing during the copy operation	CPS	Tasks that attempt to interrupt a CPS instruction are delayed until the instruction is done. To estimate the execution time of the CPS instruction, refer to the ControlLogix System User Manual, publication 1756-UM001. Use interlock application code to ensure a remote client is not updating the source while the CPS instruction is executing.

If the source or destination is:	And need to:	Then select:	Notes
	Allow the source data to change during the copy operation	COP	
None of the above	--------------->	COP	

The COP and CPS instructions operate on contiguous memory and perform a straight byte-to-byte memory copy.

When the Source and Dest are different data types the number of bytes copied equals the smaller of:

- Requested amount equals Length x (the number of bytes in a destination element)
- The number of bytes in the destination tag
- For
  Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, or GuardLogix 5580 controller
  s: the number of bytes in the source tag

Tip: The end of the destination or source tag is defined as the last byte of the base tag. If the tag is a structure, the end of the tag is the last byte of the last element of the structure. This means the COP and CPS instruction could write past the end of a member array but will never write past the end of the base tag.

IMPORTANT: Test and confirm that the instruction does not change data that it should not change.

Examples

Example 1

Copy an array.

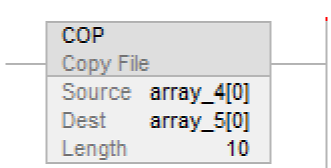
When enabled, the COP instruction copies 40 bytes from array_4 to array_5.

array_4 is a DINT (4 bytes per element) and contains 10 elements (total size = 40 bytes)

array_5 is a DINT (4 bytes per element) and contains 10 elements (total size = 40 bytes).

The Length says 10 destination elements should be copied so 40 bytes are copied.

Ladder Diagram



Structured Text

COP(array_4[0],array_5[0],10);

Example 2

Copy a structure.

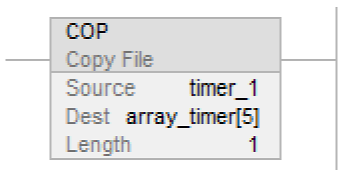
When enabled, the COP instruction copies the structure timer_1 into element 5 of array_timer.

timer_1 is a TIMER (total size = 12 bytes).

array_timer is a TIMER (12 bytes per element) and contains 10 elements (total size = 120 bytes).

The Length says 1 destination elements so 12 bytes are copied.

Ladder Diagram



Structured Text

COP(timer_1,array_timer[5],1);

Example 3

Copy array data while preventing the data from being changed until the copy is complete.

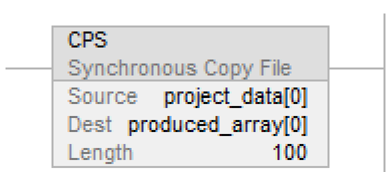
The project_data array (100 elements) stores a variety of values that change at different times in the application. To send a complete image of project_data at one instance in time to another controller, the CPS instruction copies project_data to produced_array. While the CPS instruction copies the data, no I/O updates or other tasks can change the data. The produced_array tag produces the data on a ControlNet network for consumption by other controllers.

project_data is a DINT (4 bytes per element) and contains 100 elements (total size = 400 bytes)

produced_array is a DINT (4 bytes per element) and contains 100 elements (total size = 400 bytes).

The Length says 100 destination elements so 400 bytes are copied.

Ladder Diagram



Structured Text

CPS(project_data[0],produced_array[0],100);

Example 4

Copy data to a produced tag while preventing the data from being sent until the copy is complete.

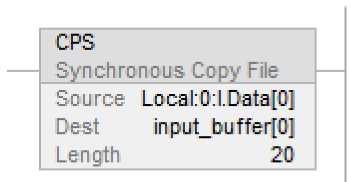
Local:0:I.Data stores the input data for the DeviceNet network that is connected to the 1756-DNB module in slot 0. To synchronize the inputs with the application, the CPS instruction copies the input data to input_buffer. While the CPS instruction copies the data, no I/O updates can change the data. As the application executes, it uses for its inputs the input data in input_buffer.

Local:0:I.Data is a DINT (4 bytes per element) and contains 2 elements (total size = 8 bytes)

input_buffer is a DINT (4 bytes per element) and contains 20 elements (total size = 80 bytes).

The Length says 20 destination elements should be copied (4 X 20 = 80 bytes). However the source can only provide 8 bytes so 8 bytes are copied.

Ladder Diagram

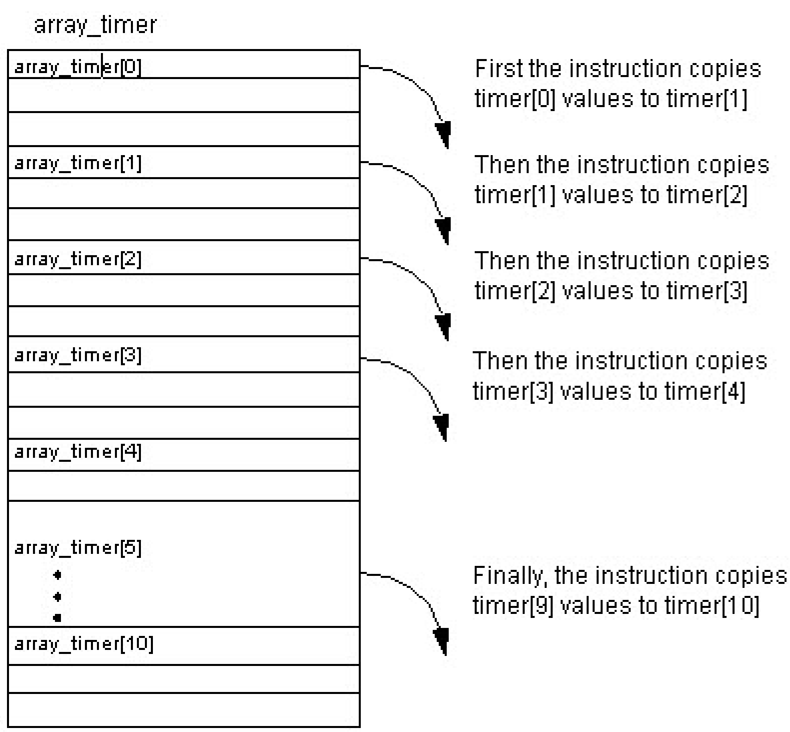


Structured Text

CPS(Local:0:I.Data[0], input_buffer[0], 20);

Example 5

Initialize an array structure, initialize the first element and the use COP to replicate it to the rest of the array.

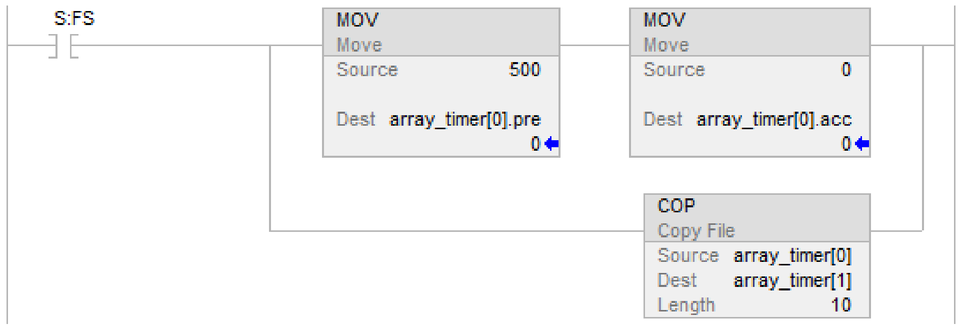


This example initializes an array or timer structures. When enabled, the MOV instructions initialize the .PRE and .ACC values of the first array_timer element. When enabled, the COP instruction copies a contiguous block of bytes, starting at array_timer[0]. The length is nine timer structures.

array_timer is a TIMER (12 bytes per element) and contains 15 elements (total size = 180 bytes)

The Length says 10 destination elements so 120 bytes are copied.

Ladder Diagram



Structured Text

IF S:FS THEN

array_timer[0].pre := 500;

array_timer[0].acc := 0;

COP(array_timer[0],array_timer[1],10);

END_IF;

Example 6

Copy different sized arrays.

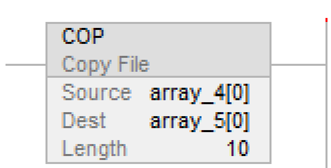
When enabled, the COP instruction copies bytes from SINT array_6 to DNT array_7.

array_6 is a SINT (1 byte per element) and contains 5 elements (total size = 5 bytes)

array_7 is a DINT (4 bytes per element) and contains 10 elements (total size = 40 bytes).

The Length says 20 destination elements should be copied (4 X 20 = 80 bytes). However the dest can only accept 40 bytes and the source can only provide 5 bytes so 5 bytes are copied.

Ladder Diagram



Structured Text

COP(array_4[0],array_5[0],10);

File Arithmetic and Logic (FAL)

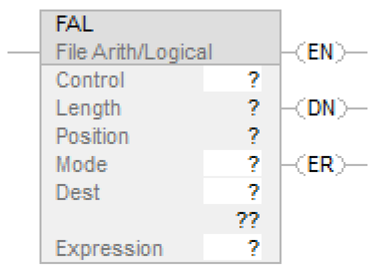
This instruction applies to the Compact GuardLogix 5370 and Compact GuardLogix 5380, CompactLogix 5370, CompactLogix 5380, and CompactLogix 5480, ControlLogix 5570 and ControlLogix 5580, and GuardLogix 5570 and GuardLogix 5580 controllers.

The FAL instruction performs copy, arithmetic, logic, and function operations on data stored in an array. When the rung-condition-in of the FAL instruction transitions from false to true, the expression given will be executed over the specified mode of iteration.

There are rules for allowable operators in safety applications. See Valid Operators.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Control	CONTROL	CONTROL	Tag	Control structure for the operation
Length	DINT	DINT	Immediate	This represents the CONTROL structure .LEN
Position	DINT	DINT	Immediate	This represents the CONTROL structure .POS
Mode	DINT	DINT	Immediate	Shows how to distribute the operation. Select INC, ALL, or enter number in the range of 1 to 2147483647
Expression	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate Tag	An expression consisting of tags and/or immediate values separated by operators.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Tag	The value of the Expression will be stored in destination.

Length and Position (corresponding to .LEN and .POS in the control tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the FAL instruction is enabled.
.DN	BOOL	The done bit is set when the instruction has operated on the last element (.POS = .LEN).
.ER	BOOL	When an overflow occurs, both platforms will set .ER and sop executing the instruction. The following controllers will generate an overflow: • CompactLogix 5370 • ControlLogix 5570
.LEN	DINT	The length specifies the number of elements in the array on which the FAL instruction operates.
.POS	DINT	The position is initialized to 0 when the instruction starts and is incremented each time the loop operates.

The value of the expression is stored in the specified destination tag. When an overflow occurs, it will set the ER bit and stop the execution. Once FAL completes all of the configured iterations, the .DN bit will be set.

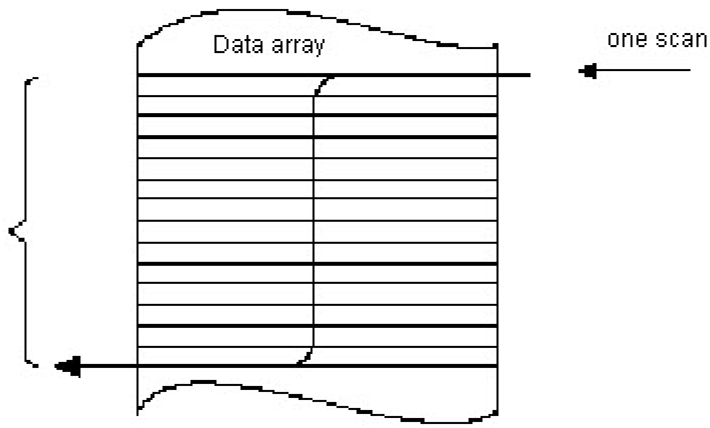
Select Mode of Operation

For FAL instructions, the mode tells the controller how to distribute the array operation.

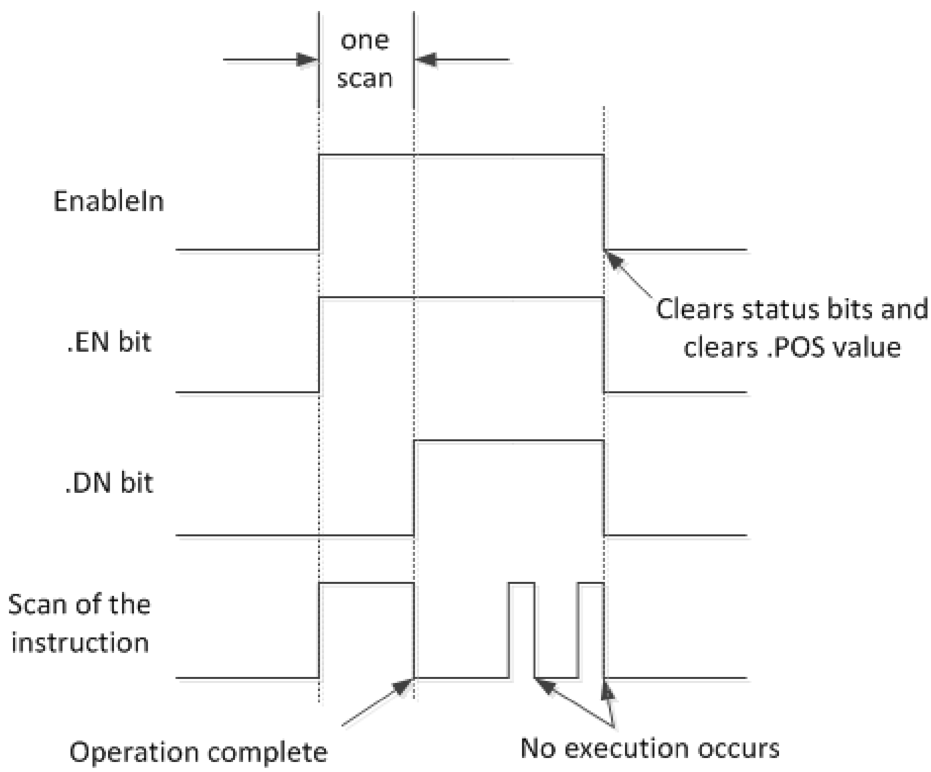
If:	Select this mode:
Operating on all of the specified elements in an array before continuing to the next instruction.	All
Distributing array operation over a number of scans. Enter the number of elements to operate on per scan (1-2147483647).	Numerical
Manipulating one element of the array each time the EnableIn goes from false to true.	Incremental

All Mode

In All Mode, the instruction operates on all the specified elements of the array before continuing to the next instruction. The operation begins when the instruction’s EnableIn goes from false to true. The position (.POS) value in the control structure points to the element in the array that the instruction is currently using. Operation stops when the .POS value equals or exceeds the .LEN value, and when overflow occurs in the expression and the .ER bit is set to true.



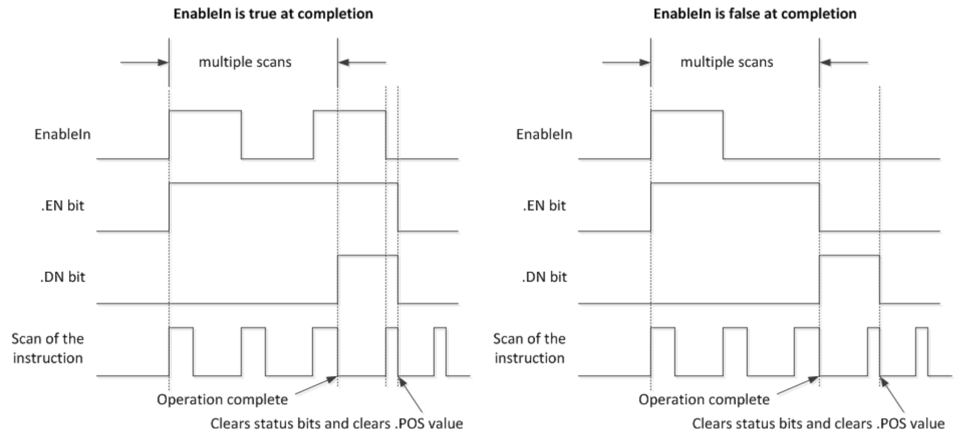
The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is true. The .DN bit, the .EN bit, and the .POS value are cleared when the EnableIn is false. Only then can another execution of the instruction be triggered by a false-to-true transition of EnableIn.



Numerical Mode

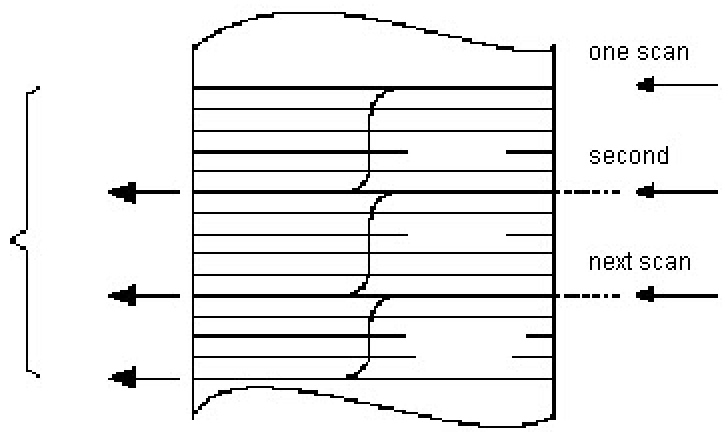
Numerical mode distributes the array operation over a number of scans. Use this mode when working with non-time-critical data or large amounts of data. Enter the number of elements to operate on for each scan, which keeps scan time shorter.

Execution is triggered when the EnableIn goes from false to true. Once triggered, the instruction is executed each time it is scanned for the number of scans necessary to complete operating on the entire array. Once triggered, EnableIn can change repeatedly without interrupting execution of the instruction.



Avoid using the results of a file instruction operating in numerical mode until the .DN bit is set.

The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is set.

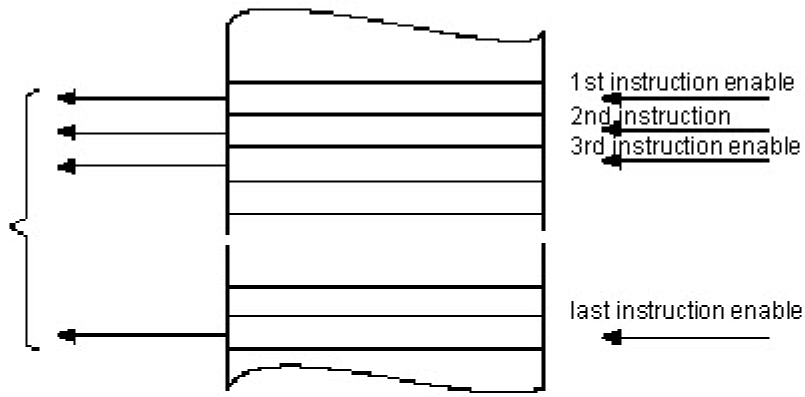


If the EnableIn is true at completion, the .EN and .DN bit are true until the EnableIn goes false. When the EnableIn goes false, these bits are cleared and the .POS value is cleared.

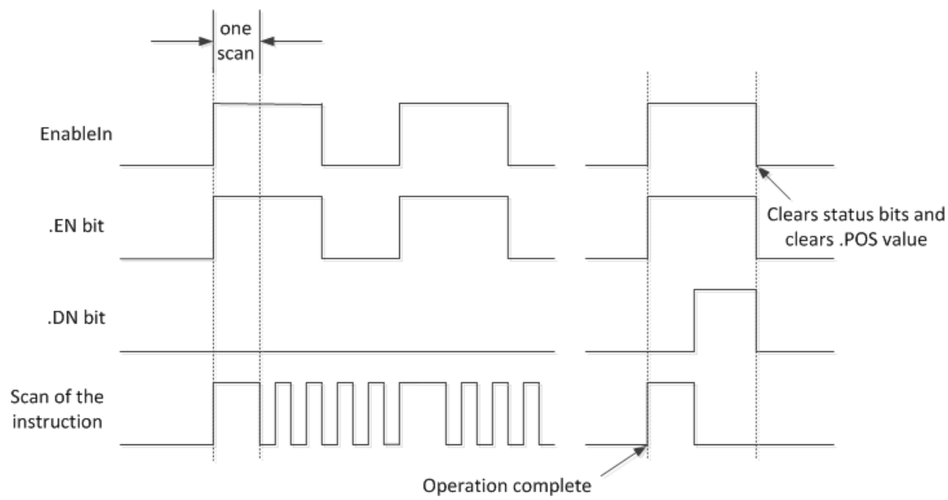
If the EnableIn is false at completion, the .EN bit is cleared immediately. One scan after the .EN bit is cleared, the .DN bit and the .POS value are cleared.

Incremental Mode

Incremental mode manipulates one element of the array each time the instruction's EnableIn goes from false to true.



The following timing diagram shows the relationship between status bits and instruction operation. Execution occurs only in a scan in which the EnableIn goes from false to true. Each time this occurs, only one element of the array is manipulated. If the EnableIn remains true for more than one scan, the instruction only executes during the first scan.



The .EN bit is set when EnableIn is true. The .DN bit is set when the last element in the array has been manipulated. When the last element has been manipulated and the EnableIn goes false, the .EN bit, the .DN bit, and the .POS value are cleared.

The difference between incremental mode and numerical mode at a rate of one element per scan is:

Numerical mode with any number of elements per scan requires only one false-to-true transition of the EnableIn to start execution. The instruction continues to execute the specified number of elements each scan until completion regardless of the state of the EnableIn.

Incremental mode requires the EnableIn to change from false to true to manipulate one element in the array.

Format expressions

For each operator that you use in an expression, you must provide one or two operands (tags or immediate values). Use the following table to format operators and operands within an expression.

For operators that operate on:	Use this format:	Example
One operand	operator(operand)	ABS(tag)
Two operands	operand_a operator operand_b	tag_b + 5 tag_c AND tag_d (tag_e**2) MOD (tag_f / tag_g)

Determine the order of operation

The operations in the expression are performed by the instruction in a prescribed order, not necessarily the order they appear. The order of operation can be specified by grouping terms within parentheses, forcing the instruction to perform an operation within the parentheses ahead of other operations.

Operations of equal order are performed from left to right.

Order	Operation
1	( )
2	ABS, ACOS, ASIN, ATAN, COS, DEG, BCD_TO, LN, LOG, RAD, SIN, SQRT, TAN, TO_BCD, TRUNC
3	**
4	- (negate), NOT
	*, /, MOD
6	- (subtract), +
7	AND
8	XOR
9	OR

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.POS < 0 or .LEN < 0	4	21

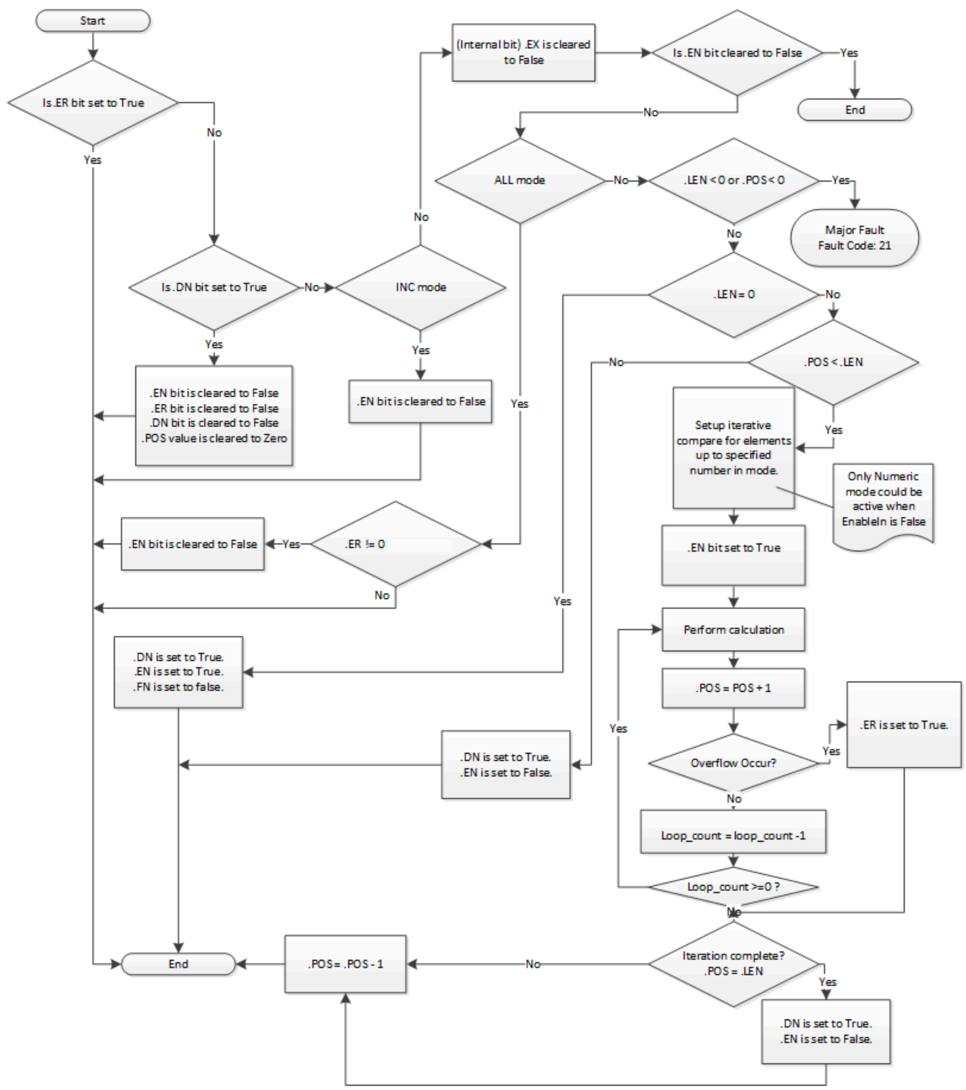
See Index Through Arrays for array-indexing faults.

Execution

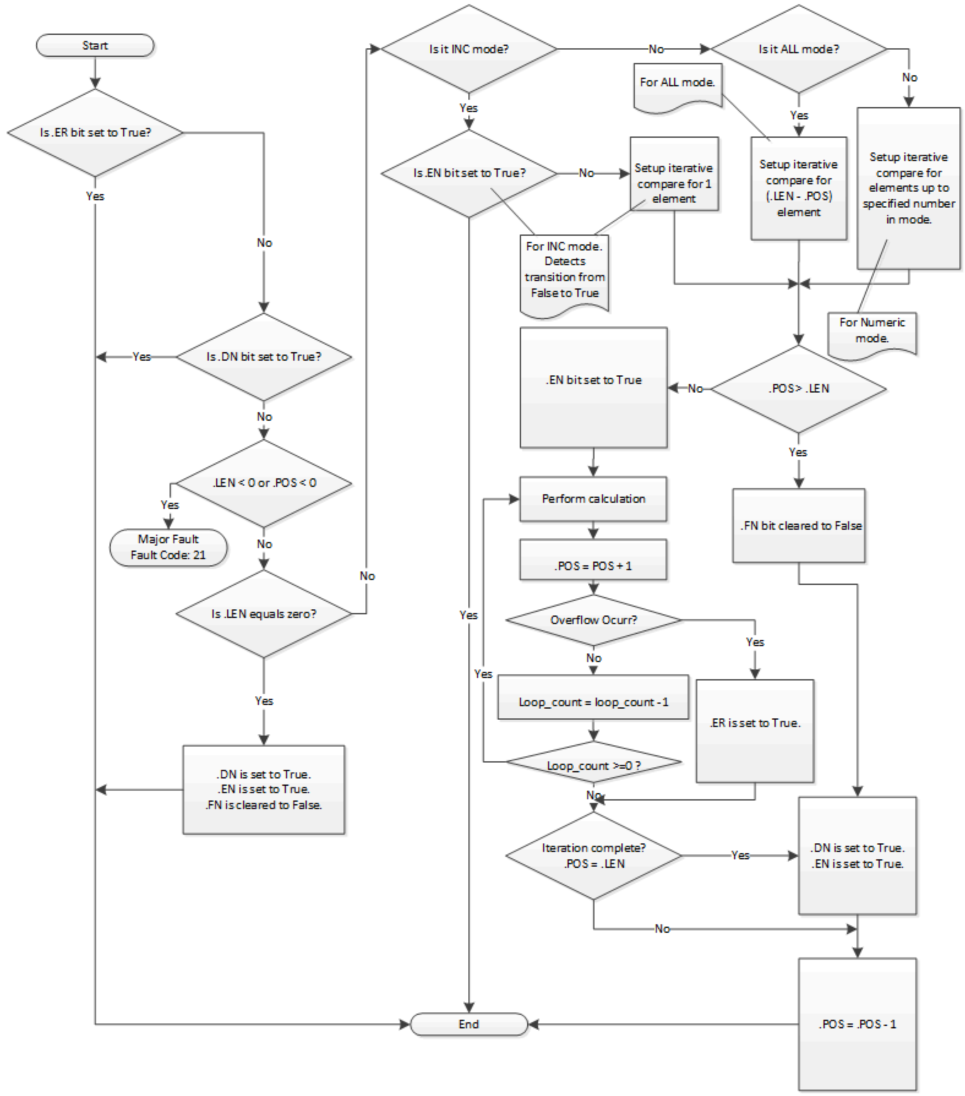
Ladder Diagram

Condition / State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in. See FAL Flow Chart (Rung-condition-out is False)
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. See FAL Flow Chart (Rung-condition-out is True)
Postscan	N/A

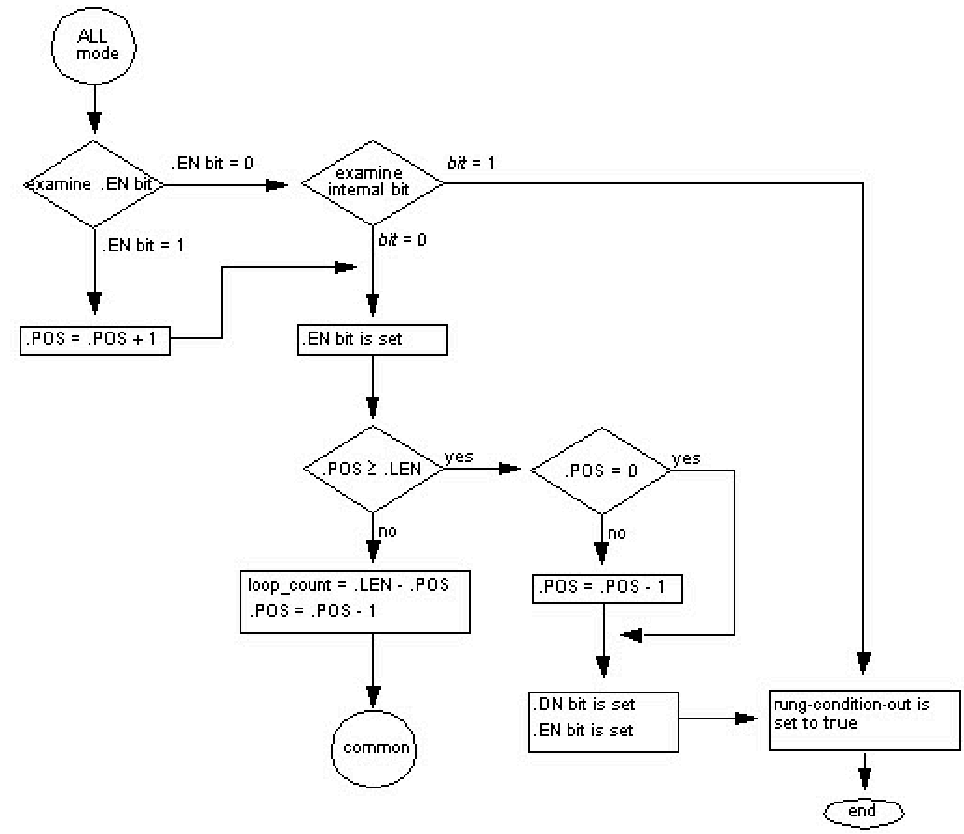
FAL Flow Chart (Rung-condition-out is False)



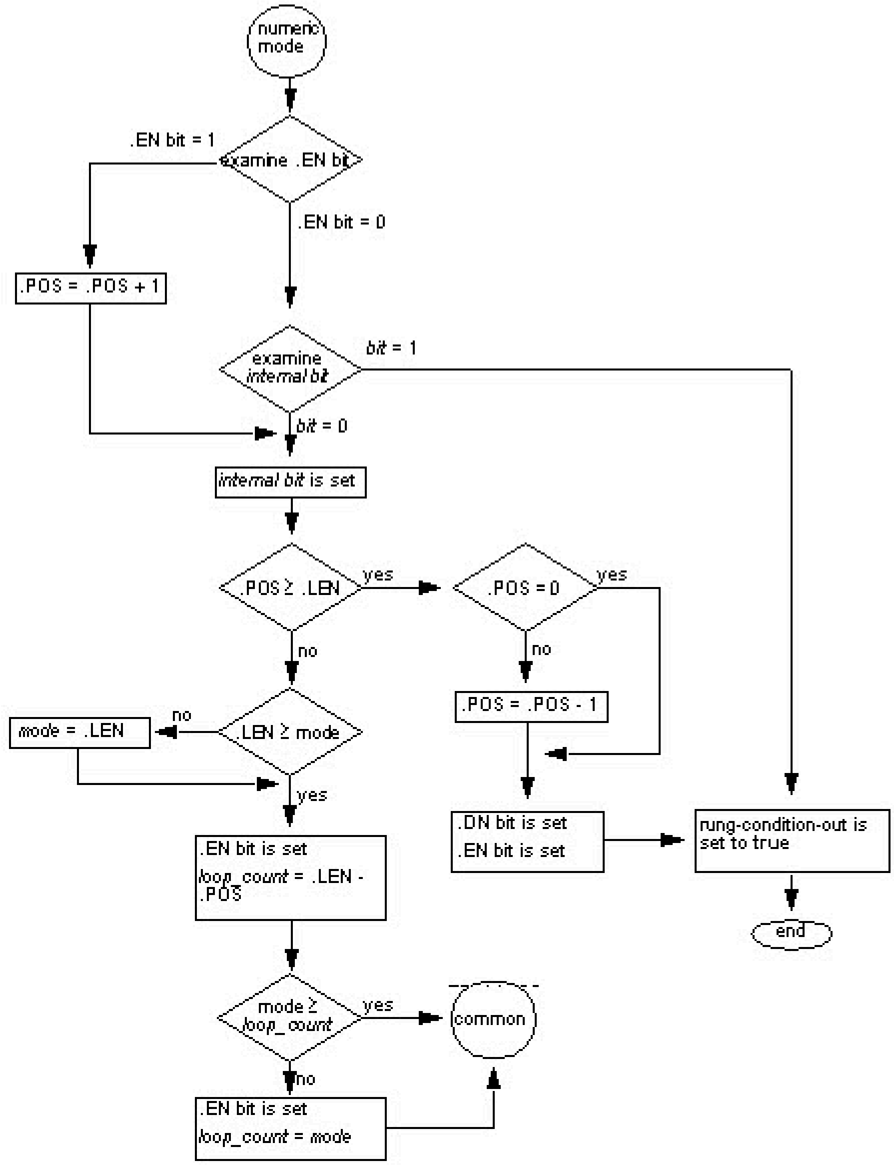
FAL Flow Chart (Rung-condition-out is True)



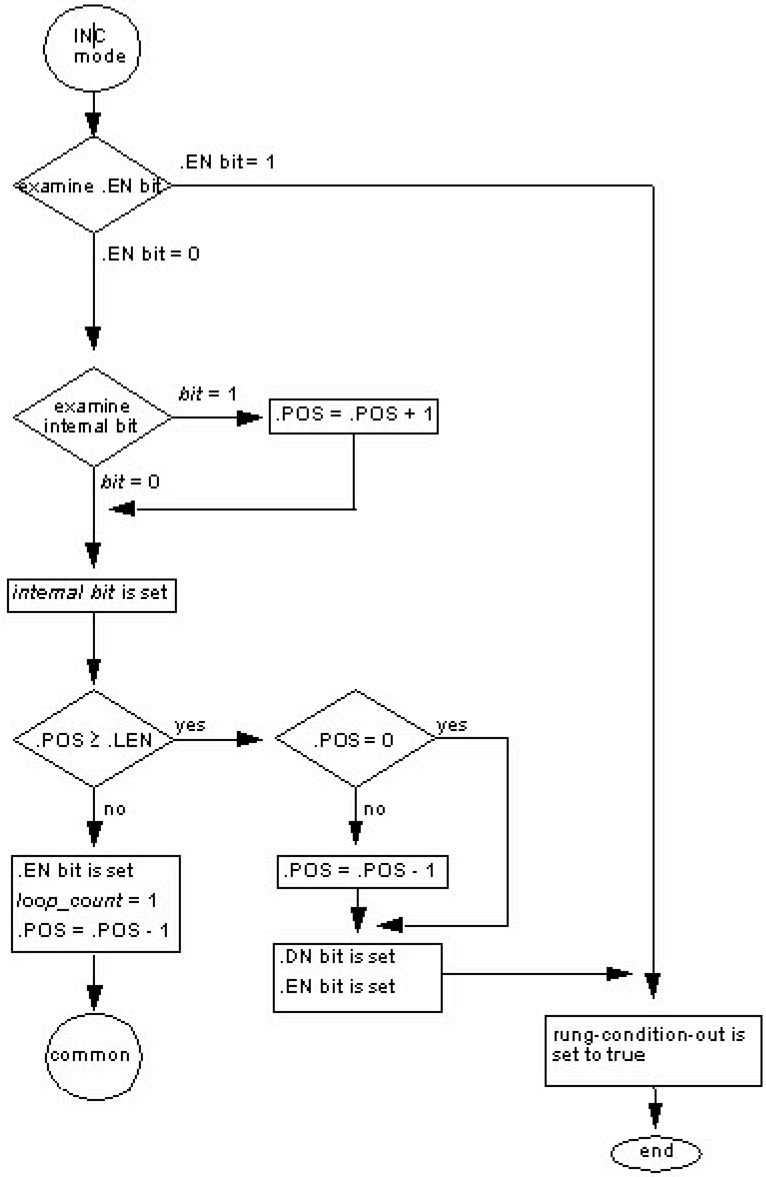
FAL Flow Chart (All Mode)



FAL Flow Chart (Numerical Mode)



FAL Flow Chart (Incremental Mode)

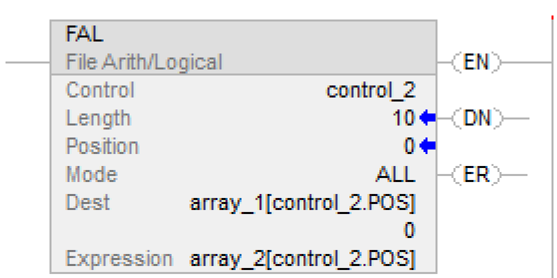


Examples

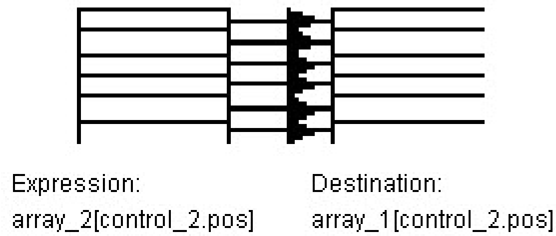
Example 1

Array-to-array.

Ladder Diagram



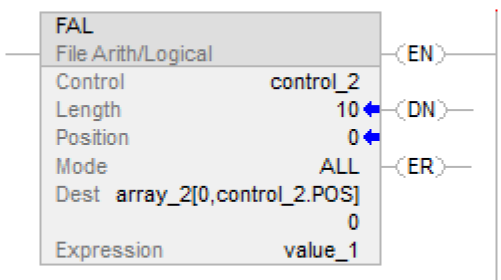
When enabled, the FAL instruction copies each element of array_2 into the same position within array_1.



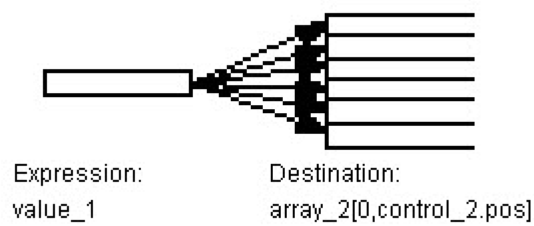
Example 2

Element-to-array copy.

Ladder Diagram

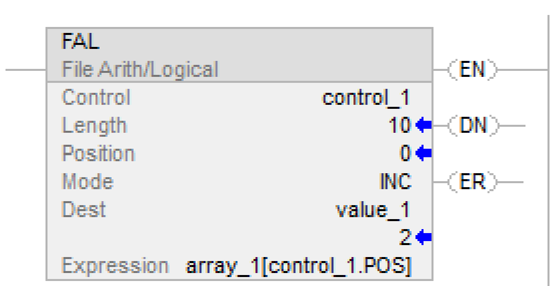


When enabled, the FAL instruction copies value_1 into the first 10 positions of the second dimension of array_2.

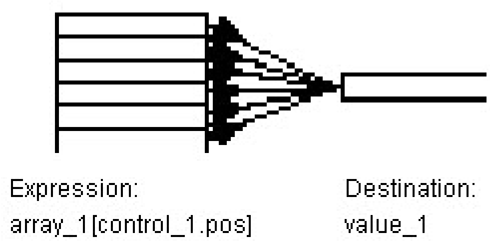


Example 3:

Array-to-element copy.

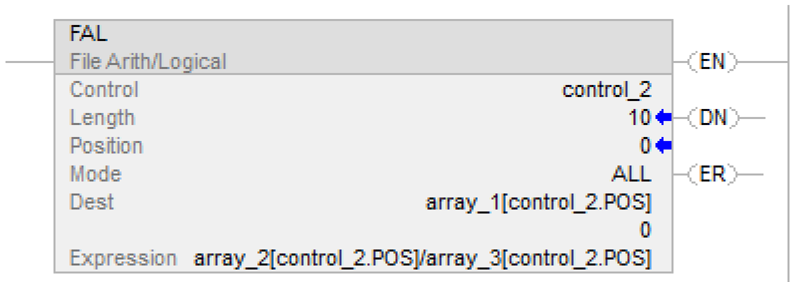


Each time the FAL instruction is enabled, it copies the current value of array_1 to value_1. The FAL instruction uses incremental mode, so only one array value is copied each time the instruction is enabled. The next time the instruction is enabled, the instruction overwrites value_1 with the next value in array_1.

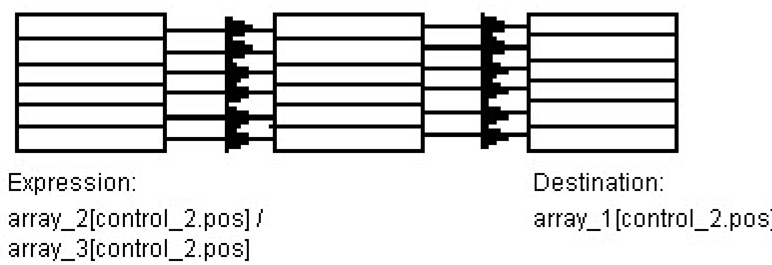


Example 4:

Arithmetic operation: array / array to array

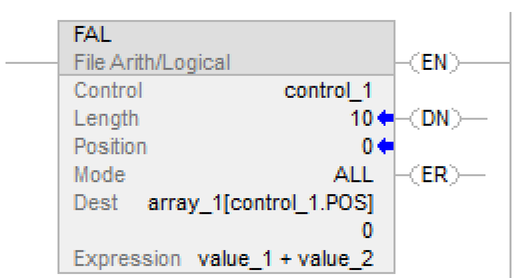


When enabled, the FAL instruction divides the value in the current position of array_2 with the value in the current position of array_3 and stores the result in the current position of array_1.

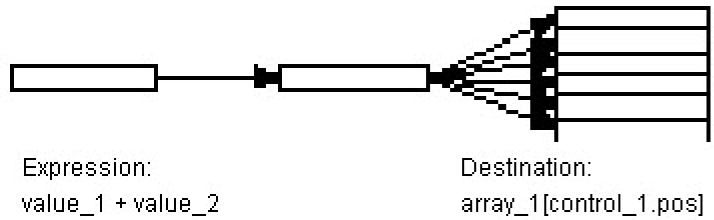


Example 5:

Arithmetic operation: array / array to array

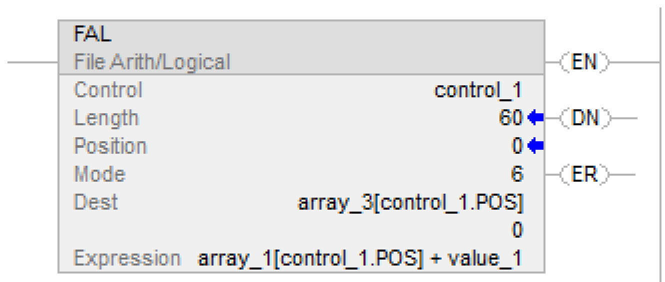


When enabled, the FAL instruction adds value_1 and value_2 and stores the result in the current position of array_1.

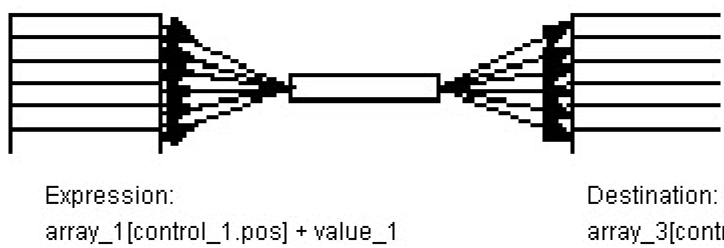


Example 6:

Arithmetic operation: array + element to array

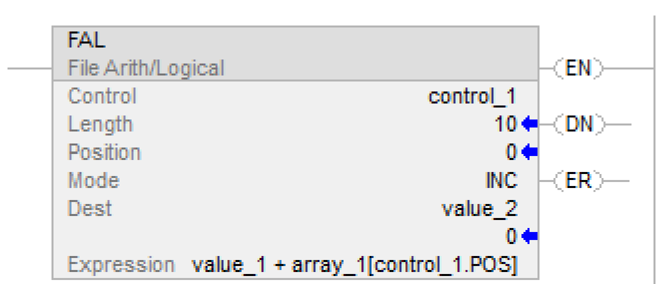


When enabled, the FAL instruction adds the value at the current position in array_1 to value_1 and stores the result in the current position in array_3. The instruction must execute 10 times for the entire array_1 and array_3 to be manipulated.

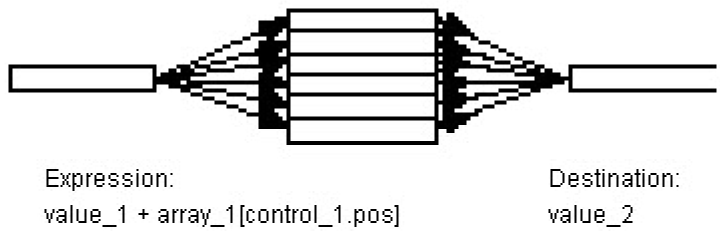


Example 7:

Arithmetic operation: (element + array) to element

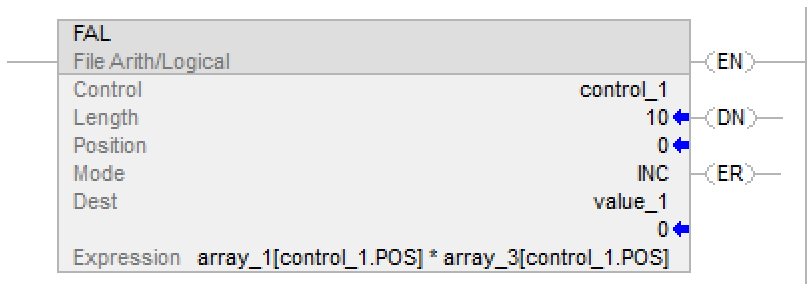


Each time the FAL instruction is enabled, it adds value_1 to the current value of array_1 and stores the result in value_2. The FAL instruction uses incremental mode, so only one array value is added to value_1 each time the instruction is enabled. The next time the instruction is enabled, the instruction overwrites value_2.

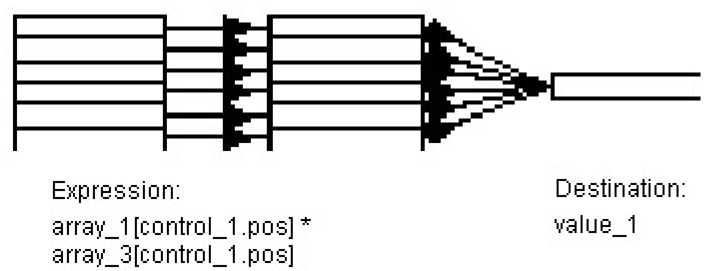


Example 8:

Arithmetic operation: (array * array) to element



When enabled, the FAL instruction multiplies the current value of array_1 by the current value of array_3 and stores the result in value_1. The FAL instruction uses incremental mode, so only one pair of array values is multiplied each time the instruction is enabled. The next time the instruction is enabled, the instruction overwrites value_1.



File Average (AVE)

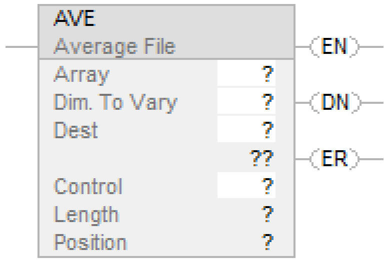
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The AVE instruction calculates the mean of a set of values.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram

Operand	Type	Format	Description
Array Tag	SINT INT DINT REAL	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 of the dimensions is: array[0,1,2]
Destination	SINT INT DINT 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	Offset into the specified array which identifies the current

Operand	Type	Format	Description
			element that the instruction is accessing. initial value is typically 0

Length and Position (corresponding to .LEN and .POS in the control tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

Description

The AVE instruction calculates the average of a set of values.

IMPORTANT: Make sure the Length does not cause the instruction to exceed the specified Dimension to vary. If this happens, the destination will be incorrect. For more information, see Viewing an Array as a Block of Memory.

If an overflow occurs during expression evaluation, the instructions reads past the end of an array, the instruction sets the ER bit and stops execution

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

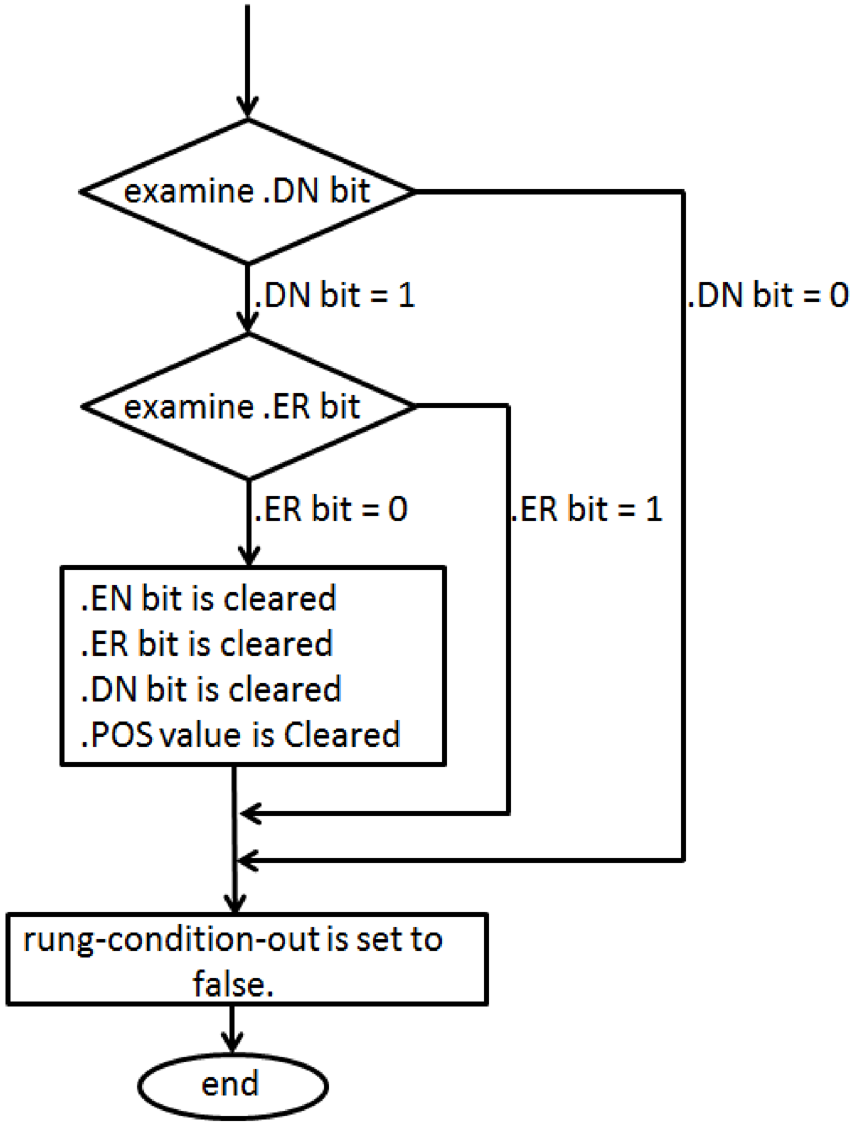
None specific to this instruction. See Common Attributes for operand related faults.

Execution

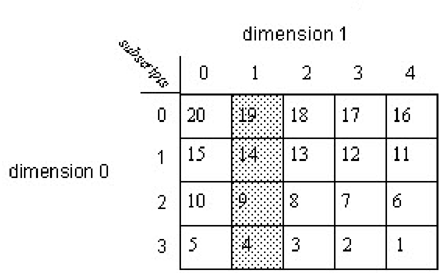
Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN bit is cleared. The .DN bit is cleared. If .ER bit is zero during prescan, all the control bits (.DN, .EN, .EU, .EM, .UL, .IN and .FD) will be cleared to zero.
Rung-condition-in is false.	See AVE Flow Chart (False)
Rung-condition-in is true.	The AVE instruction calculates the average by adding all the specified elements in the array and dividing by the number of elements.
Postscan	N/A.

AVE Flow Chart (False)



Example 1

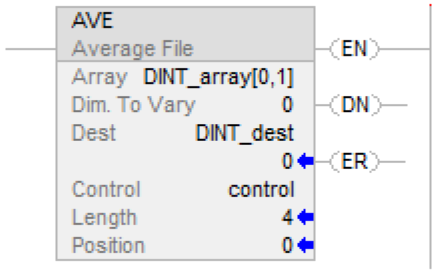


subscripts	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

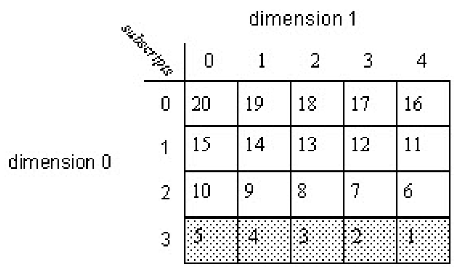
$$AVE = \frac{19+14+9+4}{4} = \frac{46}{4} = 11.5$$

dint_ave = 12

Ladder Diagram



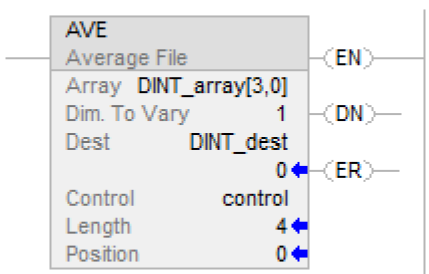
Example 2



$$AVE = \frac{5+4+3+2+1}{5} = \frac{15}{5} = 3$$

dint_ave = 3

Ladder Diagram



File Fill (FLL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, CompactLogix 5380, CompactLogix 5480, and ControlLogix 5580 controllers. Controller differences are noted where applicable.

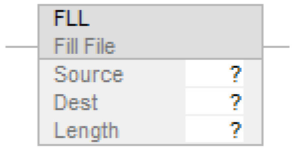
The FLL instruction fills a block of memory with the provided source value. The Source remains unchanged.

If the destination array is SINT, INT, DINT, or REAL, and the type of source value is different, the source value will be converted to the destination type before it is stored. Smaller integer types will be converted to large ones by sign-extension.

If the destination array is a structure, the source value will be written without conversion.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data Conversions.

Ladder Diagram

Operand	Data Type	Format	Description
Source	SINT INT DINT REAL	immediate tag	Element to copy
Destination	SINT INT DINT REAL structure	tag	Initial element to be overwritten by the Source.
Length	DINT INT SINT	immediate tag	Number of destination elements to fill.

The number of bytes filled is the smaller of:

- Requested amount = Length x (number of bytes in a destination element)
- The number of bytes in the destination tag

Tip: The end of the destination tag is defined as the last byte of the base tag. If the tag is a structure, the end of the tag is the last byte of the last element of the structure. This means the FLL instruction could write past the end of a member array, but will never write past the end of the base tag. Test and confirm that the FLL instruction does not change data that should not be changed.

For best results, the Source and Destination should be the same type. Use FLL to fill a structure with a constant, such as 0s.

If initializing a structure, be sure to have one instance containing the initial values, and use COP to replicate it. FLL can be used, for example, zero out the entire structure.

If the Source is:	And the Destination is:	The Source is converted to:
SINT, INT, DINT, or REAL	SINT	SINT
SINT, INT, DINT, or REAL	INT	INT
SINT, INT, DINT, or REAL	DINT	DINT
SINT, INT, DINT, or REAL	REAL	REAL

Conversion from larger integers to smaller integers will result in truncation (the high bits are discarded). Once the source is converted, it is written to the destination N times, where N = byte count. Sign extension results when converting from smaller integers to larger integers. REAL numbers will be rounded when converted to integers.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

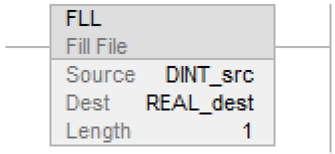
Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction fills the memory.
Postscan	N/A

Example

The FLL instruction copies number of destination elements specified by the Length from the DINT_src type source operand into a REAL_dest type destination.

Ladder Diagram



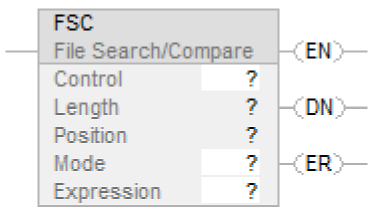
File Search and Compare (FSC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, CompactLogix 5380, CompactLogix 5480, and ControlLogix 5580 controllers. Controller differences are noted where applicable.

The FSC instruction compares values in an array, element by element.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Control	CONTROL	CONTROL	Tag	Control structure for the operation
Length	DINT	DINT	Immediate	This represents the CONTROL structure .LEN
Position	DINT	DINT	Immediate	This represents the CONTROL structure .POS
Mode	DINT	DINT	Immediate	Shows how to distribute the operation. Select INC, ALL, or enter number in the range of 1 to 2147483647
Expression	SINT INT DINT REAL STRING	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL String type	Immediate Tag	An expression consisting of tags and/or immediate values separated by operators

Length and Position (corresponding to .LEN and .POS in the control tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the FSC 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 not modified.
.IN	BOOL	The inhibit bit indicates the FSC instruction detected a true comparison.

Mnemonic	Data Type	Description
		You must clear this bit to continue the search operation.
.FD	BOOL	The found bit indicates the FSC instruction detected a true comparison.
.LEN	DINT	The length specifies the number of elements in the array on which the instruction operates.
.POS	DINT	The position contains the position of the current element that the instruction is accessing.

Description

When the EnableIn of the FSC instruction transitions from false to true, the expression is evaluated over the specified mode of iteration.

If the evaluation result is true, the instruction sets the .FD bit, and the .POS value reflects the array position where the instruction found the true comparison. The instruction sets the .IN bit to prevent further iteration.

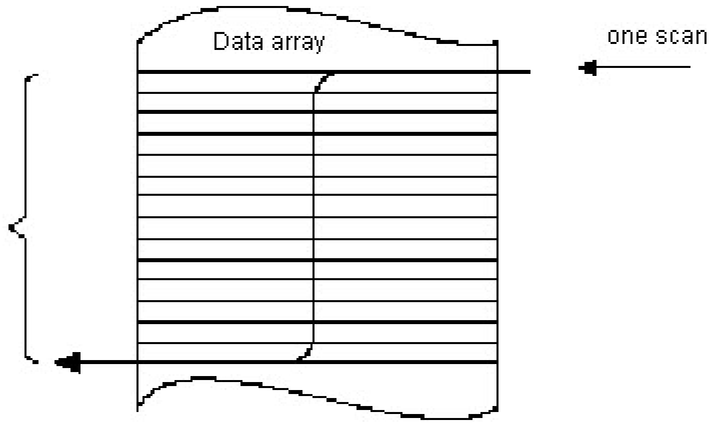
Select Mode of Operation

For FSC instructions, the mode tells the controller how to distribute the array operation.

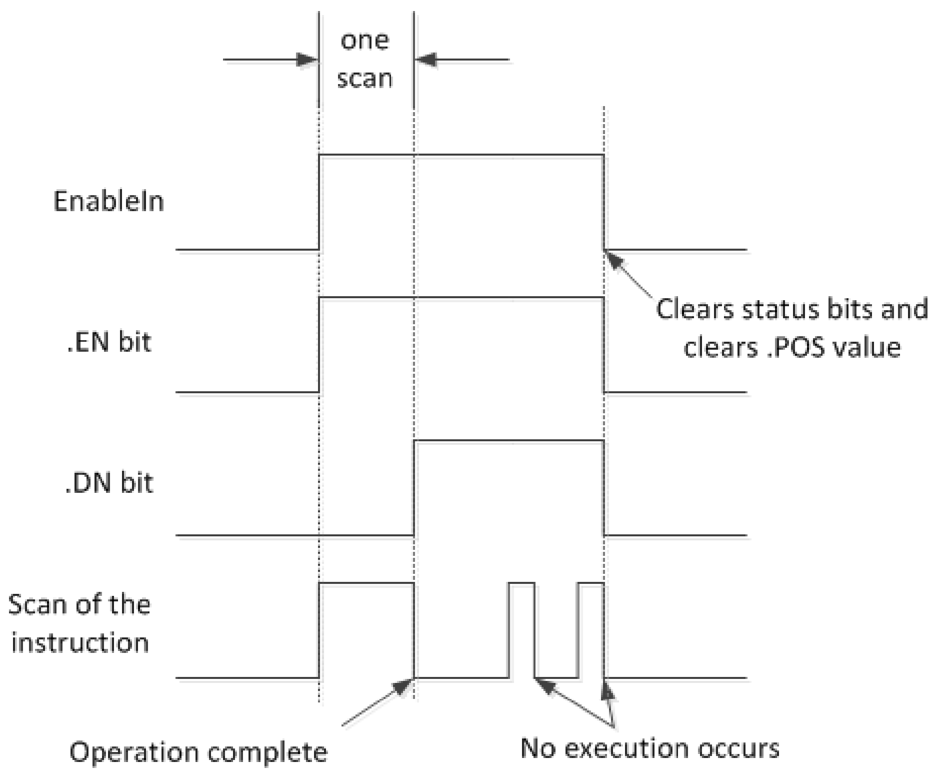
If you want to:	Select this mode:
Operate on all of the specified elements in an array before continuing on to the next instruction.	All
Distribute array operation over a number of scans. Enter the number of elements to operate on per scan (1-2147483647).	Numerical
Manipulate one element of the array each time the EnableIn goes from false to true.	Incremental

All Mode

In All mode, all the specified elements in the array are operated on before continuing on to the next instruction. The operation begins when the instruction's EnableIn goes from false to true. The position (.POS) value in the control structure points to the element in the array that the instruction is currently using. Operation stops under two conditions. When the .POS value equals or exceeds the .LEN value, AND when the expression evaluates to true.



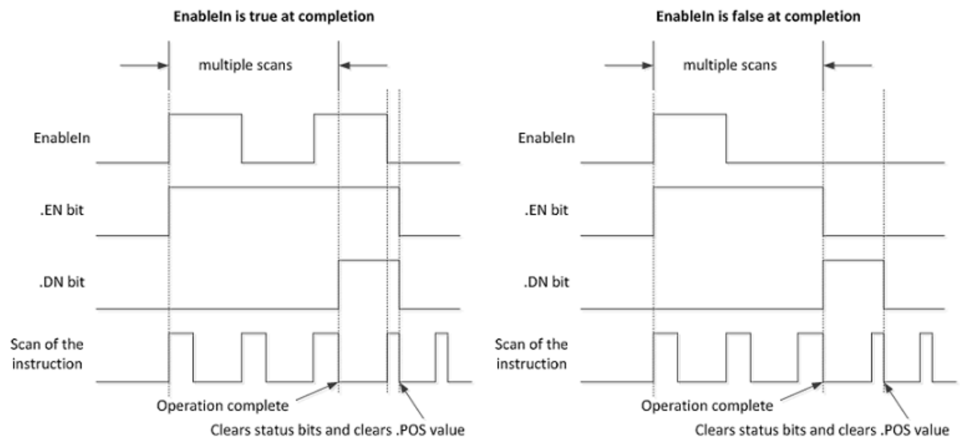
The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is true. The .DN bit, the .EN bit, and the .POS value are cleared when the EnableIn is false. Only then can another execution of the instruction be triggered by a false-to-true transition of EnableIn.



Numerical Mode

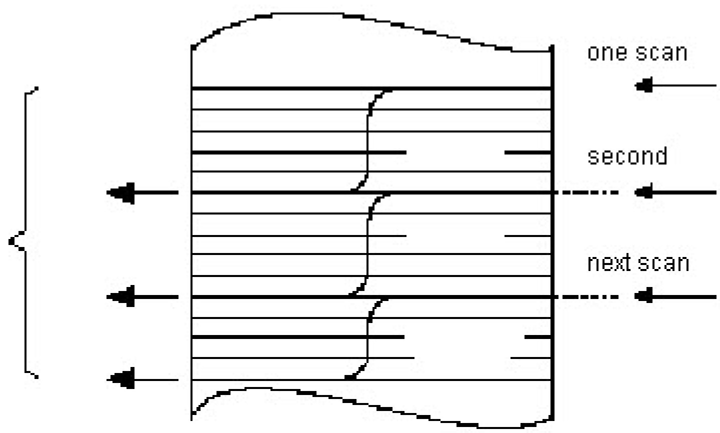
Numerical mode distributes the array operation over a number of scans. This mode is useful when working with non-time-critical data or large amounts of data. You enter the number of elements to operate on for each scan, which keeps scan time shorter.

Execution is triggered when the EnableIn goes from false to true. Once triggered, the instruction is executed each time it is scanned for the number of scans necessary to complete operating on the entire array. Once triggered, EnableIn can change repeatedly without interrupting execution of the instruction.



Avoid using the results of a file instruction operating in numerical mode until the .DN or .IN bit is true.

The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is true.

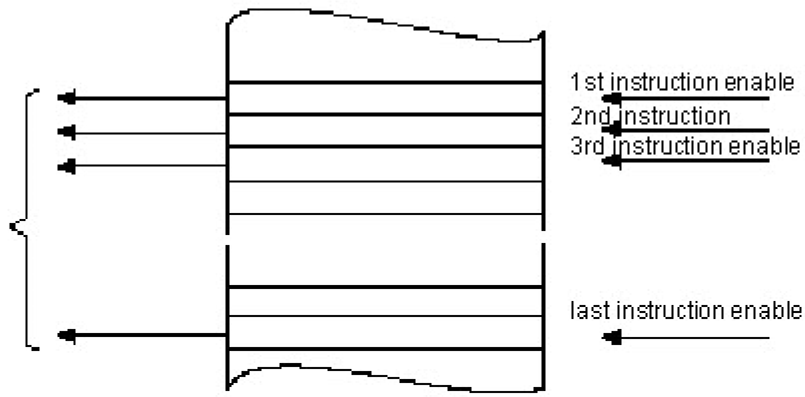


If the EnableIn is true at completion, the .EN and .DN bit are true until the EnableIn goes false. When the EnableIn goes false, these bits are cleared and the .POS value is cleared.

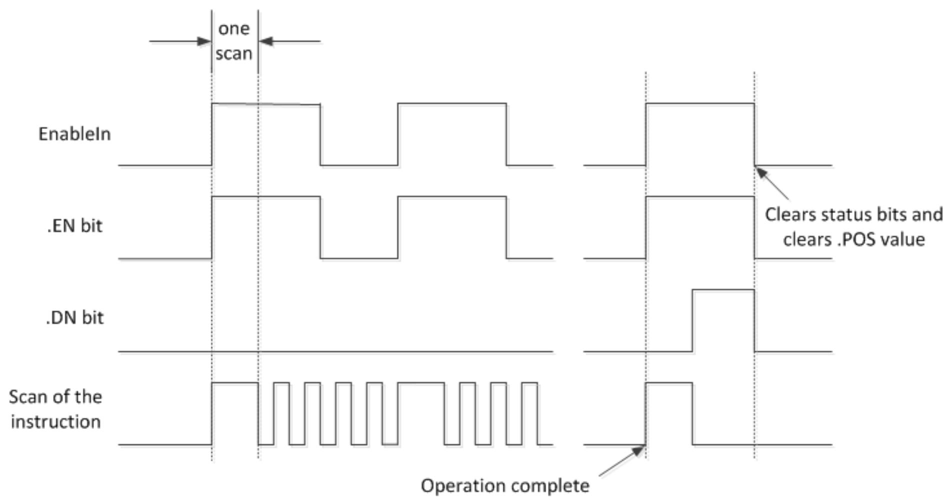
If the EnableIn is false at completion, the .EN bit is cleared immediately. One scan after the .EN bit is cleared, the .DN bit and the .POS value are cleared.

Incremental Mode

Incremental mode manipulates one element of the array each time the instruction's EnableIn goes from false to true.



The following timing diagram shows the relationship between status bits and instruction operation. Execution occurs only in a scan in which the EnableIn goes from false to true. Each time this occurs, only one element of the array is manipulated. If the EnableIn remains true for more than one scan, the instruction only executes during the first scan.



The .EN bit is set when rung-condition-in is true. The .DN bit is set when the last element in the array has been manipulated. When the last element has been manipulated and the rung-condition-in goes false, the .EN bit, the .DN bit, and the .POS value are cleared.

The difference between incremental mode and numerical mode at a rate of one element per scan is:

Numerical mode with any number of elements per scan requires only one false-to-true transition of the EnableIn to start execution. The instruction continues to execute the specified number of elements each scan until completion regardless of the state of the EnableIn.

Incremental mode requires the EnableIn to change from false to true to manipulate one element in the array.

Format expressions

For each operator that you use in an expression, you must provide one or two operands (tags or immediate values). Use the following table to format operators and operands within an expression.

For operators that operate on:	Use this format:	Example
One operand	operator(operand)	ABS(tag)
Two operands	operand_a operator operand_b	tag_b + 5 tag_c AND tag_d (tag_e**2) MOD (tag_f / tag_g)

Determine the order of operation

The operations you write into the expression are performed by the instruction in a prescribed order, not necessarily the order you write them. You can override the order of operation by grouping terms within parentheses, forcing the instruction to perform an operation within the parentheses ahead of other operations.

Operations of equal order are performed from left to right.

Order	Operation
1	( )
2	ABS, ACOS, ASIN, ATAN, COS, DEG, BCD_TO, LN, LOG, RAD, SIN, SQRT, TAN, TO_BCD, TRUNC
3	**
4	- (negate), NOT, !
5	*, /, MOD
6	- (subtract), +
7	AND
8	XOR
9	OR
10	<, <=, >, >=, =, <>
11	&&
12	^^
13	||

Use strings in an expression

To use strings of ASCII characters in an expression, follow these guidelines:

An expression lets you compare two string tags.

You cannot enter ASCII characters directly into the expression.

Only the following operands are permitted:

Operator	Description
=	Equal
<	Less than
<=	Less than or equal
>	Greater than

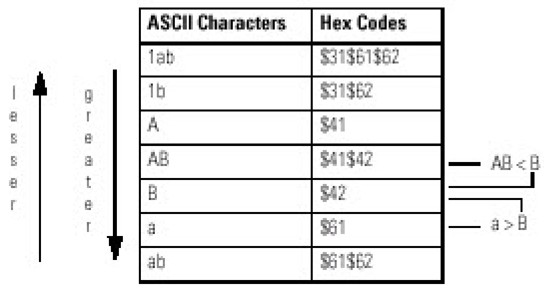
Operator	Description
>=	Greater than or equal
<>	Not equal

Strings are equal if their characters match.

ASCII characters are case-sensitive. Uppercase A ($41) is not equal to lowercase a ($61).

The hexadecimal values of the characters determine if one string is less than or greater than another string.

When the two strings are sorted as in a telephone directory, the order of the strings determine which one is greater.



Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.POS < 0 or .LEN < 0	4	21

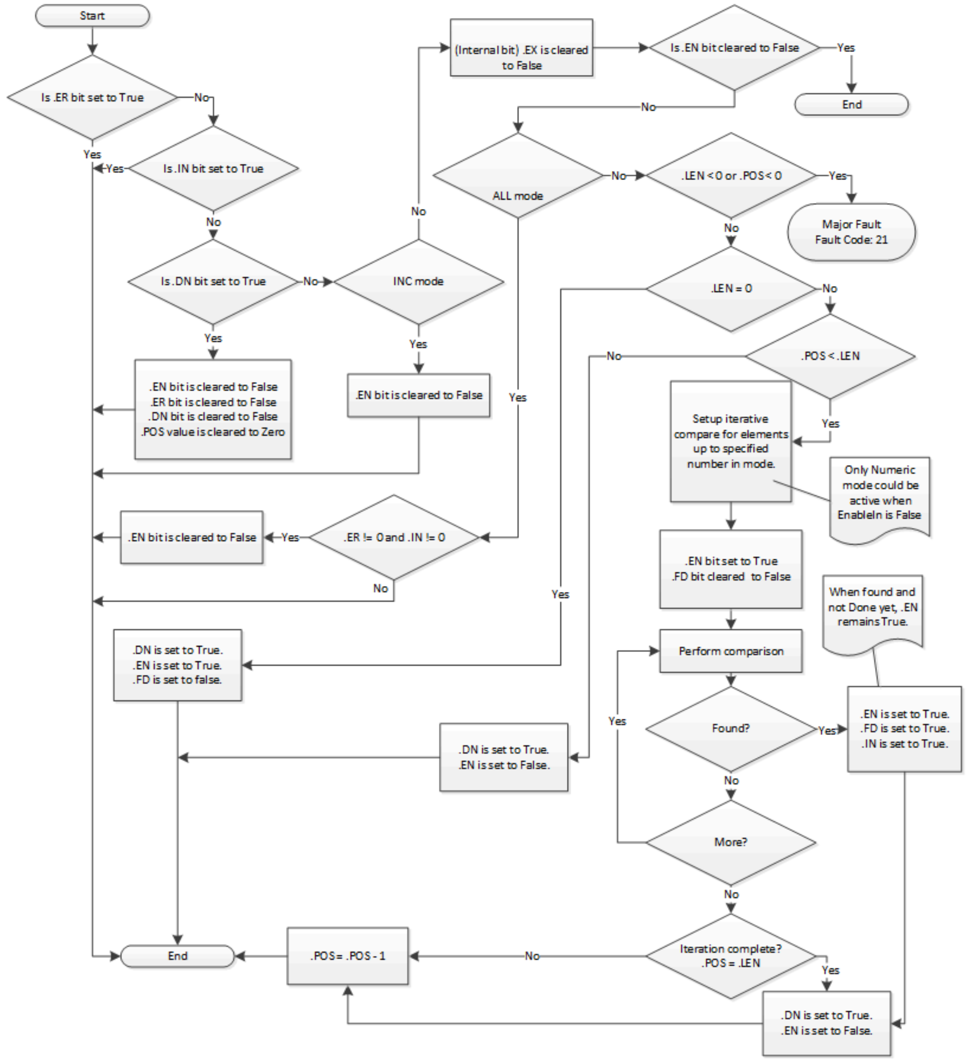
See Common Attributes for operand related faults. See Index Through Arrays for array-indexing faults.

Execution

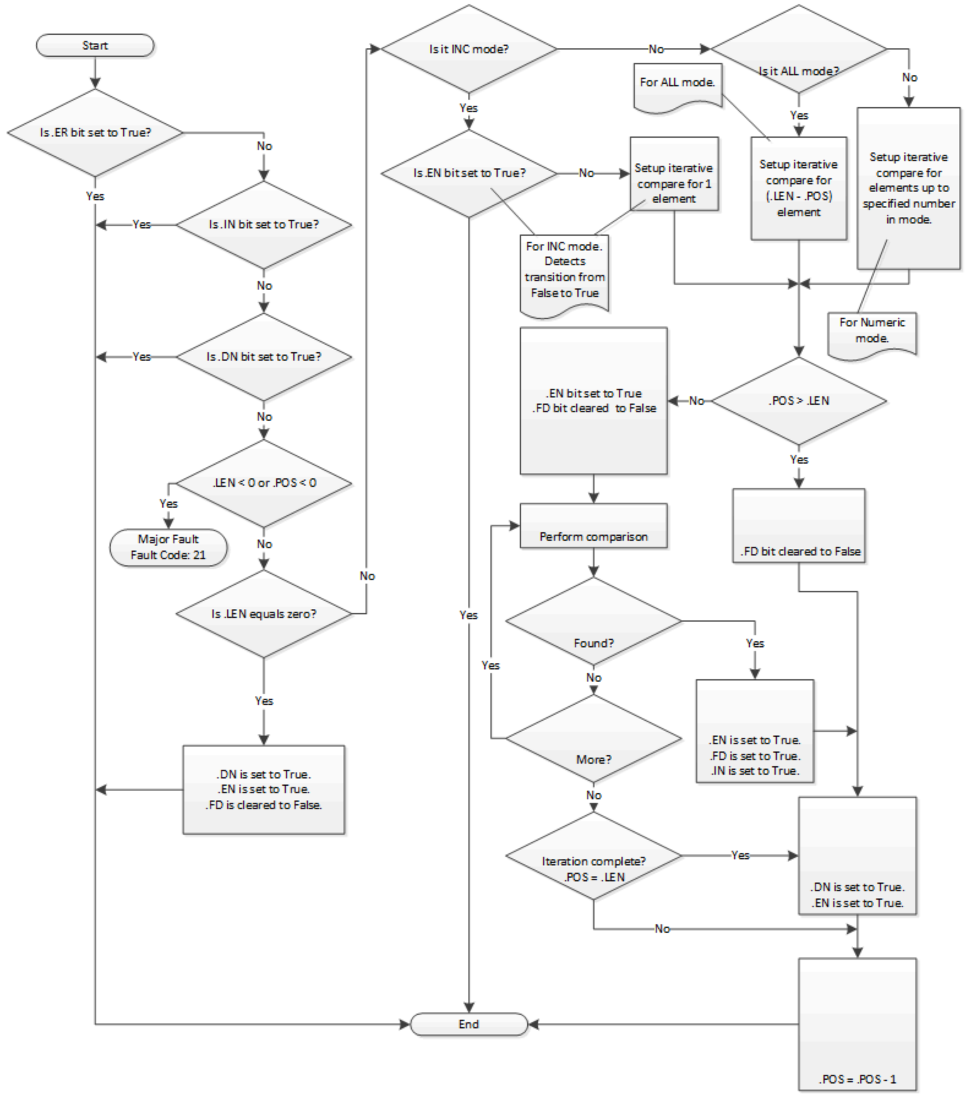
Ladder Diagram

Condition / State	Action Taken
Prescan	N/A
Rung-condition-in is false	See FSC Flow Chart (Rung-condition-out is False)
Rung-condition-in is true	See FSC Flow Chart (Rung-condition-out is True)
Postscan	N/A

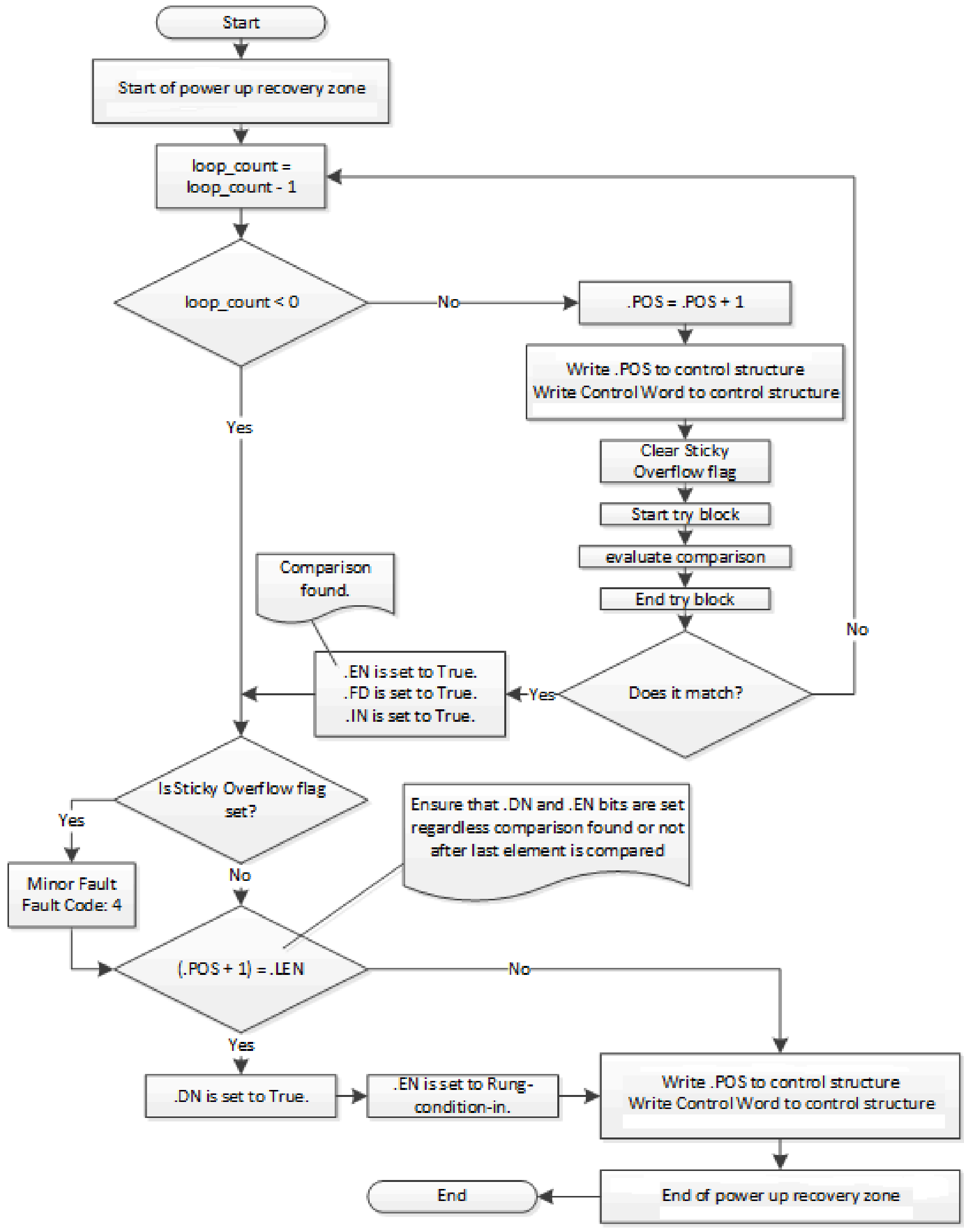
FSC Flow Chart (Rung-condition-out is False)



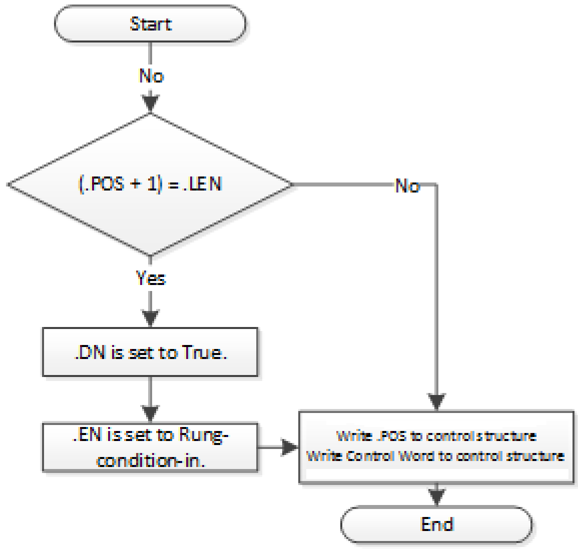
FSC Flow Chart (Rung-condition-out is True)



FSC Flow Chart (FSC Common Subflow)



FSC Flow Chart (FSC Common Exception Subflow)

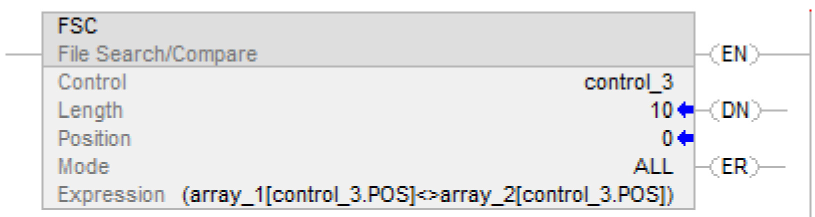


Examples

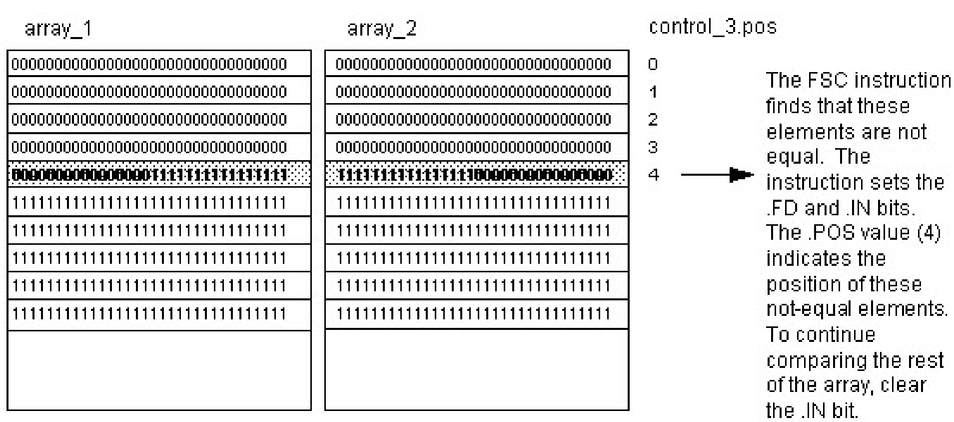
Example 1

Search between two DINT arrays for elements that are not equal.

Ladder Diagram



When enabled, the FSC instruction compares each of the first 10 elements in array_1 to the corresponding elements in array_2. When an element is found that is not equal, the FD and IN bits are set. The POS identifies the location of the not equal elements. Clear the IN bit to search the rest of the array.

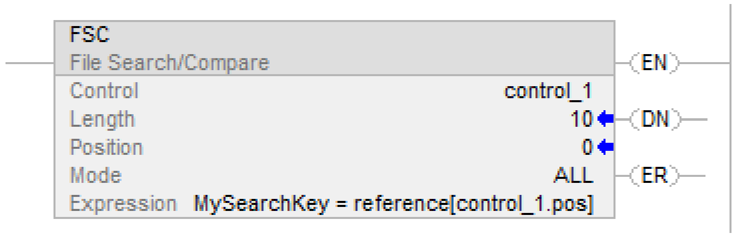


Example 2

Search for a string within a STRING array.

When enabled, the FSC instruction compares characters in code to 10 elements in code_table.

When a string in code_table is found that matches code, the FD and IN bits are set. The POS identifies the location of the matching strings. Clear the IN bit to search the rest of the array.



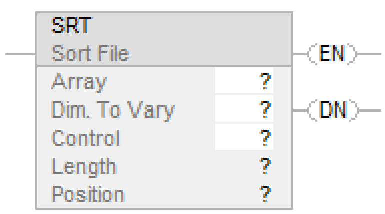
File Sort (SRT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, CompactLogix 5380, CompactLogix 5480, and ControlLogix 5580 controllers. Controller differences are noted where applicable.

The SRT instruction sorts a set of values in one dimension (Dim to vary) of the array into ascending order.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

SRT(Array,Dimtovary,Control);

Operands

Ladder Diagram

Operand	Type	Format	Description
Array	SINT INT DINT REAL	Array tag	array to sort specify the first element of the group of elements to sort

Operand	Type	Format	Description
Dimension to vary	DINT	Immediate (0, 1, 2)	which dimension to use the order of the dimensions is: array[0,1,2]
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

Length and Position (corresponding to .LEN and .POS in the control tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

Structured Text

Operand	Type	Format	Description
Array	SINT INT DINT REAL	Array tag	array to sort specify the first element of the group of elements to sort
Dimension to vary	DINT	Immediate (0, 1, 2)	which dimension to use the order of the dimensions is: array[0,1,2]
Control	CONTROL	Tag	control structure for the operation
Length	DINT	Immediate	Number of elements of the array to sort. The specified Length and Position values are accessed from the .LEN and .POS members of the CONTROL structure.
Position	DINT	Immediate	current element in the array initial value is typically 0 The specified Length and Position values are accessed from the .LEN and .POS members of the CONTROL structure.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the SRT instruction is enabled.
.DN	BOOL	The done bit is set when the instruction has operated on the last element in the Array.
.ER	BOOL	The error bit is set when either .LEN < 0 or .POS < 0. Either of these conditions also generates a major fault. When .ER bit is set, the instruction does not execute.
.LEN	DINT	The length word specifies the number of elements in the array on which the instruction operates.
.POS	DINT	The position word identifies the current element that the instruction is accessing.

Description

The SRT instruction sorts a set of values in one dimension (Dim to vary) of the Array into ascending order.

IMPORTANT: You must test and confirm that the instruction does not change data that you don't want it to change.

The SRT instruction operates on contiguous data memory. For the

CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers

only, the scope of the instruction is constrained by the base tag. The SRT instruction will not write data outside of the base tag but can cross member boundaries. If you specify an array that is a member of a structure, and the length exceeds the size of that array you must test and confirm that the SRT instruction does not change data you do not want changed.

In the

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

, the data is constrained by the specified member.

In this transitional instruction, the relay ladder toggles the rung-condition-in from false to true for the instruction to execute.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	No

Controllers	Affects Math Status Flags
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.POS < 0 or .LEN < 0	4	21
Dimension to vary > number of dimensions	4	20
Length > end of array	4	20

See *Common Attributes* for operand related faults.

Execution

Ladder Diagram

Condition / State	Action Taken
Prescan	N/A.
Rung-condition-in is false	.EN bit is cleared to false .EN bit is cleared to false .DN bit is cleared to false
Rung-condition-in is true	The instruction executes
Postscan	N/A.

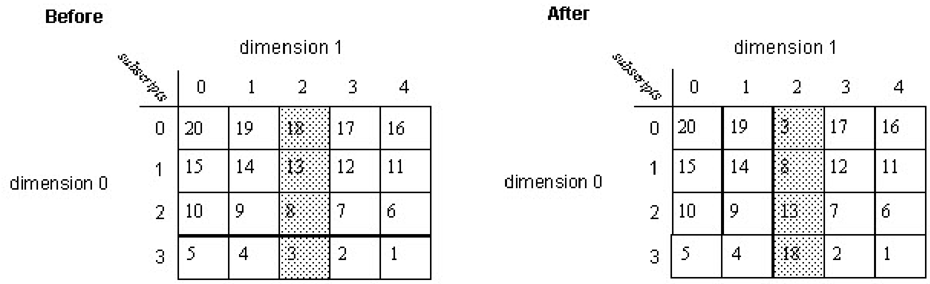
Structured Text

Condition / State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	Since this instruction requires a transition to execute it is executed false and then true. See the Ladder Diagram table for details.
Postscan	See Postscan in the Ladder Diagram table.

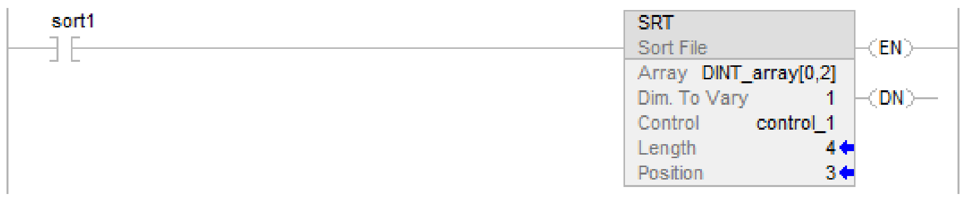
Examples

Example 1

Sort DINT_array, which is DINT[4,5].



Ladder Diagram



Structured Text

IF sort1 then

control_1.LEN := 4;

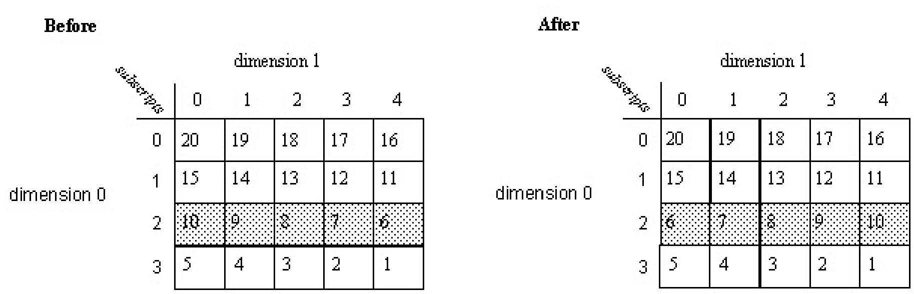
control_1.POS := 0;

SRT(DINT_array[0,2],0, control_1);

END_IF;

Example 2

Sort DINT_array, which is DINT[4,5].



Ladder Diagram

Structured Text

ctrl.LEN := 4;

ctrl.POS := 0;

SRT(DINT_array[0,2],0, ctrl);

File Standard Deviation (STD)

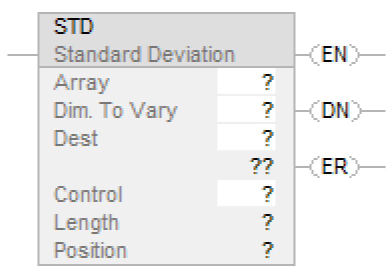
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

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.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram

Operand	Type	Format	Description
Array	SINT INT DINT 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
Dimension to vary	DINT	immediate (0, 1, 2)	which dimension to use the order of the dimensions is: array[0,1,2]
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	Offset into the specified array which identifies the current element that the instruction is accessing. initial value is typically 0

Length and Position (corresponding to .LEN and .POS in the control tag) are pseudo-operands. For details, see Pseudo-operand initialization on page 856.

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the STD instruction is enabled.
.DN	BOOL	The done bit is set when the instruction has operated on the last element in the Array.
.ER	BOOL	The error bit is set when the instruction generates an overflow. The instruction stops executing until the program clears the .ER bit. The .POS value stores the position of the element that caused the overflow.
.LEN	DINT	The length word specifies the number of elements in the array on which the instruction operates.

Mnemonic	Data Type	Description
.POS	DINT	The position word is an offset into the specified array which identifies the current element that the instruction is accessing.

Description

The standard deviation is calculated according to this formula:

$$\text{Standard Deviation} = \sqrt{\frac{\left(\sum_{i=1}^{N} [\langle X_{(start+i)} - AVE\rangle^2]\right)}{(N-1)}}$$

Where:

start = dimension-to-vary subscript of the array operand

xi = variable element in the array

N = number of specified elements in the array

$$AVE = \frac{\left(\sum_{i=1}^{N} x_{(start+i)}\right)}{N}$$

IMPORTANT: Make sure the Length does not cause the instruction to exceed the specified Dimension to vary. If this happens, the Destination will be incorrect.

If an overflow occurs during expression evaluation or if the instructions reads past the end of an array, the instruction sets the ER bit and stops execution.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, based on programming language. See Math Status Flags.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
.POS < 0 or .LEN < 0	4	21

A major fault will occur if:	Fault type	Fault code
Dimension to vary > number of dimensions	4	20

See Common Attributes for operand-related faults.

Execution

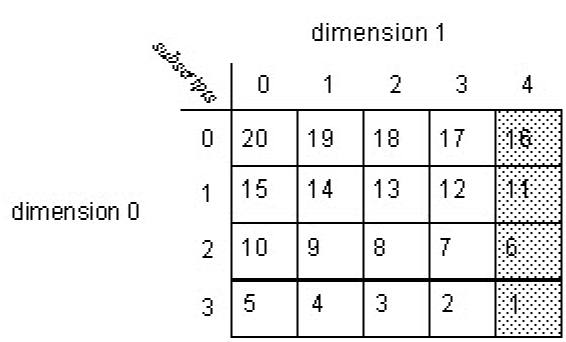
Ladder Diagram

Condition / State	Action Taken
Prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared.
Rung-condition-in is false	The .EN bit is cleared. The .ER bit is cleared. The .DN bit is cleared. The .POS value is cleared. The rung-condition-out is false.
Rung-condition-in is true	Internally, the instruction uses a FAL instruction to calculate the average: Expression = standard deviation calculation Mode = ALL
Postscan	N/A.

Examples

Example 1

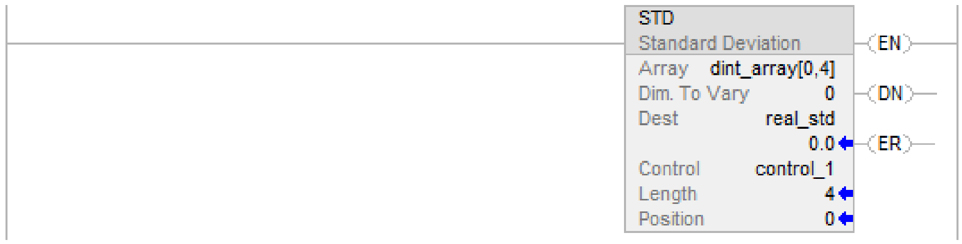
Calculate the standard deviation of arrayDint, which is DINT[4,5].



$$STD = \sqrt{\frac{(16-8.5)^2 + (11-8.5)^2 + (6-8.5)^2 + (1-8.5)^2}{(4-1)}} = 6.454972$$

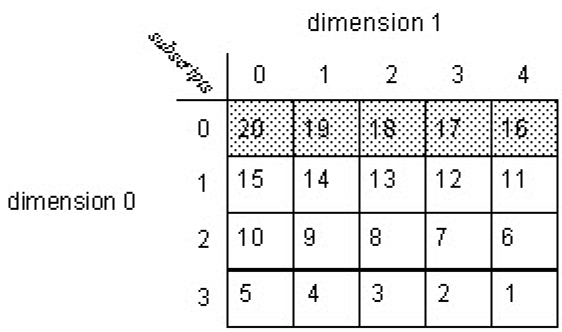
$$real\_std = 6.454972$$

Ladder Diagram

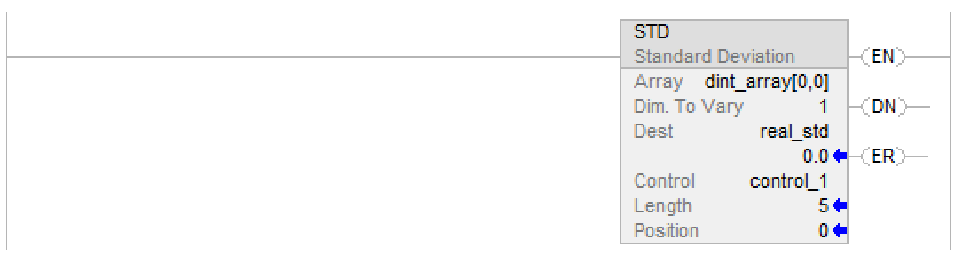


Example 2

Calculate the standard deviation of dint_array, which is DINT[4,5].



Ladder Diagram



Size In Elements (SIZE)

This information applies to the Compact GuardLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, Compact GuardLogix 5480, and ControlLogix 5580 controllers.

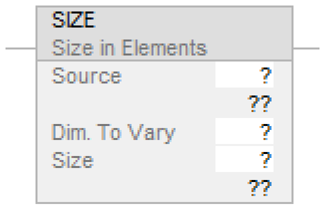
The SIZE instruction finds the number of elements (size) in the designated dimension of the Source array or string operand and places the result in the Size operand. The instruction finds the size of one dimension of an array.

The instruction operates on:

- Arrays
- Arrays in a structure
- Arrays that are part of a larger array
- String tags

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

SIZE(Source,Dimtovary,Size);

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See *Data Conversions*.

Ladder Diagram

<table>
<tr><th>Operand</th><th>Data Type</th><th>Format</th><th colspan="2">Description</th></tr>
<tr><td>Source</td><td>SINT
INT
DINT
REAL
structure
String type</td><td>Array tag</td><td colspan="2">First element of the array on which the instruction is to operate
Tags that are not array are not accepted during verification</td></tr>
<tr><td rowspan="4">Dimension to Vary</td><td rowspan="4">DINT</td><td rowspan="4">immediate

(0, 1, 2)</td><td colspan="2">Dimension to use:</td></tr>
<tr><td>For the size of:</td><td>Enter:</td></tr>
<tr><td>first dimension</td><td>0</td></tr>
<tr><td>second dimension</td><td>1</td></tr>
</table>

Operand	Data Type	Format	Description	
			third dimension	2
Size	SINT INT DINT REAL	tag	Tag to store the number of elements in the specified dimension of the array	

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Index Through Arrays* for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. The instruction executes.
Postscan	N/A

Structured Text

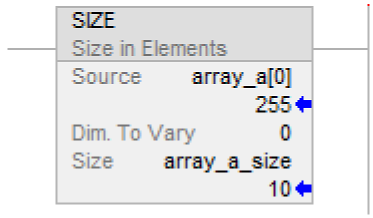
Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See rung-condition-in is true in Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Examples

Example 1

Find the number of elements in dimension 0 (first dimension) of array_a. Store the size in array_a_size. In this example, dimension 0 of array_a has 10 elements.

Ladder Diagram



Structured Text

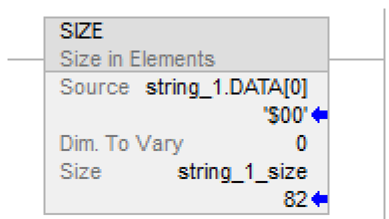
SIZE(array_a,0,array_a_size);

Example 2

Find the number of elements in the DATA member of string_1, which is a string. Stores the size in string_1_size.

In this example, the DATA member of string_1 has 82 elements. The string uses the default STRING data type. Since each element holds one character, string_1 can contain up to 82 characters.

Ladder Diagram



Structured Text

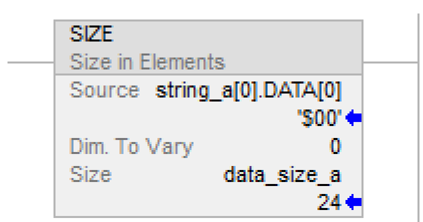
SIZE(string_1.DATA[0],0,string_1_size);

Example 3

String_a is an array of string structures. The SIZE instruction finds the number of elements in the DATA member of the string structure and stores the size in data_size_a.

In this example, the DATA member has 24 elements. The string structure has a user-specified length of 24.

Ladder Diagram

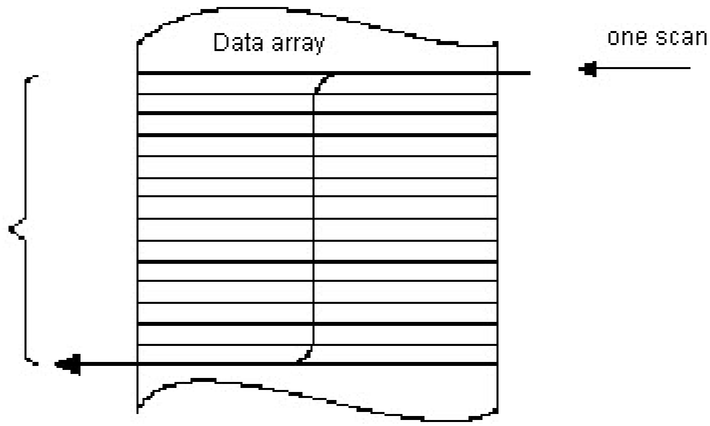


Structured Text

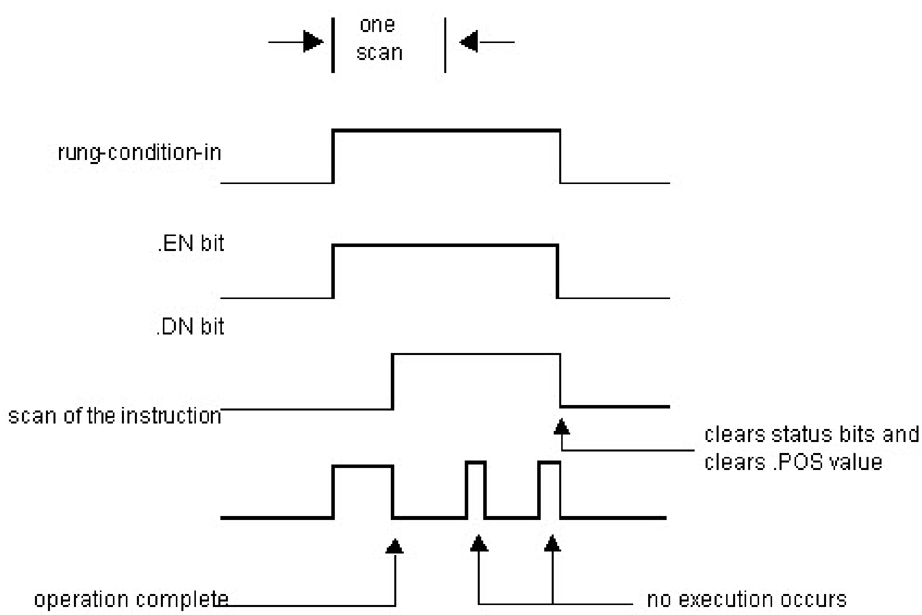
SIZE(string_a.[0].DATA[0],0,data_size_a);

All Mode

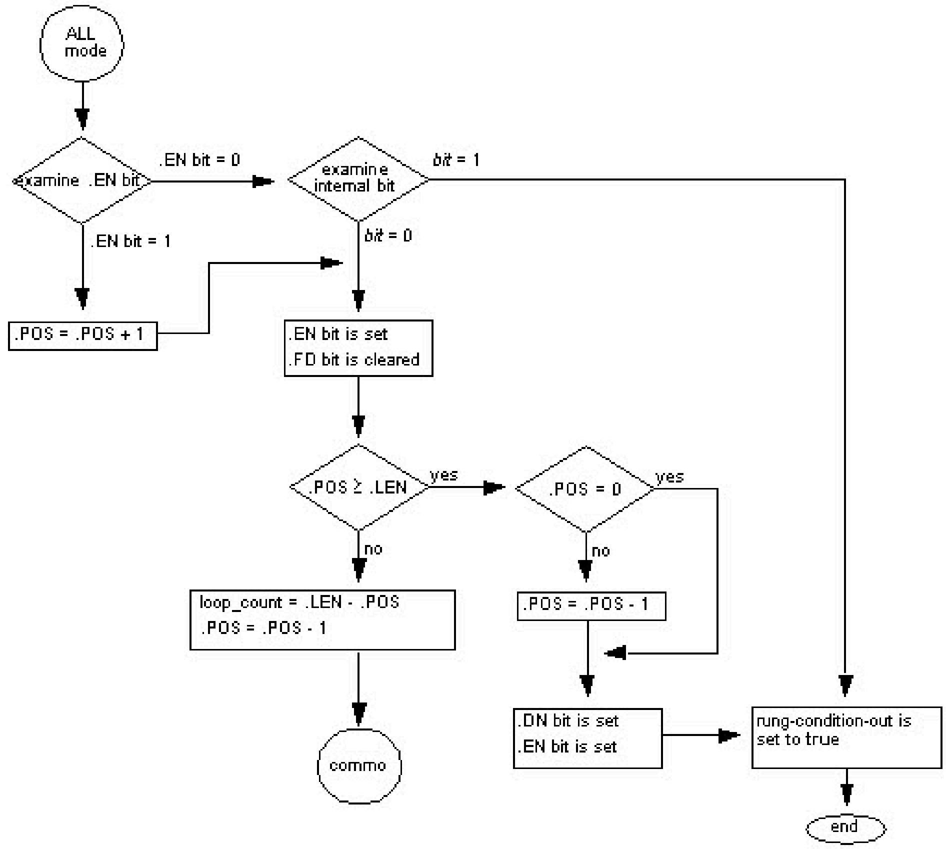
In All mode, all the specified elements in the array are operated on before continuing on to the next instruction. The operation begins when the instruction's rung-condition-in goes from false to true. The position (.POS) value in the control structure points to the element in the array that the instruction is currently using. Operation stops when the .POS value equals the .LEN value.



The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is set. The .DN bit, the .EN bit, and the .POS value are cleared when the rung-condition-in is false. Only then can another execution of the instruction be triggered by a false-to-true transition of rung-condition-in



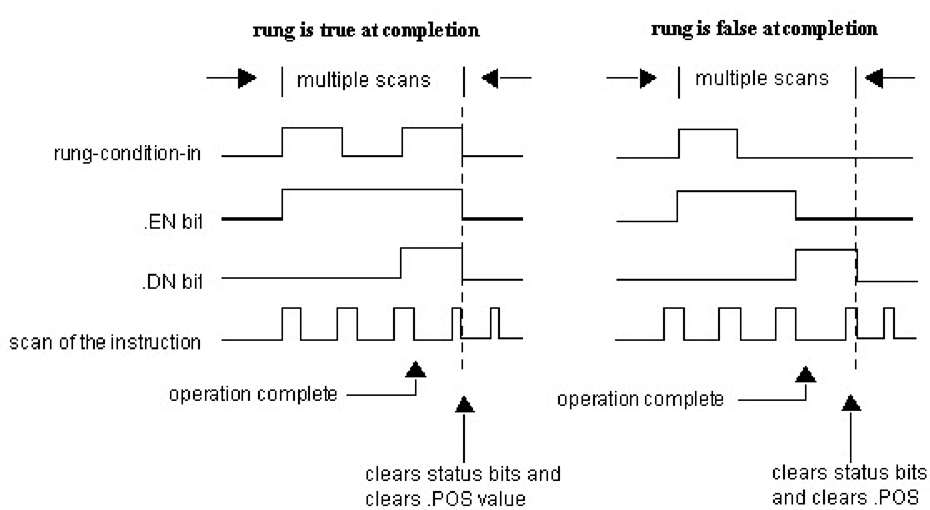
All Mode Flow Chart-FSC



Numerical Mode

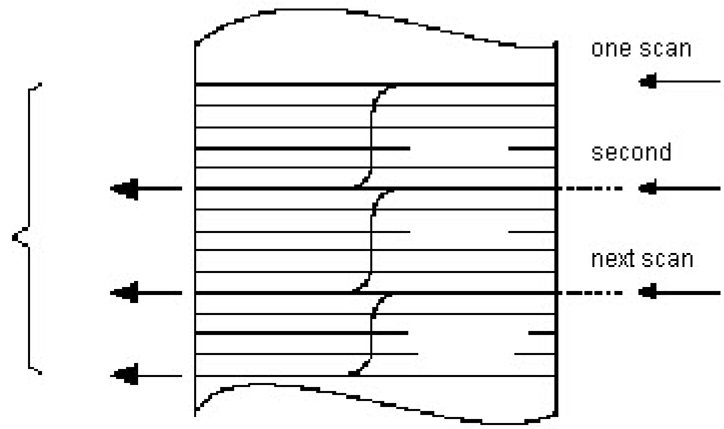
Numerical mode distributes the array operation over a number of scans. This mode is useful when working with non-time-critical data or large amounts of data. You enter the number of elements to operate on for each scan, which keeps scan time shorter.

Execution is triggered when the rung-condition-in goes from false to true. Once triggered, the instruction is executed each time it is scanned for the number of scans necessary to complete operating on the entire array. Once triggered, rung-condition-in can change repeatedly without interrupting execution of the instruction.



Avoid using the results of a file instruction operating in numerical mode until the .DN bit is set.

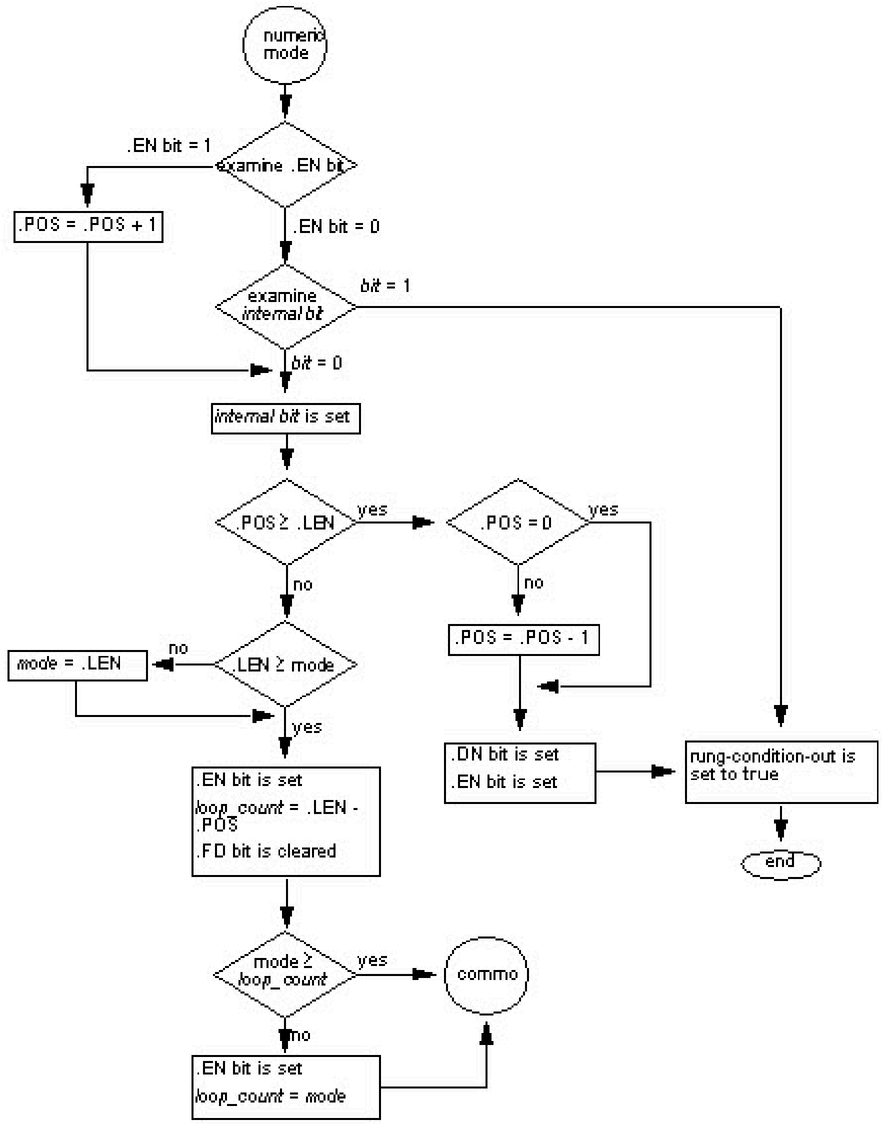
The following timing diagram shows the relationship between status bits and instruction operation. When the instruction execution is complete, the .DN bit is set.



If the rung-condition-in is true at completion, the .EN and .DN bit are set until the rung-condition-in goes false. When the rung-condition-in goes false, these bits are cleared and the .POS value is cleared.

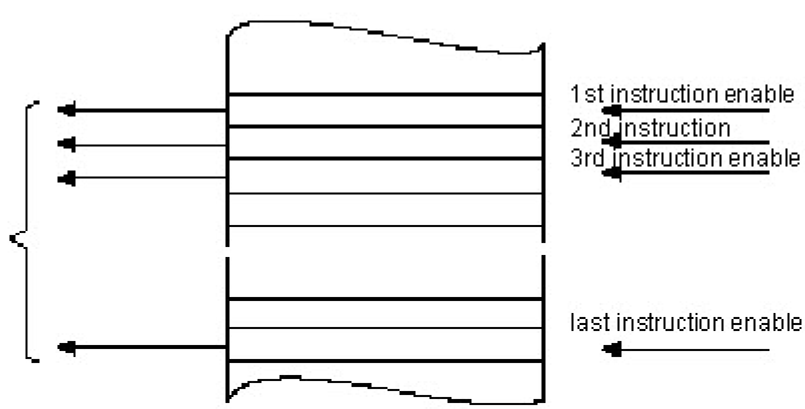
If the rung-condition-in is false at completion, the .EN bit is cleared immediately. One scan after the .EN bit is cleared, the .DN bit and the .POS value are cleared.

Numeric Mode Flow Chart-FSC

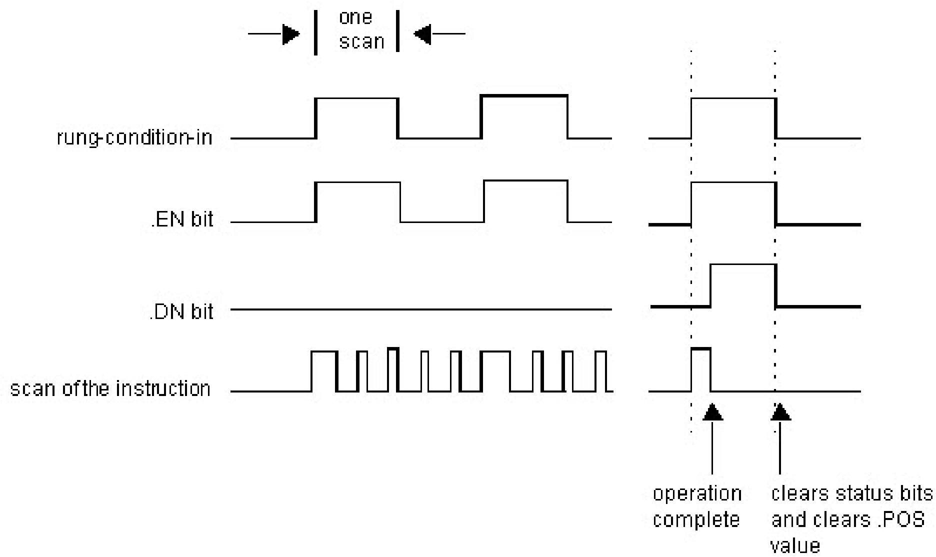


Incremental Mode

Incremental mode manipulates one element of the array each time the instruction's rung-condition-in goes from false to true.



The following timing diagram shows the relationship between status bits and instruction operation. Execution occurs only in a scan in which the rung-condition-in goes from false to true. Each time this occurs, only one element of the array is manipulated. If the rung-condition-in remains true for more than one scan, the instruction only executes during the first scan

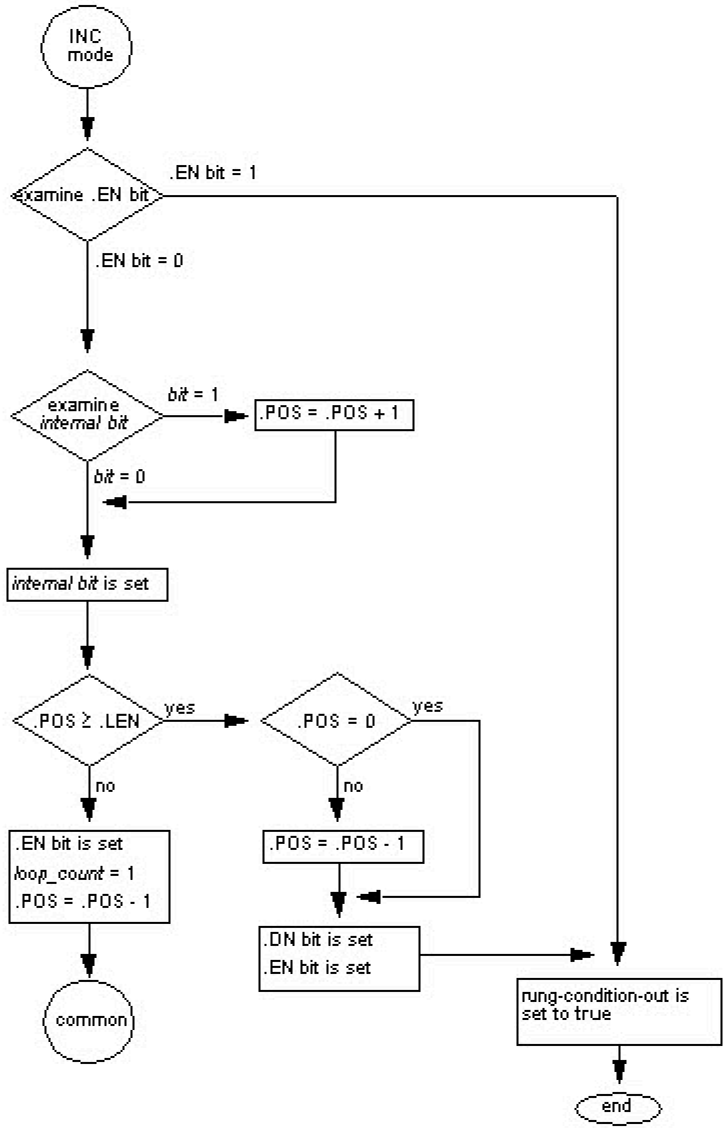


The .EN bit is set when rung-condition-in is true. The .DN bit is set when the last element in the array has been manipulated. When the last element has been manipulated and the rung-condition-in goes false, the .EN bit, the .DN bit, and the .POS value are cleared.

The difference between incremental mode and numerical mode at a rate of one element per scan is:

- Numerical mode with any number of elements per scan requires only one false-to-true transition of the rung-condition-in to start execution. The instruction continues to execute the specified number of elements each scan until completion regardless of the state of the rung-condition-in.
- Incremental mode requires the rung-condition-in to change from false to true to manipulate one element in the array.

Incremental Mode Flow Chart-FSC



Array Tag

When you enter an array tag, make sure to specify the first element of the array to manipulate. Do not use CONTROL.POS to identify the beginning element because the instruction modifies the .POS value as it operates, which could corrupt the result.

Standard Deviation

The standard deviation is calculated according to this formula:

$$\text{Standard Deviation} = \sqrt{\frac{\left(\sum_{i=1}^{N}\left[\langle X_{(start+i)} - AVE\rangle^{2}\right]\right)}{(N-1)}}$$

Where:

- start = dimension-to-vary subscript of the array operand
- xi = variable element in the array
- N = number of specified elements in the array
- AVE = $\dfrac{\left(\sum_{i=1}^{N} x_{(start+i)}\right)}{N}$

Array (File)/Shift Instructions

Use the array (file)/shift instructions to modify the location of data within arrays.

Available Instructions

Ladder Diagram

If you want to:	Use this instruction:
Load bits into, shift bits through, and unload bits from a bit array one bit at a time.	BSL on page 569 BSR on page 572
Load and unload values in the same order.	FFL on page 576 FFU on page 582
Load and unload values in reverse order.	LFL on page 588 LFU on page 594

You can mix data types, but loss of accuracy and rounding error might occur.

The bold data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

Bit Shift Left (BSL)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The BSL instruction shifts the specified bits within the Array one position left.

When enabled, the instruction unloads the uppermost bit of the specified bits to the .UL bit, shifts the remaining bits one position left, and loads Bit address into bit 0 of Array.

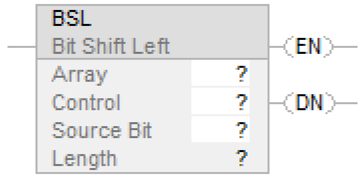
IMPORTANT: You must test and confirm that the instruction does not change data that you do not want it to change.

The BSL instruction operates on contiguous data memory. The data is constrained by the specified member.

In this transitional instruction, the relay ladder toggles the rung-condition-in from false to true for the instruction to execute.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

Operand	Type	Format	Description
Array	DINT ARRAY	tag	Array to modify specify the first element where to begin the shift
Control	CONTROL	tag	Control structure for the operation
Source Bit	BOOL	tag	Bit to shift into the vacated position.
Length	DINT	immediate	Number of bits in the array to shift

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the BSL instruction is enabled.
.DN	BOOL	The done bit is set to indicate that bits shifted one position to the left.
.UL	BOOL	The unload bit is the instruction's output. The .UL bit stores the status of the bit that was shifted out of the range of bits.
.ER	BOOL	The error bit is set when .LEN < 0.
.LEN	DINT	The length specifies the number of array bits to shift.

Affects Math Status Flags

No

Major/Minor Faults

A Major Fault Occurs If	Fault Type	Fault Code
The LEN exceeds the size of the array	4	20

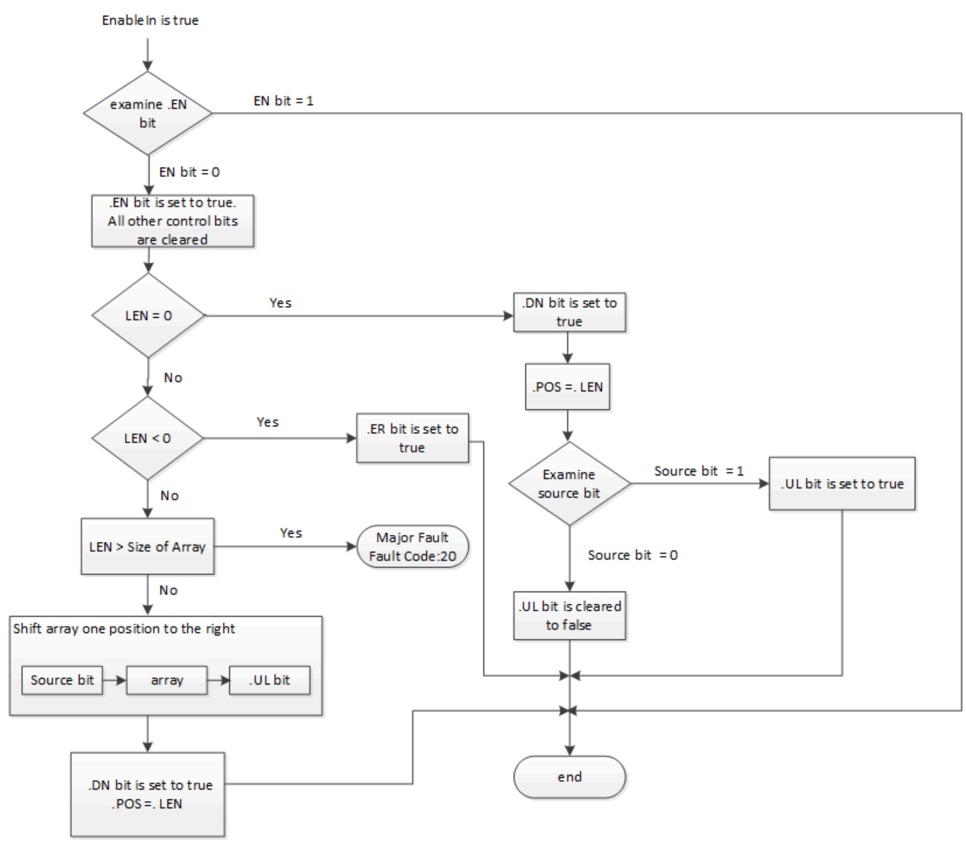
See Common Attributes for General Instructions on page 849 for operand related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN bit is cleared to false. The .DN bit is cleared to false. The .ER bit is cleared to false. The .POS value is cleared
Rung-condition-in is false	The .EN bit is cleared to false. The .DN bit is cleared to false. The .ER bit is cleared to false. The .POS value is cleared.
Rung-condition-in is true	See BSL Flow Chart (True).
Postscan	N/A

BSL Flow Chart (True)

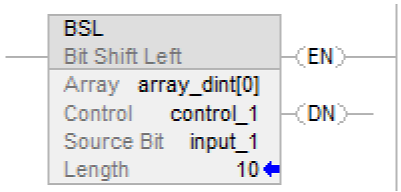


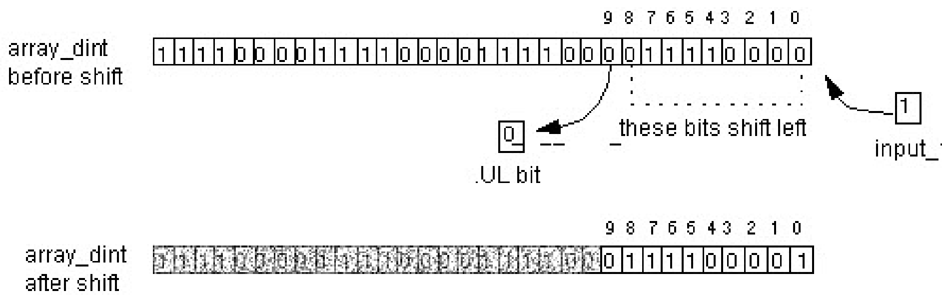
Examples

Example 1

When enabled, the BSL instruction starts at bit 0 in array_dint[0]. The instruction unloads array_dint[0].9 into the .UL bit, shifts the remaining bits, and loads input_1 into array_dint[0].0. The remaining bits (10-31) are invalid.

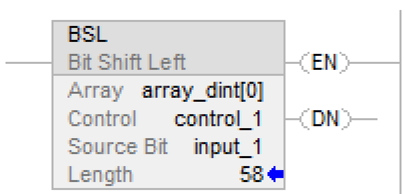
Ladder Diagram

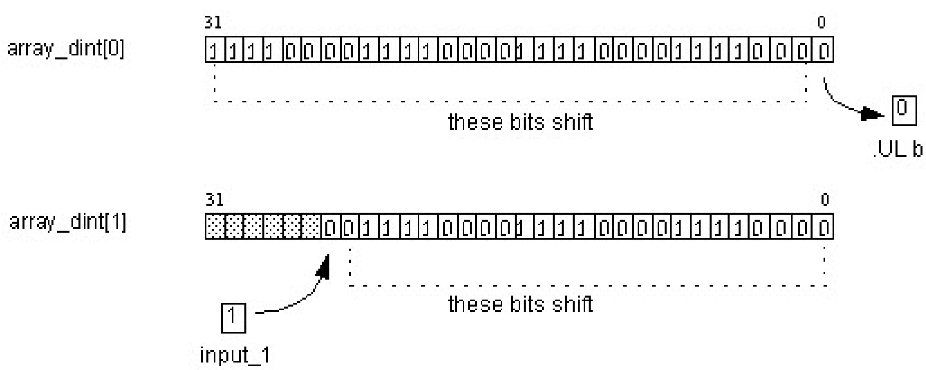




Example 2:

When enabled, the BSL instruction starts at bit 0 in array_dint[0]. The instruction unloads array_dint[1].25 into the .UL bit, shifts the remaining bits, and loads input_1 into array_dint[0].0. The remaining bits (31-26 in array_dint[1]) are invalid.





Bit Shift Right (BSR)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The BSR instruction shifts the specified bits within the Array one position right. When enabled, the instruction unloads the value at bit 0 of Array to the .UL bit, shifts the remaining bits one position right, and loads the bit from the Bit address.

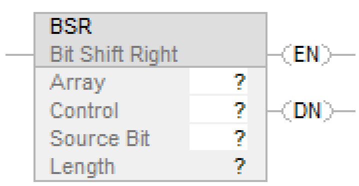
IMPORTANT: Test and confirm that the instruction changed the correct data. The BSR instruction operates on continuous memory. If an Array is a member array, the instruction may shift beyond the boundary of the array into other members following it. Be sure to carefully select a length that does cause this scenario to occur.

The BSR instruction operates on contiguous data memory.

If the instruction tries to read past the end of an array (the LEN is too big), the instruction sets the .ER bit and generates a major fault.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram

Operand	Data Type	Format	Description
Array	DINT ARRAY	tag	Array to modify specify the first element to be shifted.
Control	CONTROL	tag	Control structure for the operation
Source Bit	BOOL	tag	Bit to load into the vacated position.

Operand	Data Type	Format	Description
Length	DINT	immediate	Number of bits in the array to shift

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the BSR instruction is enabled.
.DN	BOOL	The done bit is set to indicate that bits shifted one position to the right.
.UL	BOOL	The unload bit is the instruction's output. The .UL bit stores the status of the bit that was shifted out of the range of bits.
.ER	BOOL	The error bit is set when .LEN < 0.
.LEN	DINT	The length specifies the number of array bits to shift.

Affects Math Status Flags

No

Major/Minor Faults

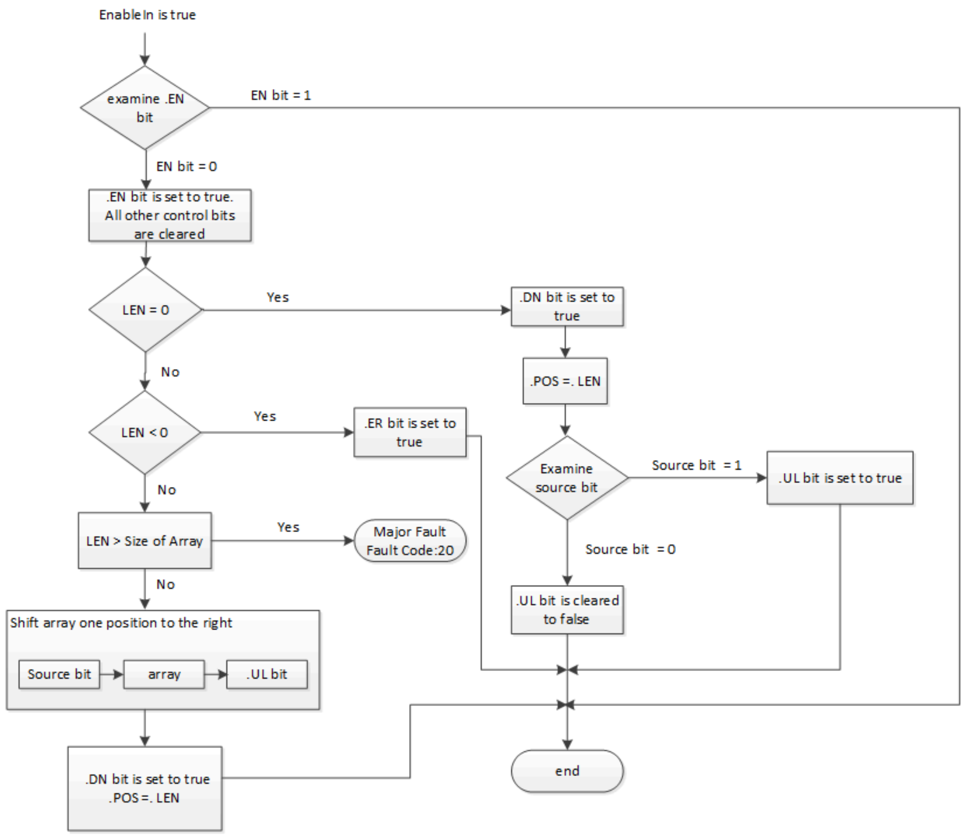
None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN bit is cleared to false. The .DN bit is cleared to false. The .ER bit is cleared to false. The .POS value is cleared.
Rung-condition-in is false	The .EN bit is cleared to false. The .DN bit is cleared to false. The .ER bit is cleared to false. The .POS value is cleared.
Rung-condition-in is true	See the following BSR Flow Chart (True)
Postscan	N/A

BSR Flow Chart (True)

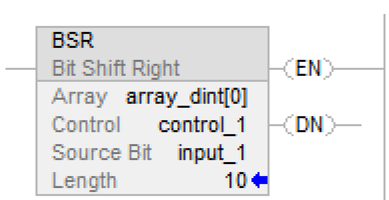


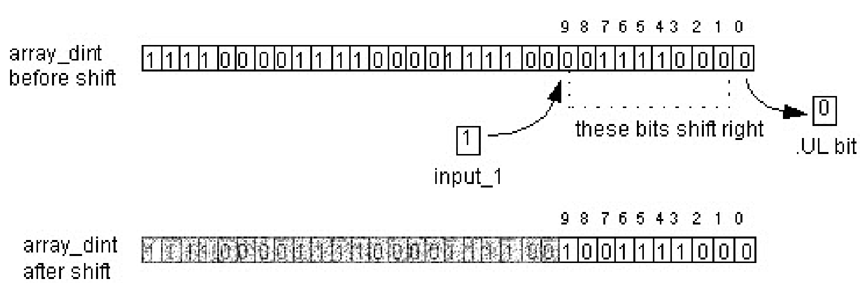
Examples

Example 1

When enabled, the BSR instruction copies array_dint[0].0 to the .UL bit, shifts 0-9 to the right, and loads the input_1 into array_dint[0].9. The remaining bits (10-31) are invalid, which indicates the bits may not be modified.

Ladder Diagram

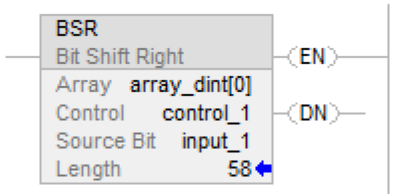


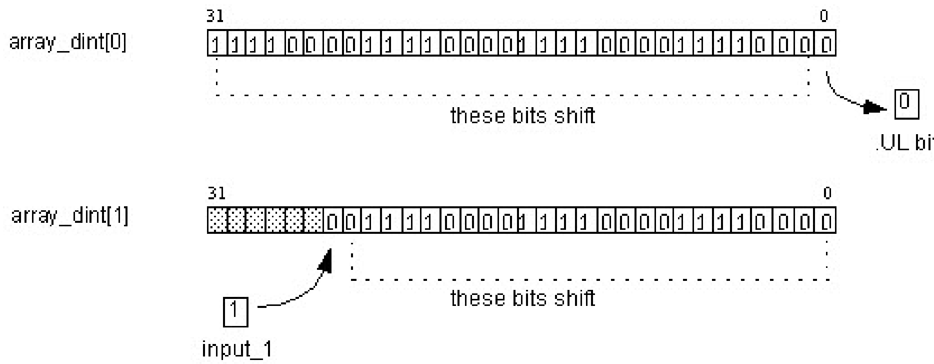


Example 2

When enabled, the BSR instruction copies array_dint[0].0 to the .UL bit, shifts 0-9 to the right, and loads the input_1 into array_dint[1].25.. The remaining bits (31-26 in dint_array[1]) are invalid, which indicates that the bits may not be modified. Note how array_dint[1].0 shifts across words into array_dint[0].31.

Ladder Diagram





FIFO Load (FFL)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The FFL instruction copies the Source value to the FIFO.

Use the FFL instruction with the FFU instruction to store and retrieve data in a first-in/first-out order. When used in pairs, the FFL and FFU instructions establish an asynchronous shift register.

Typically, the Source and the FIFO are the same data type.

When enabled, the FFL instruction loads the Source value into the position in the FIFO identified by the .POS value. The instruction loads one value each time the instruction is enabled, until the FIFO is full.

IMPORTANT: You must test and confirm that the instruction does not change data that you don't want it to change.

The FFL instruction operates on contiguous memory.

The data is constrained by the specified member.

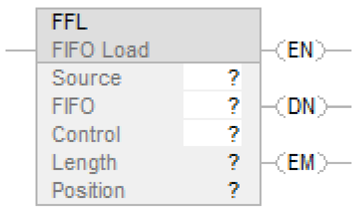
If the instruction tries to read past the end of an array, the instruction generates a major fault.

Typically, the Source and the FIFO are the same data type. If Source and FIFO data types mismatch, the instruction converts the Source value to the data type of the FIFO tag.

A smaller integer converts to a larger integer by sign-extension.

Available Languages

Ladder Diagram



Operands

Conversion only occurs if the type of the source operand does not match the type of the FIFO.

Ladder Diagram

Operand	Type	Format	Description
Source	SINT INT DINT REAL String type structure	immediate tag	Data to be stored in the FIFO
FIFO	SINT INT DINT REAL String type structure	array tag	FIFO to modify Specify the first element of the FIFO
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

Operand	Type	Format	Description
			initial value is typically 0

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the FFL instruction is enabled.
.DN	BOOL	The done bit is set to indicate that the FIFO is full. The .DN bit inhibits loading the FIFO until .POS < .LEN.
.EM	BOOL	The empty bit indicates the FIFO is empty. If .LEN is < or = to 0 or .POS < 0, the .EM bit and .DN bits are set.
.LEN	DINT	The length word specifies the maximum number of elements in the FIFO.
.POS	DINT	The position word identifies the location in the FIFO where the instruction loads the next value.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault Type	Fault Code
The (starting element + .POS) is past the end of FIFO array	4	20

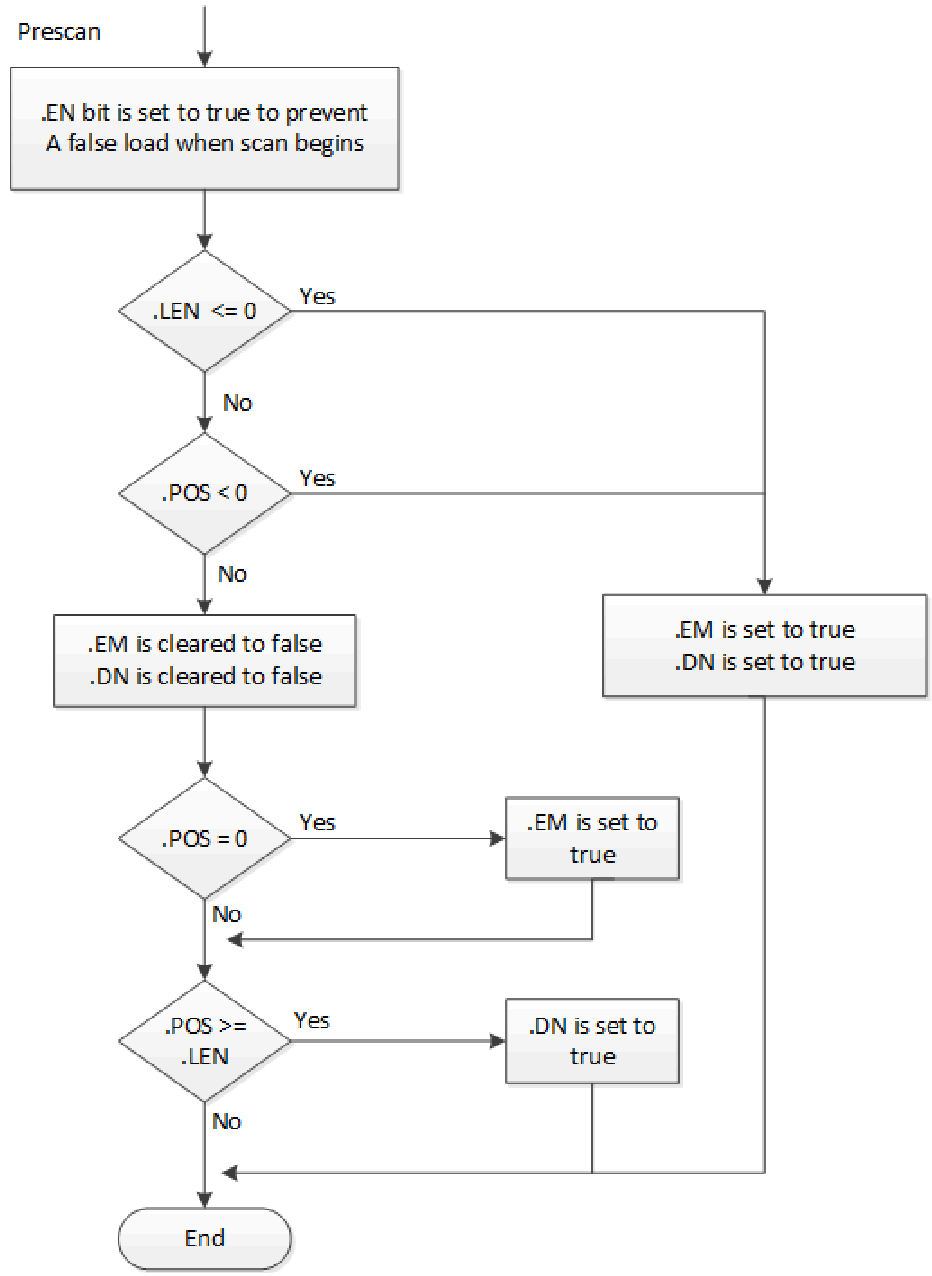
See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

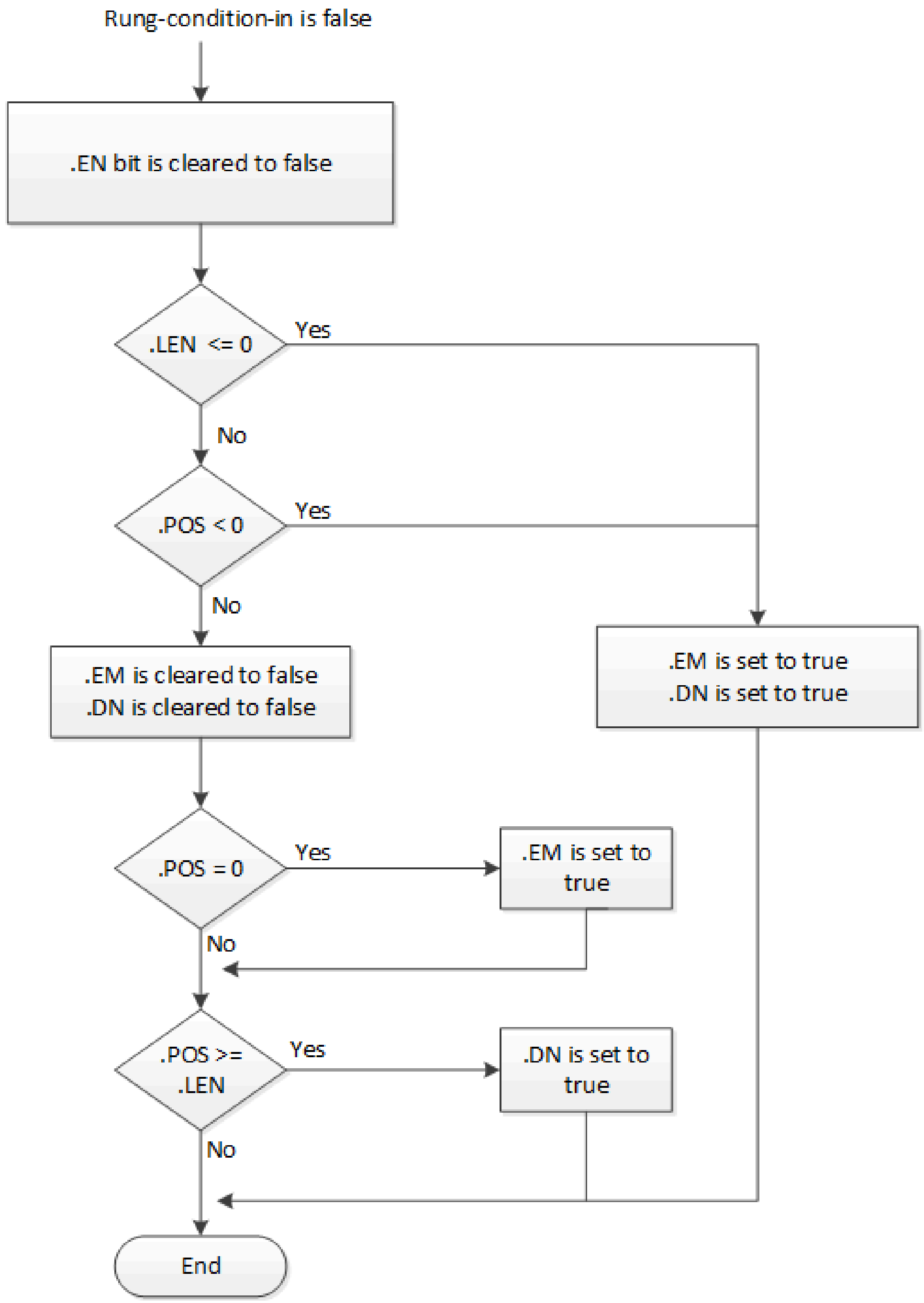
Ladder Diagram

Condition/State	Action Taken
Prescan	See the FFL Flow Chart (Prescan).
Rung-condition-in is false	See FFL Flow Chart (False)
Rung-condition-in is true	See FFL Flow Chart (True)
Postscan	N/A

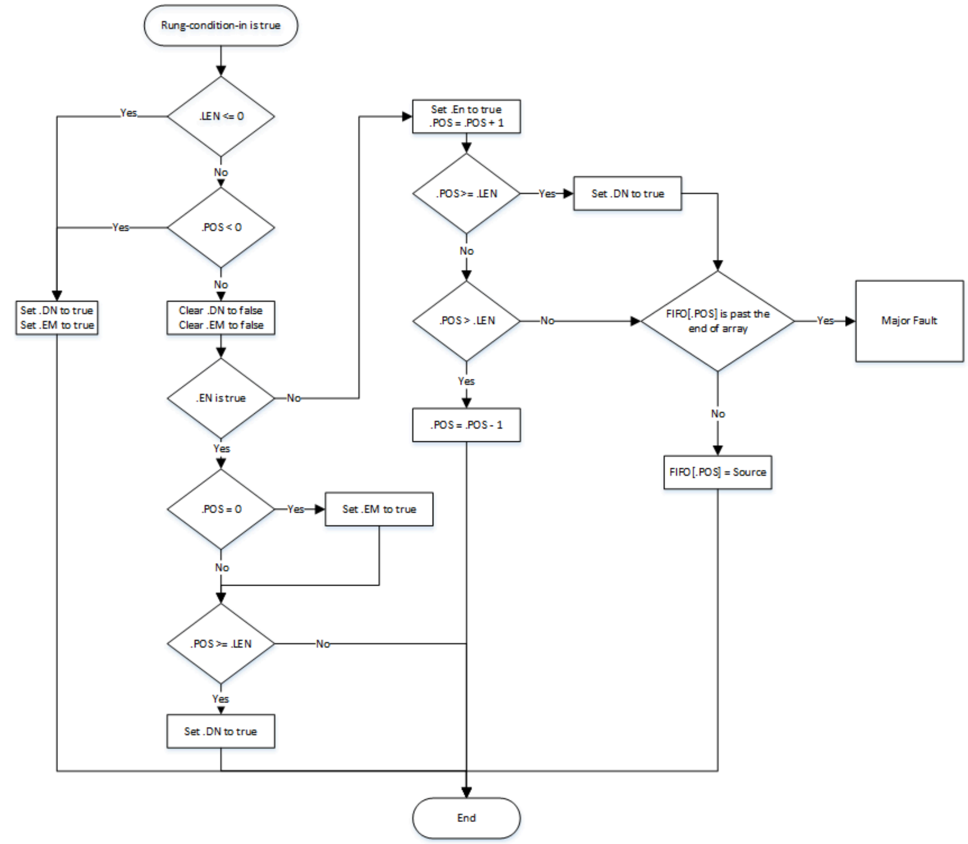
FFL Flow Chart (Prescan)



FFL Flow Chart (False)



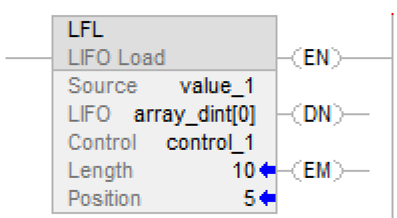
FFL Flow Chart (True)

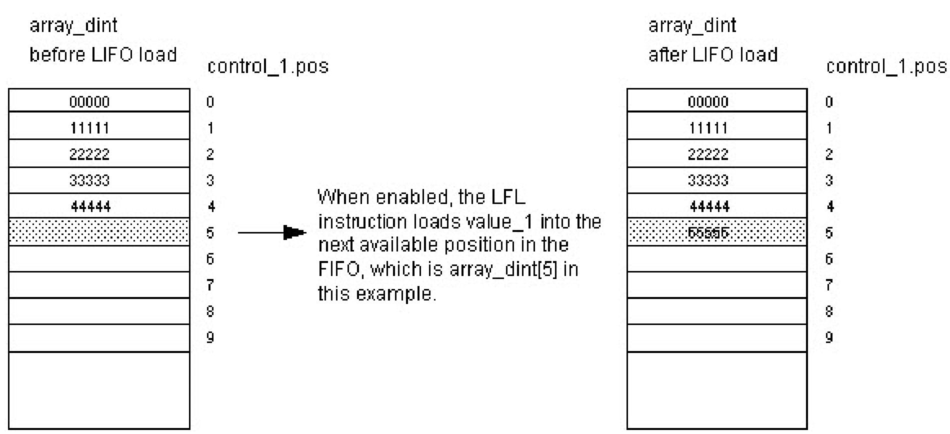


Examples

Example 1

Ladder Diagram

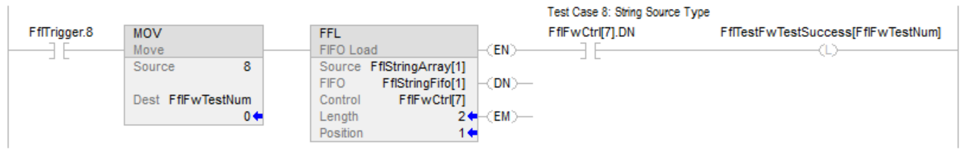




Example 2

Source array is STRING array or Structure array.

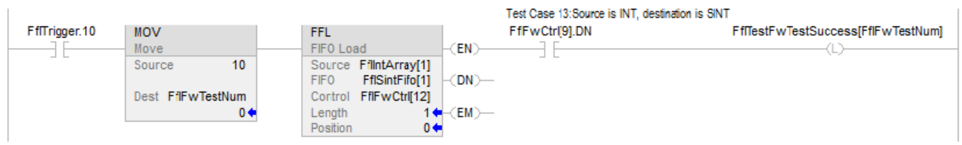
Ladder Diagram



Example 3

Data type of source mismatch data type of FIFO array.

Ladder Diagram



FIFO Unload (FFU)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

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.

Use the FFU instruction with the FFL instruction to store and retrieve data in a first-in/first-out order.

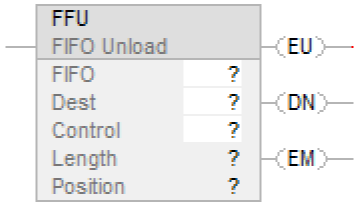
When enabled, the FFU instruction unloads data from the first element of the FIFO and places that value in the Destination. The instruction unloads one value each time the instruction is enabled, until the FIFO is empty. If the FIFO is empty, the FFU returns 0 to the Destination.

Typically, the destination and the FIFO are the same data type. If the types differ, the instruction converts the unloaded value to the type of the destination tag.

A smaller integer converts to a larger integer by sign-extension.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction.

Ladder Diagram

Operand	Type	Format	Description
FIFO	SINT INT DINT REAL String type structure	array tag	FIFO to modify Specify the first element of the FIFO Do Not use CONTROL.POS in the subscript
Destination	SINT INT DINT REAL String type structure	tag	Value unloaded from 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 loads data initial value is typically 0

CONTROL Structure

Mnemonic	Data Type	Description
.EU	BOOL	The enable unload bit indicates the FFU instruction is enabled. The .EU bit is set to prevent a false unload when the prescan begins.

Mnemonic	Data Type	Description
.DN	BOOL	The done bit is set to indicate that the FIFO is full (.POS = .LEN).
.EM	BOOL	The empty bit indicates the FIFO is empty. If .LEN is , or = to 0 or .POS < 0, the .EM bit and .DN bits are set.
.LEN	DINT	The length specifies the maximum number of elements in the FIFO.
.POS	DINT	The position identifies the end of the data that has been loaded into the FIFO.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault Type	Fault Code
The specified Length is past the end of FIFO array	4	20

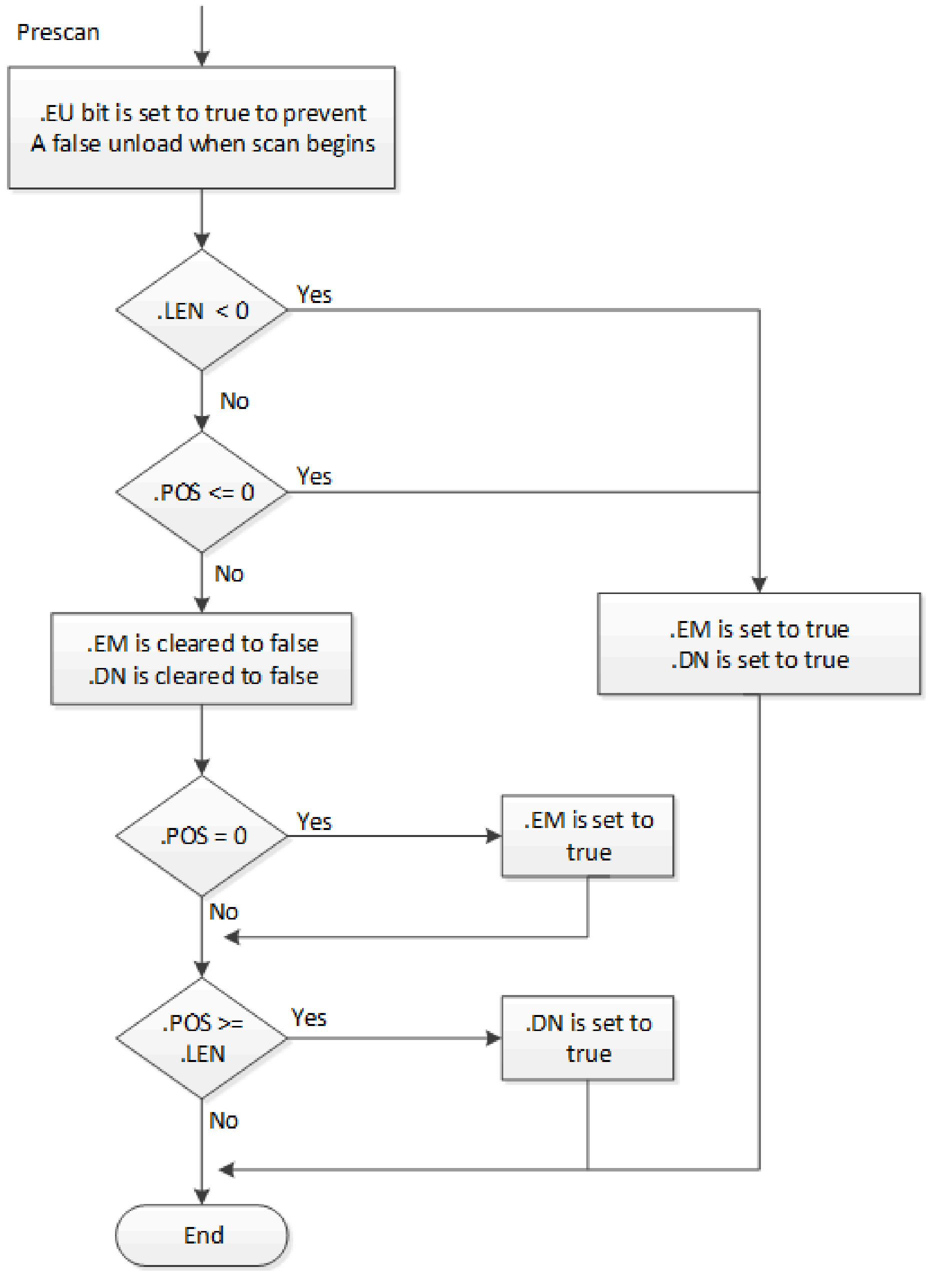
See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

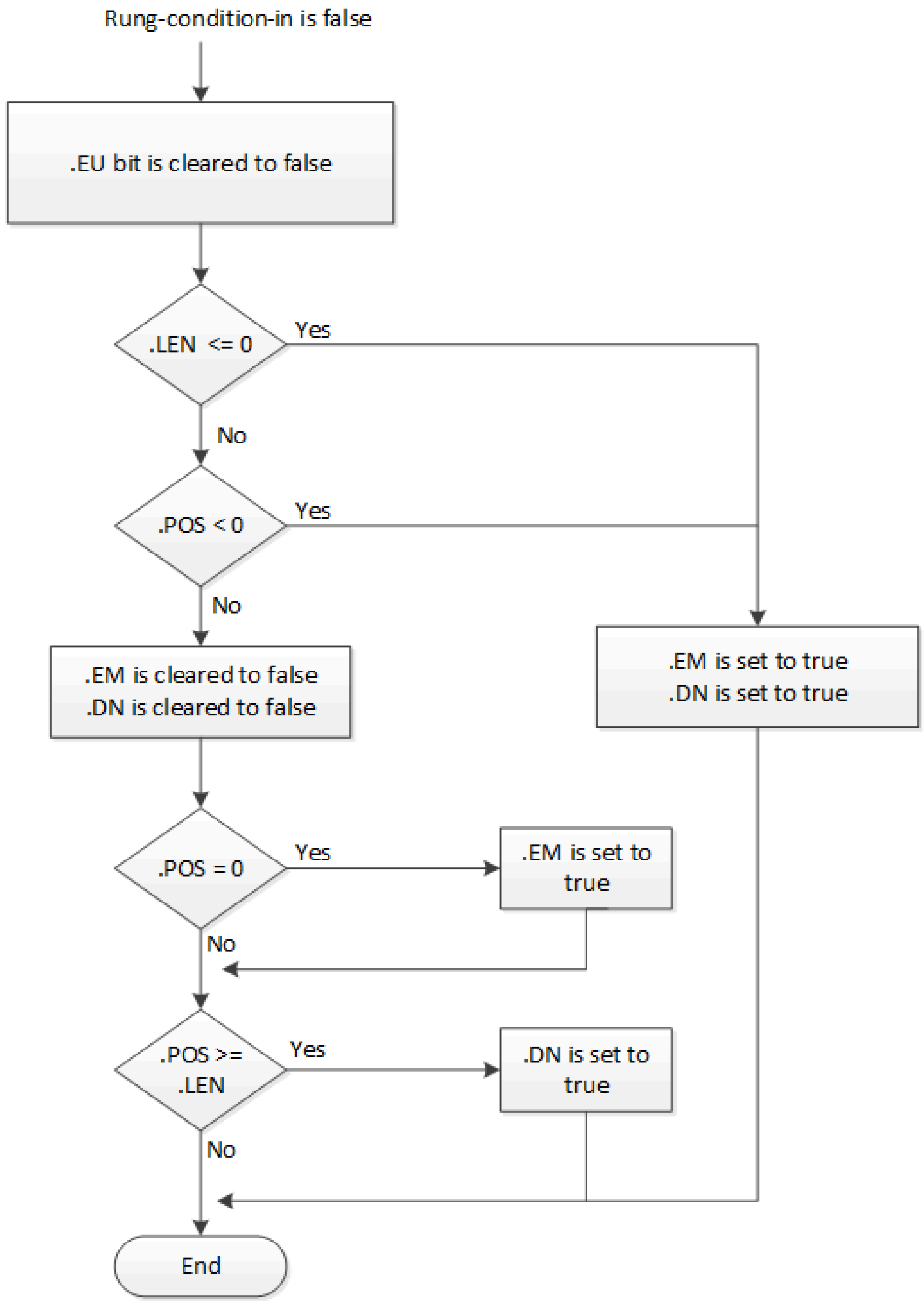
Ladder Diagram

Condition / State	Action Taken
Prescan	See FFU Flow Chart (Prescan).
Rung-condition-in is false	See FFL Flow Chart (False).
Rung-condition-in is true	See FFU Flow Chart (True)
Postscan	N/A

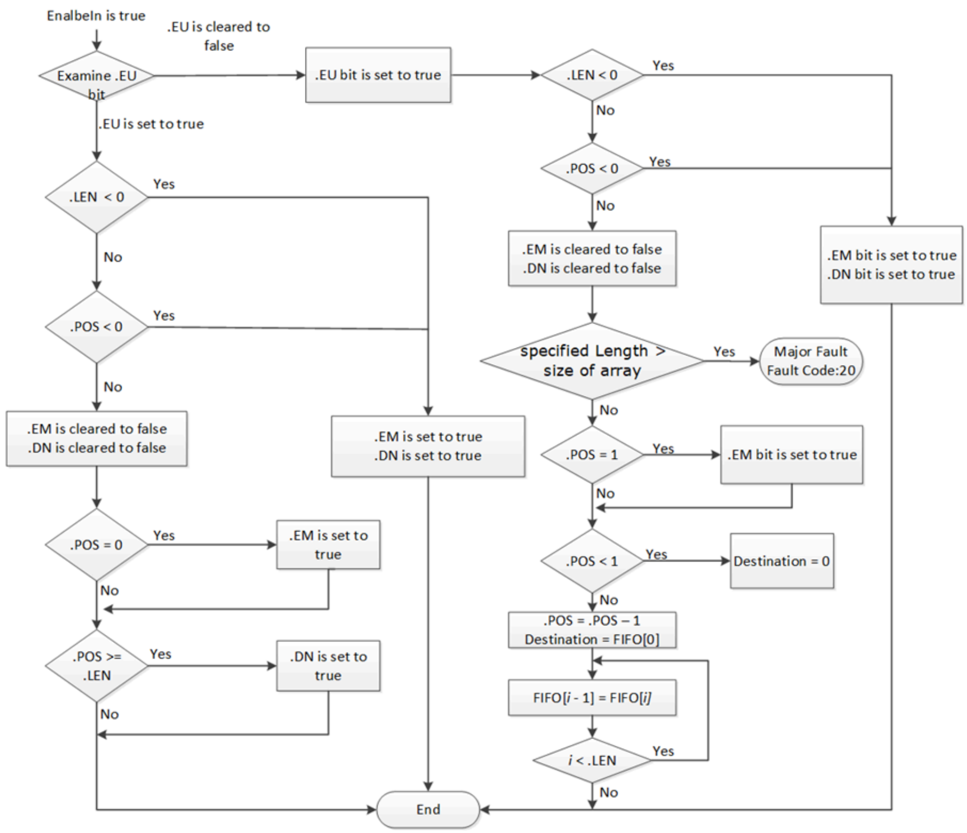
FFU Flow Chart (Prescan)



FFL Flow Chart (False)



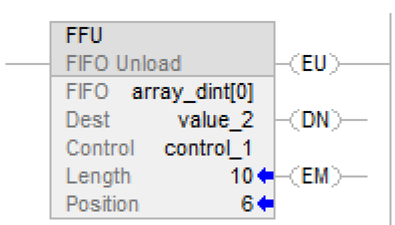
FFU Flow Chart (True)

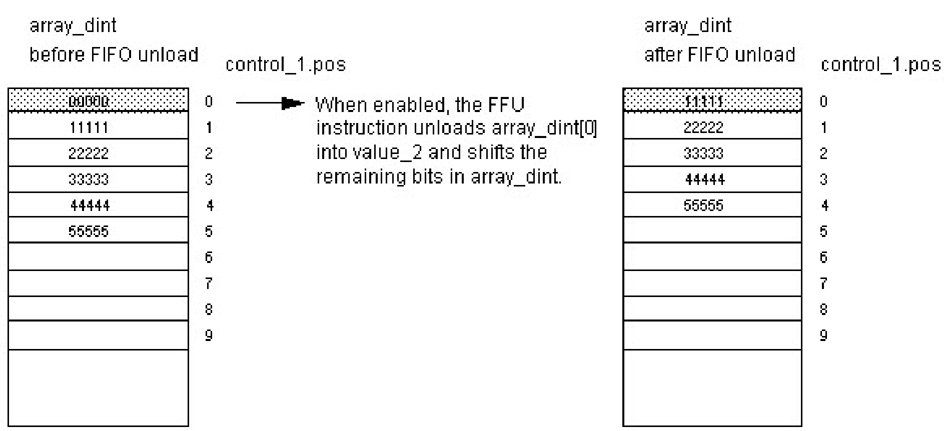


Examples

Example 1

Ladder Diagram

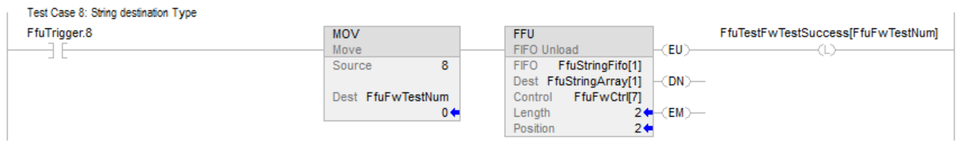




Example 2

Destination array is STRING array or Structure array

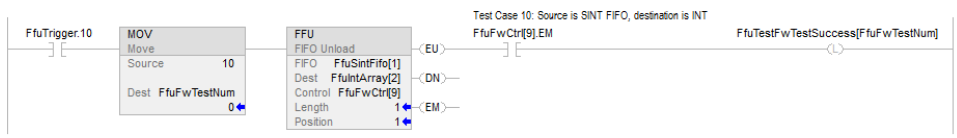
Ladder Diagram



Example 3

Data type of FIFO source array mismatch data type of destination array

Ladder Diagram



LIFO Load (LFL)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The LFL instruction copies the Source value to the LIFO.

Use the LFL instruction with the LFU instruction to store and retrieve data in a last-in/first-out order. When used in pairs, the LFL and LFU instructions establish an asynchronous shift register.

Typically, the Source and the LIFO are the same data type.

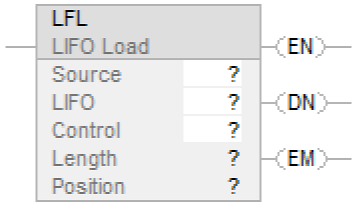
When enabled, the LFL instruction loads the Source value into the position in the LIFO identified by the .POS value. The instruction loads one value each time the instruction is enabled, until the LIFO is full.

IMPORTANT: You must test and confirm that the instruction does not change data that you don't want it to change.

The LFL instruction operates on contiguous data memory.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction.

Ladder Diagram

Operand	Type	Format	Description
Source	SINT INT DINT REAL String type structure	immediate tag	Data to be stored in the LIFO.
LIFO	SINT INT DINT REAL String type structure	array tag	LIFO to modify Specify the first element of the LIFO
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

CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the LFL instruction is enabled.

Mnemonic	Data Type	Description
.DN	BOOL	The done bit is set to indicate that the LIFO is full (.POS = .LEN). The .DN bit inhibits loading the LIFO until .POS < .LEN.
.EM	BOOL	The empty bit indicates the LIFO is empty. If .LEN < or = to 0 or .POS < 0, the .EM bit and .DN bits are set.
.LEN	DINT	The length specifies the maximum number of elements the LIFO can hold at one time.
.POS	DINT	The position identifies the location in the LIFO where the instruction will load the next value.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault Type	Fault Code
If (starting element + .POS) is past the end of LIFO array	4	20

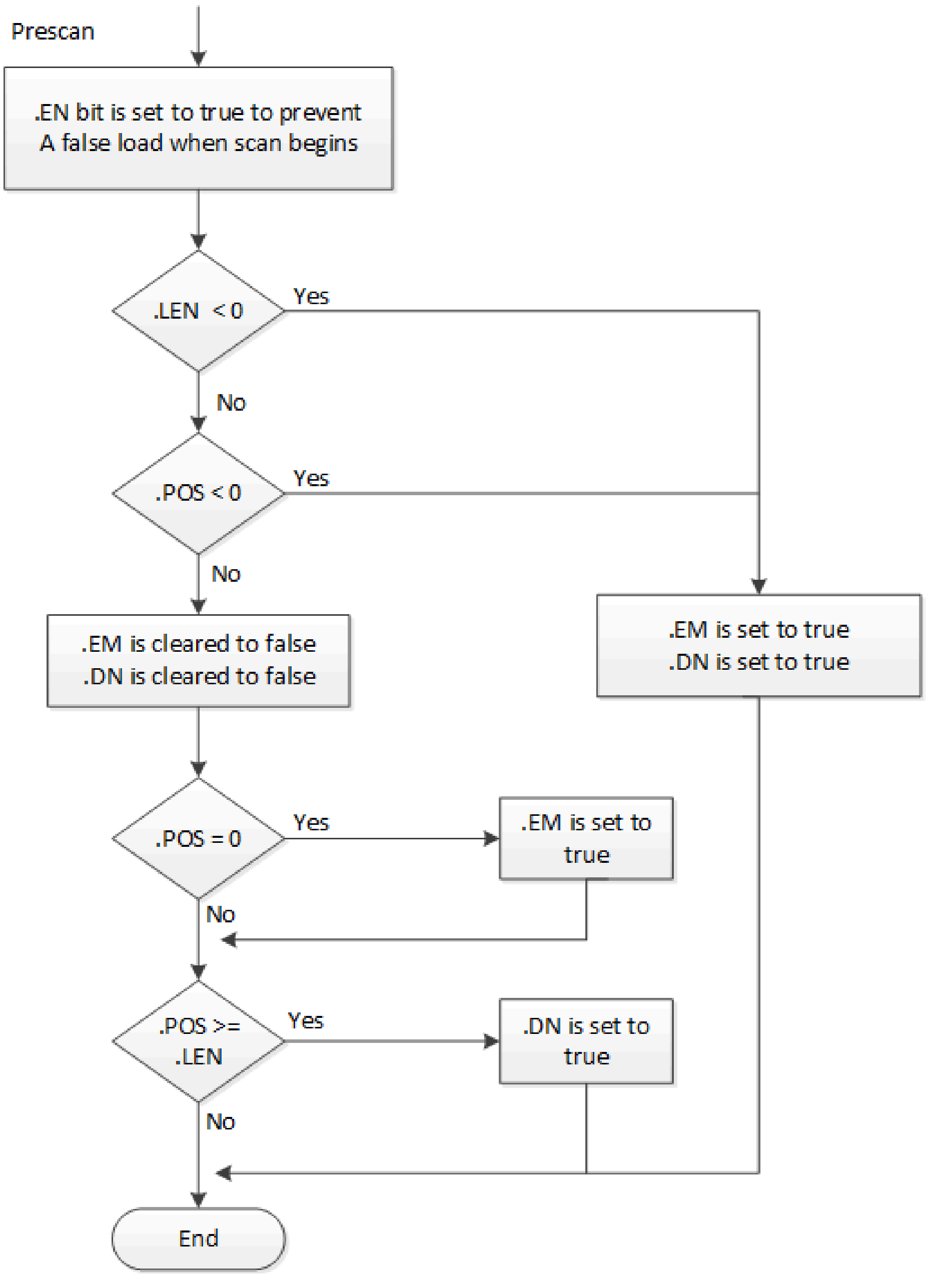
See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

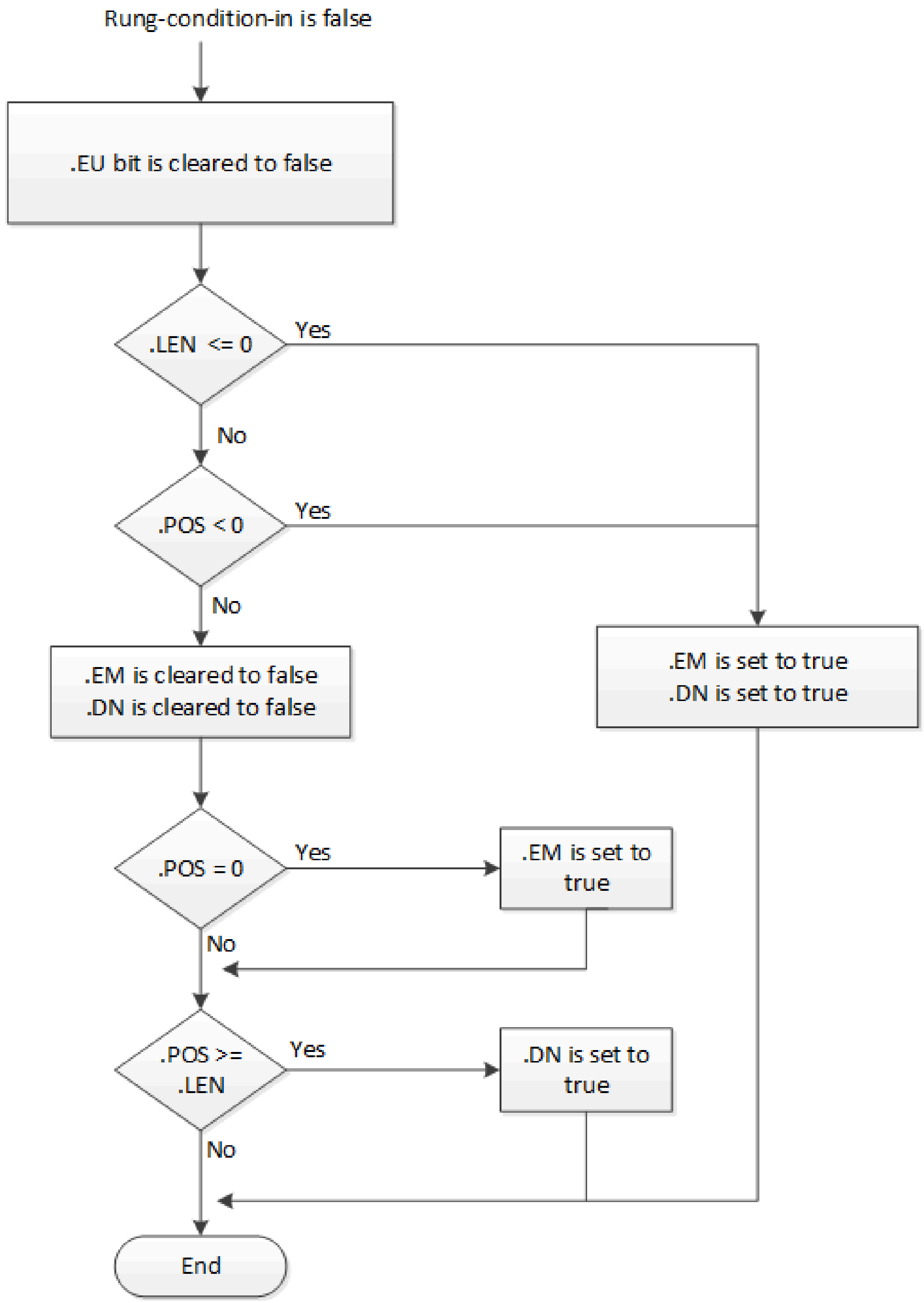
Ladder Diagram

Condition/State	Action Taken
Prescan	See LFL Flow Chart (Prescan)
Rung-condition-in is false	See LFL Flow Chart (False)
Rung-condition-in is true	See LFL Flow Chart (True)
Postscan	N/A.

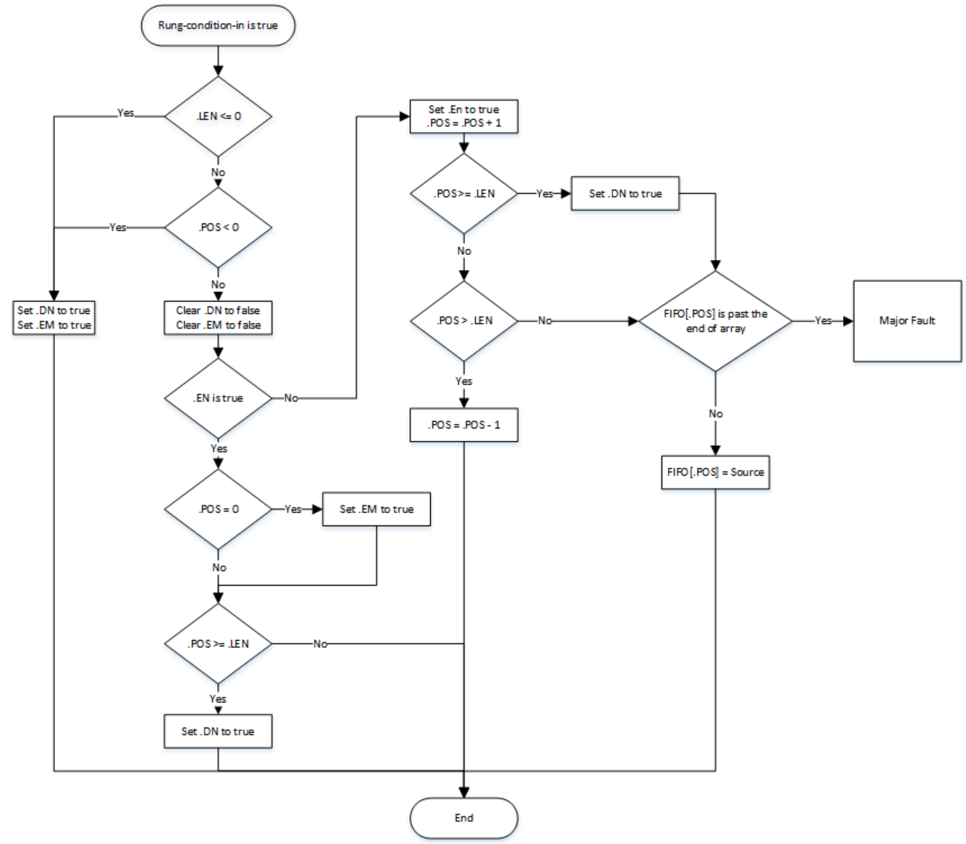
LFL Flow Chart (Prescan)



LFL Flow Chart (False)



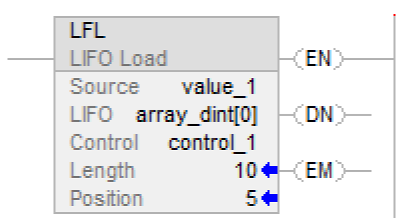
LFL Flow Chart (True)

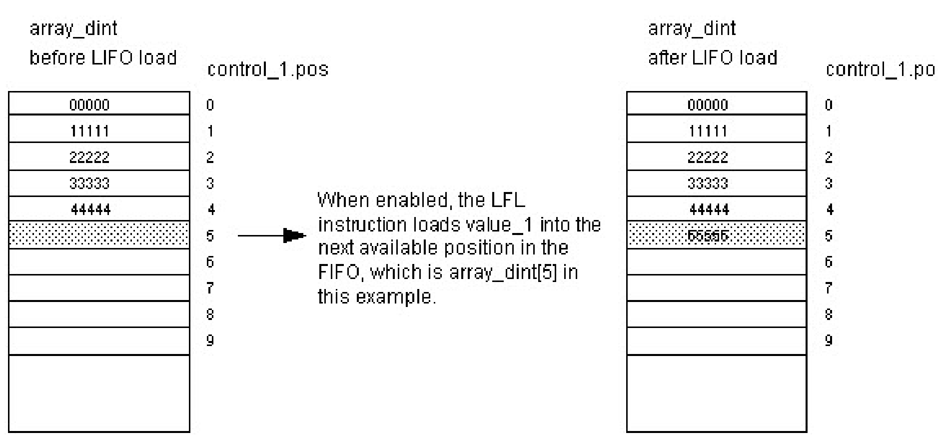


Examples

Example 1

Ladder Diagram

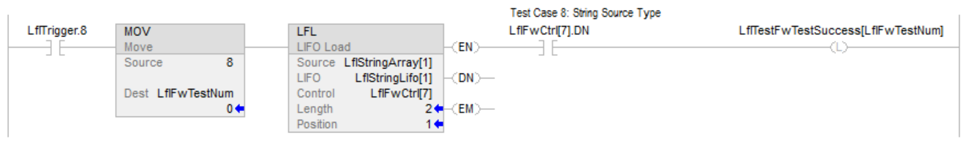




Example 2

Source array is STRING array or Structure array.

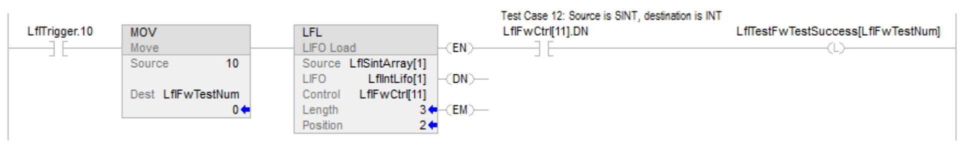
Ladder Diagram



Example 3

Data type of source mismatch data type of LIFO array.

Ladder Diagram



LIFO Unload (LFU)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The LFU instruction unloads the value at .POS of the LIFO and stores 0 in that location.

Use the LFU instruction with the LFL instruction to store and retrieve data in a last-in/first-out order.

When enabled, the LFU instruction unloads the value at .POS of the LIFO and places that value in the Destination. The instruction unloads one value and replaces it with 0 each time the instruction is enabled, until the LIFO is empty. If the LIFO is empty, the LFU returns 0 to the Destination.

IMPORTANT: You must test and confirm that the instruction does not change data that you don't want it to change.

The LFU instruction operates on contiguous memory. The scope of the instruction is constrained by the base tag. The LFL instruction will not write data outside of the base tag but can cross member boundaries. If you specify an array that is a member of a structure, and the length exceeds the size of that array you must test and confirm that the LFL instruction does not change data you do not want changed.

The data is constrained by the specified member.

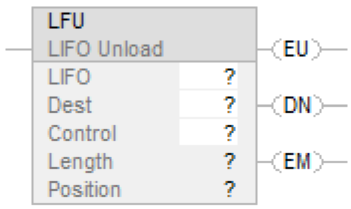
If the instruction tries to read past the end of an array, the instruction sets the .ER bit and generates a major fault.

Typically, the Source and the LIFO are the same data type. If Source and LIFO data types mismatch, the instruction converts the Source value to the data type of the FIFO tag.

A smaller integer converts to a larger integer by sign-extension.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction.

Ladder Diagram

Operand	Type	Format	Description
LIFO	SINT INT DINT REAL String type structure	array tag	LIFO to modify Specify the first element of the LIFO Not use CONTROL.POS in the subscript
Destination	SINT INT DINT REAL String type structure	tag	Value unloaded from 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

Operand	Type	Format	Description
			Initial value is typically 0

CONTROL Structure

Mnemonic	Data Type	Description
.EU	BOOL	The enable bit indicates the LFU instruction is enabled.
.DN	BOOL	The done bit is set to indicate that the LIFO is full (.POS = .LEN).
.EM	BOOL	The empty bit indicates the LIFO is empty. If .LEN < or = to 0 or .POS < 0, both the .EM bit and .DN bit are set.
.LEN	DINT	The length specifies the maximum number of elements the LIFO can hold at one time.
.POS	DINT	The position identifies the end of the data that has been loaded into the LIFO.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault Type	Fault Code
If the specified Length is past the end of LIFO array	4	20

See Common Attributes for General Instructions on page 849 for operand-related faults.

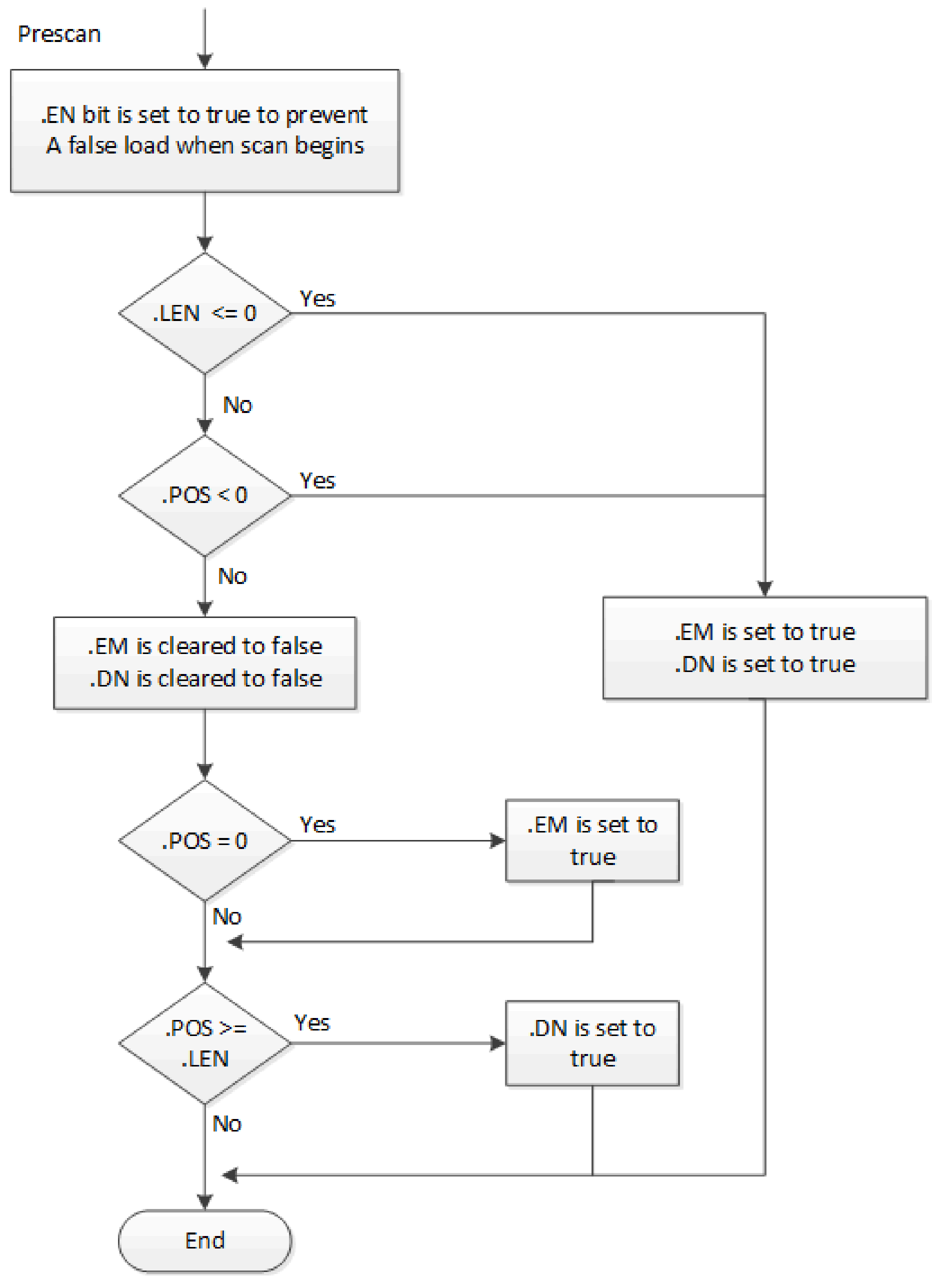
Execution

All conditions occur only during Normal Scan mode

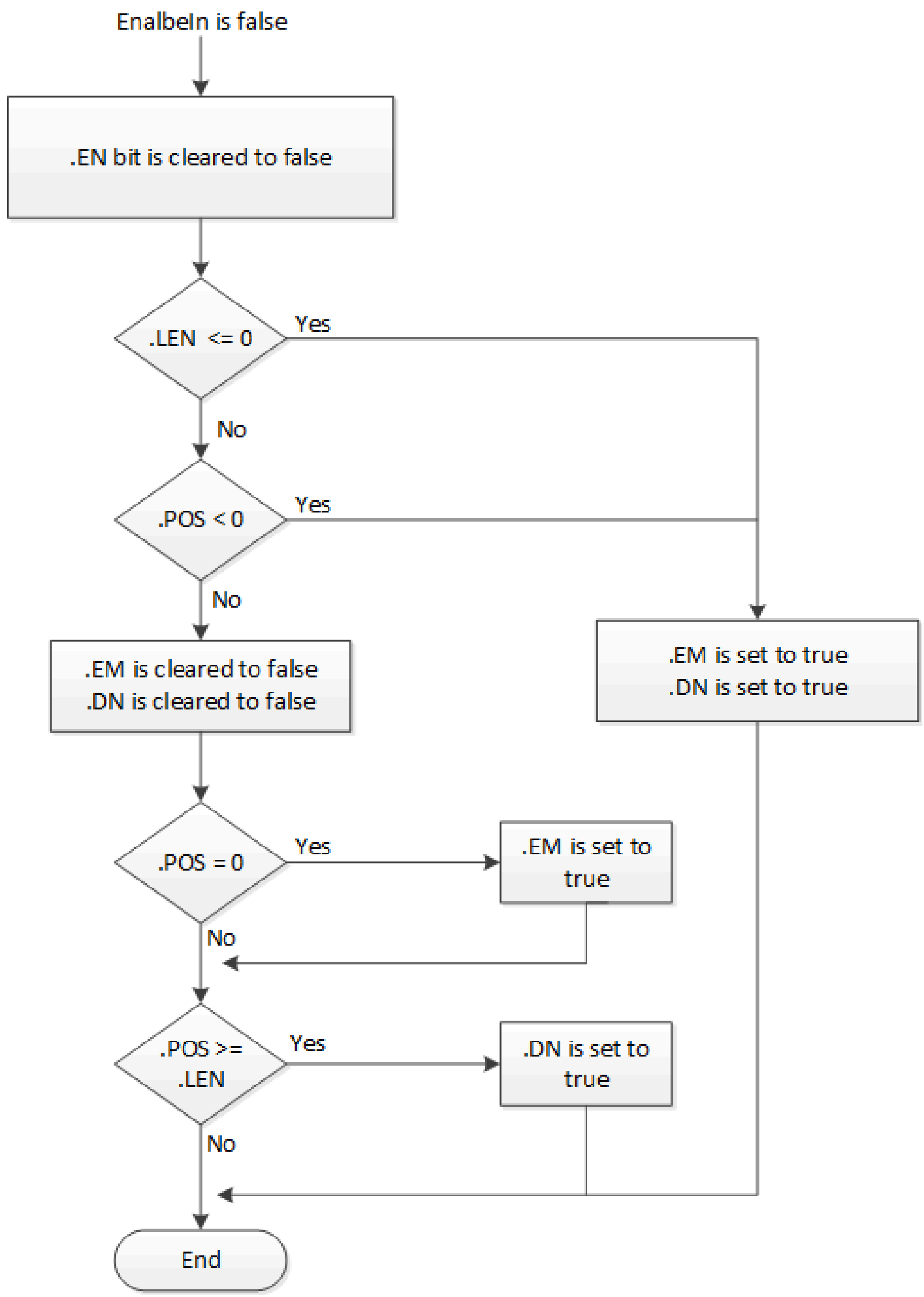
Ladder Diagram

Condition/State	Action Taken
Prescan	See LFU Flow Chart (Prescan)
Rung-condition-in is false	See LFU Flow Chart (False)
Rung-condition-in is true	See LFU Flow Chart (True)
Postscan	N/A

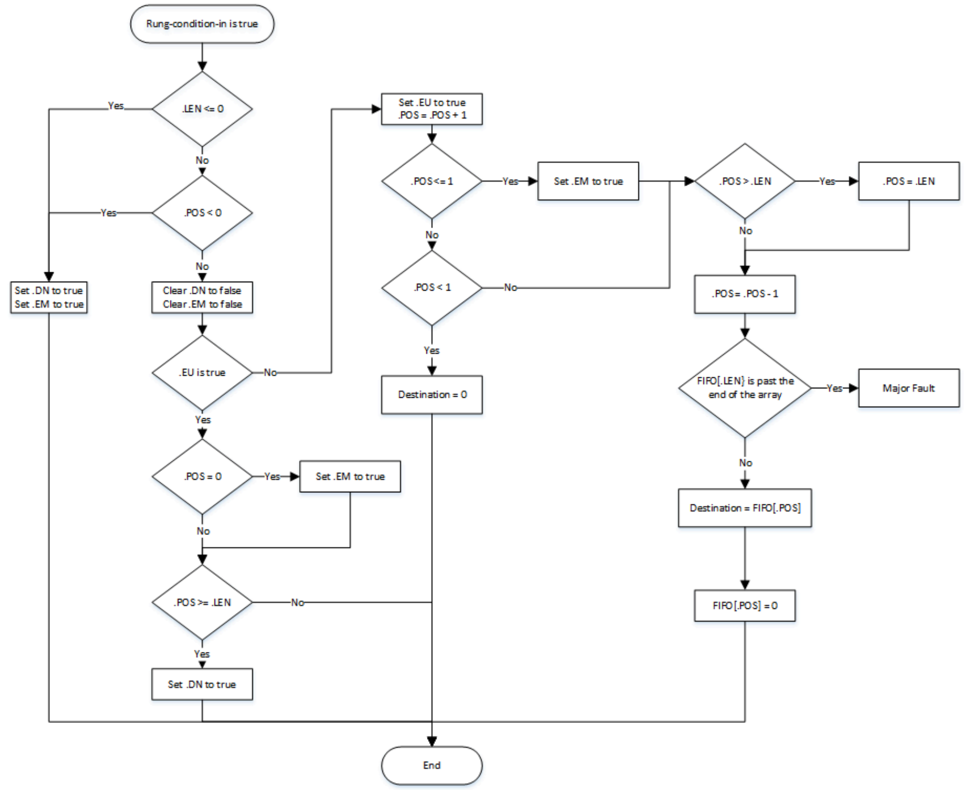
LFU Flow Chart (Prescan)



LFU Flow Chart (False)



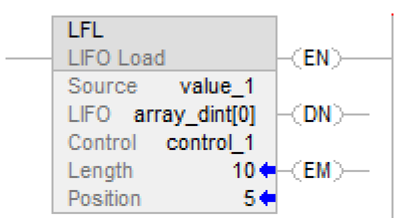
LFU Flow Chart (True)

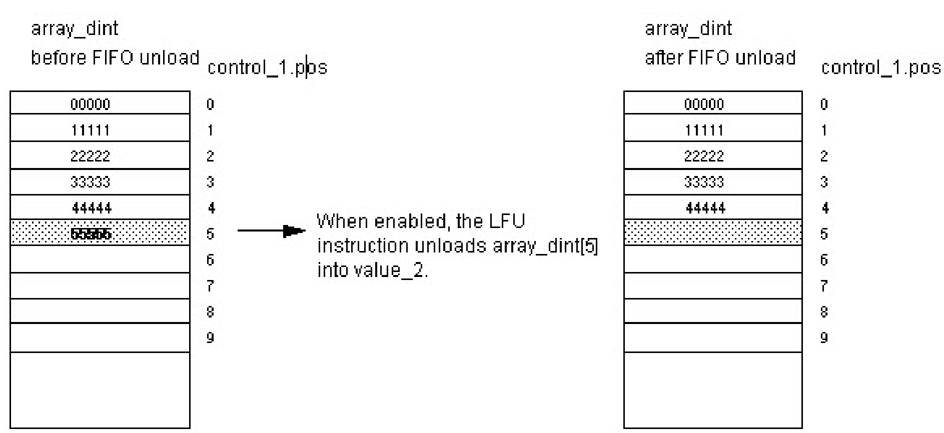


Examples

Example 1

Ladder Diagram

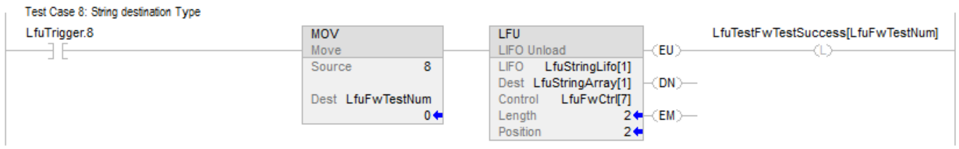




Example 2

Destination array is STRING array or Structure array

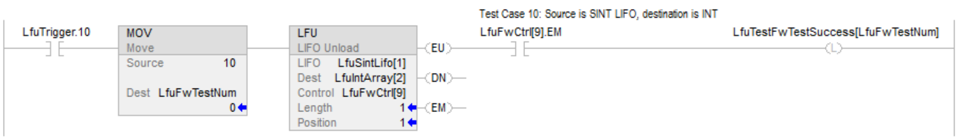
Ladder Diagram



Example 3

Data type of LIFO source array mismatch data type of destination array

Ladder Diagram



Sequencer Instructions

Sequencer instructions monitor consistent and repeatable operations.

Available Instructions

Ladder Diagram

If you want to	Use this instruction
Detect when a step is complete.	SQI on page 601
Set output conditions for the next step.	SQO on page 608
Load reference conditions into sequencer arrays	SQL on page 604

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

Sequencer Input (SQI)

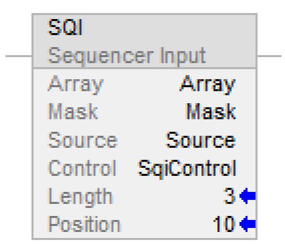
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SQI instruction detects when a step is complete in a sequence pair of SQO/SQI instructions.

When true, the SQI instruction passes the Source and current Array element through the Mask. The results of these masking operations are compared and if they are equal, rung-condition-out is set to true, otherwise rung-condition-out is cleared to false. Typically use the same CONTROL structure as the SQO and SQL instructions.

Available Languages

Ladder Diagram



Operands

The data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Operand	Type	Format	Description
Array	DINT	array tag	Sequencer array Specify the first element of the sequencer array

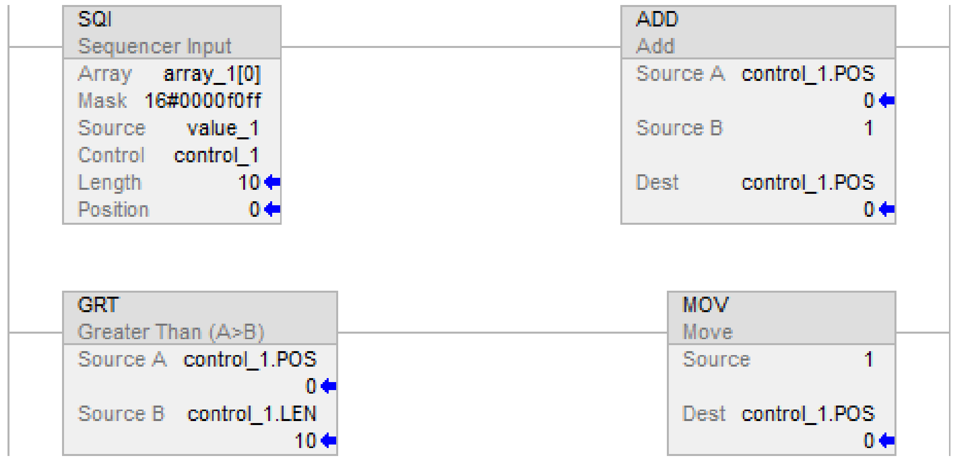
Operand	Type	Format	Description
			do not use CONTROL.POS in the subscript
Mask	SINT INT DINT	tag immediate	This operand is used to determine which bits to block (0) or pass (1) when applied to the Source and the Array element referenced by .POS. INT and SINT types are zero extended to the size of a DINT type.
Source	SINT INT DINT	tag immediate	The input data used to compare with an array element referenced by .POS..
Control	CONTROL	tag	Control structure for the operation The same control tag should be used in the SQO and SQL instructions
Length	DINT	immediate	This represents the CONTROL structure .LEN.
Position	DINT	immediate	This represents the CONTROL structure .POS.

CONTROL Structure

Mnemonic	Data Type	Description
.ER (Error)	BOOL	The instruction encountered an error.
.LEN (Length)	DINT	The length specifies the number of sequencer steps in the sequencer array
.POS (Position)	DINT	The position identifies the Array element that the instruction is currently comparing with the Source. The initial value is typically 0

Using SQI without SQO

When the SQI instruction determines a step is complete, the ADD instruction increments the sequencer array. The GRT determines whether another value is available to check in the sequencer array. The MOV instruction resets the position value after completely stepping through the sequencer array one time.



Affects Math Status Flags

No

Major/Minor Faults

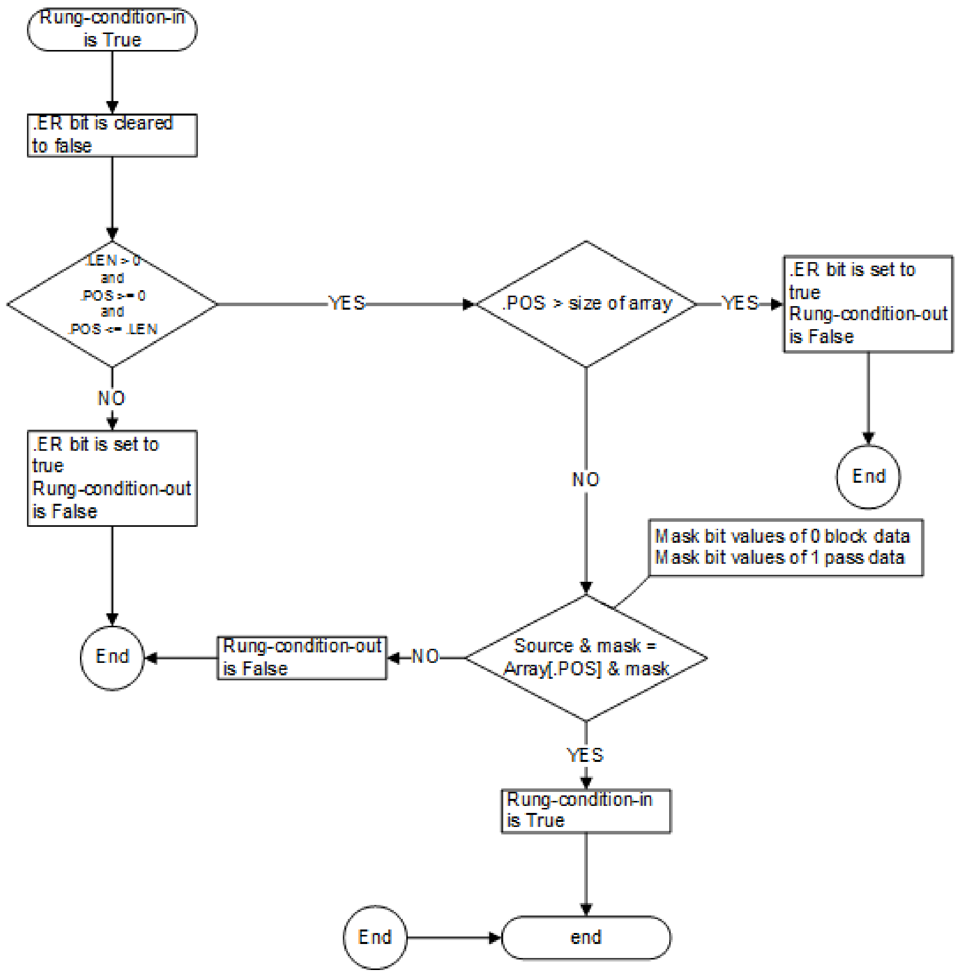
None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Ladder Diagram

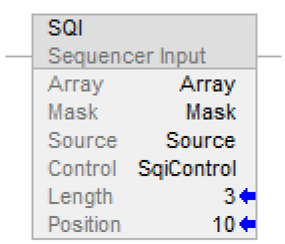
Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	See Flow Chart (True)
Postscan	N/A

Flow Chart (True)



Example

Ladder Diagram



If you use the SQI instruction without a paired SQO instruction, you have to externally increment the sequencer array.

The rung-condition-in will be set to true when the instructions enableOut will be true when the result of ANDing the array value specified by the Position e.g. Array[Position] with the Mask value is equal to the result of ANDing the Source value with the Mask value, otherwise the rung-condition-out will be cleared to false.

Sequencer Load (SQL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SQL instruction loads the source operand value into the sequencer array.

When .EN transitions from false to true, the .POS is incremented. The .POS is reset to 1 when the .POS becomes > or = to .LEN. The SQL instruction loads the Source value into the Array at the new position.

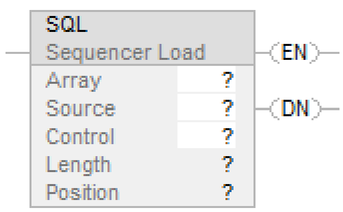
When .EN is true the SQL instruction loads the Source value into the Array at the current position.

Typically use the same CONTROL structure as the SQI and SQO instructions.

IMPORTANT: You must test and confirm that the instruction does create unwanted changes.

Available Languages

Ladder Diagram



Operands

The data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

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 INT DINT	tag immediate	Data to load into the sequencer array at a location specified by .POS.
Control	CONTROL	tag	Control structure for the operation The same control tag should be used in the SQI and SQO instructions
Length	DINT	immediate	This represents the CONTROL structure .LEN.

Position	DINT	immediate	This represents the CONTROL structure .POS.

CONTROL Structure

Mnemonic	Data Type	Description
.EN (Enable)	BOOL	The enable bit indicates the SQL instruction is enabled.
.DN (Done)	BOOL	The done bit is set when all the specified elements have been loaded into Array.
.ER (Error)	BOOL	The error bit is set when .LEN < or = to 0, .POS < 0, or .POS > .LEN.
.LEN (Length)	DINT	The length specifies the number of sequencer steps in the sequencer array.
.POS (Position)	DINT	The position identifies where in the Array the Source value will be stored.

Affects Math Status Flags

No

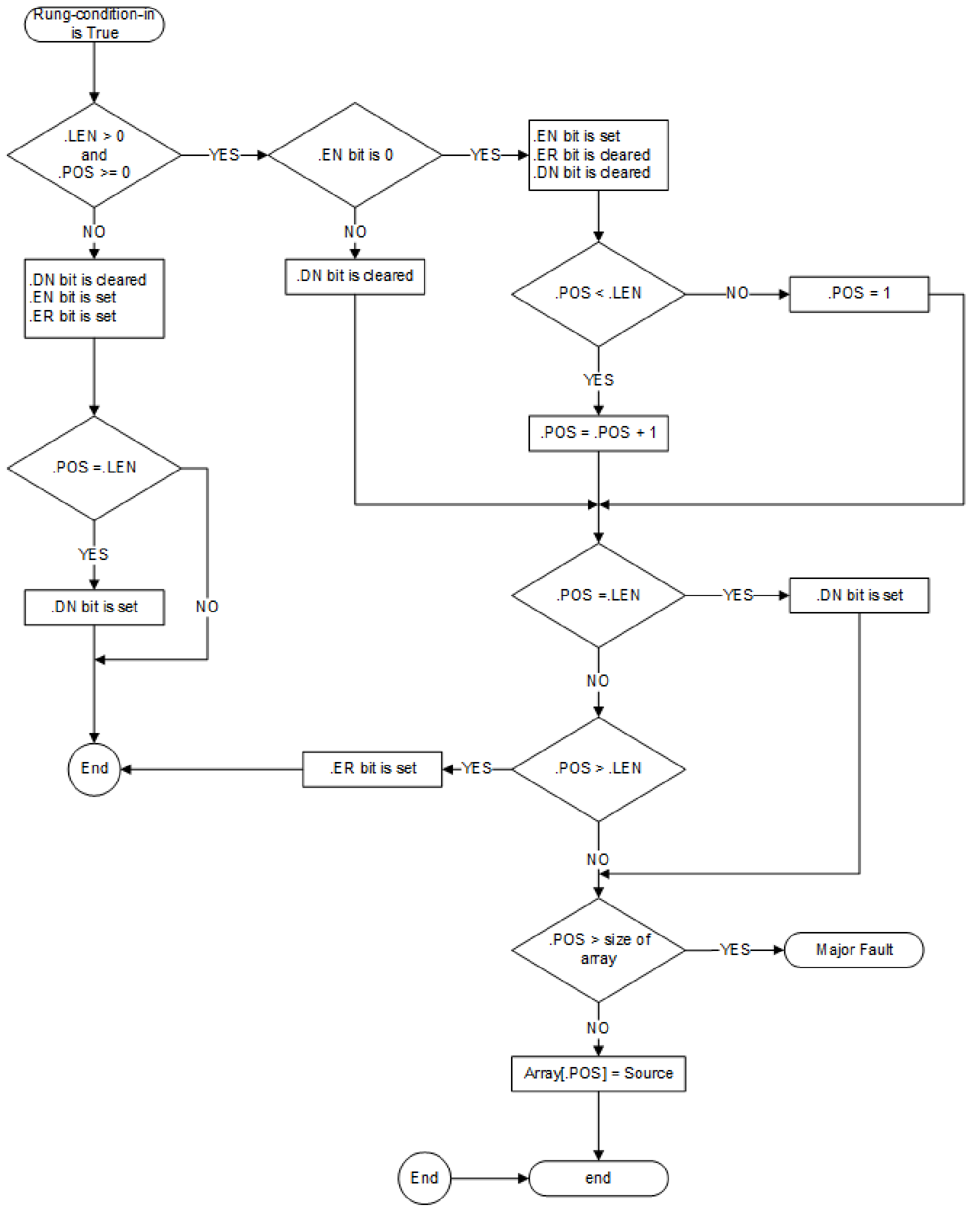
Major/Minor Faults

A major fault will occur if:	Fault Type	Fault Code
position > size of Array	4	20

Execution

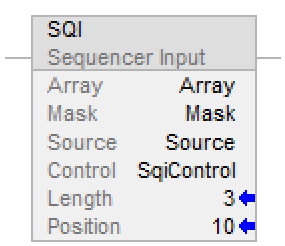
Condition/State	Action Taken
Prescan	The .EN is set to true.
Rung-condition-in is false	The .EN is cleared to false
Rung-condition-in is true	See Flow Chart (True)
Postscan	N/A

Flow Chart - True



Example

Ladder Diagram



When enabled, the SQL instruction loads value_3 into the next position in the sequencer array, which is array_dint[5] in this example.

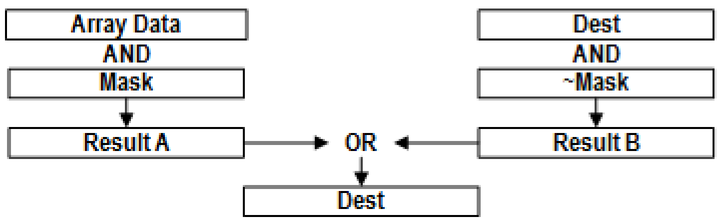
Sequencer Output (SQO)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SQO instruction sets output conditions for the next step of a sequence pair of SQO/SQI instructions.

When .EN transitions from false to true, the .POS is incremented. The .POS is reset to 1 when the .POS becomes greater than or equal to .LEN

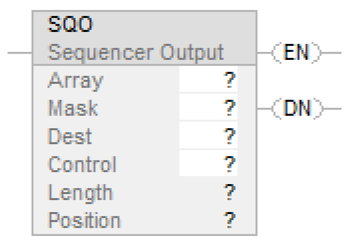
When .EN is true the SQO instruction moves the Array data at the .POS through the Mask and then moves the current Destination value through the complemented Mask. The results of those operations are ORed together and the result is stored in the Destination.



Typically, you should use the same CONTROL structure as the SQI on page 601 and SQL on page 604 instructions.

Available Languages

Ladder Diagram



Operands

The data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

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 INT DINT	tag immediate	Used to determine which bits to block (0) or pass (1) and applied during the output masking operation.

Operand	Type	Format	Description
Destination	DINT	tag	Output data from the sequencer array. This value is used in the output masking operation.
Control	CONTROL	tag	Control structure for the operation The same control tag should be used in the SQI and SQL instructions
Length	DINT	immediate	Number of elements in the Array (sequencer table) to the output
Position	DINT	immediate	Current position in the array Initial value is typically 0.

CONTROL Structure

Mnemonic	Data Type	Description
.EN (Enable)	BOOL	The enable bit indicates the SQO instruction is enabled.
.DN (Done)	BOOL	The done bit is set when .POS = .LEN
.ER (Error)	BOOL	Indicates the instruction encountered an error.
.LEN (Length)	DINT	The length specifies the number of sequencer steps in the sequencer array.
.POS (Position)	DINT	The position identifies the Array element that the instruction is currently using in the output masking operation.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

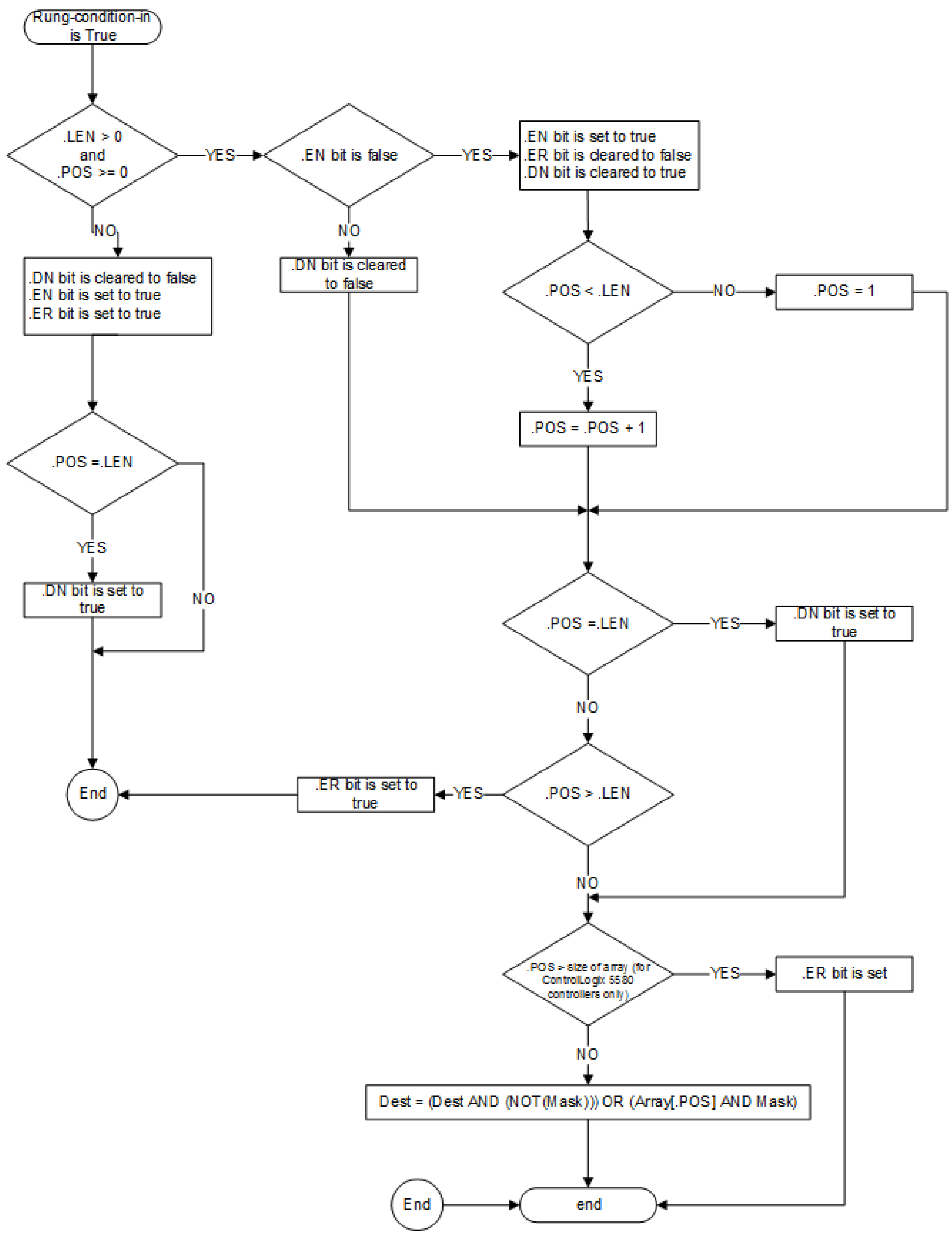
Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN is set to true.

Condition/State	Action Taken
Rung-condition-in is false	The .EN is cleared to false
Rung-condition-in is true	See the following Flow Chart (True)
Postscan	N/A

Flow Chart (True)

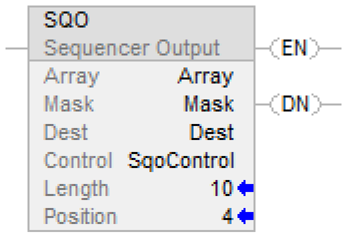


Example

The Mask value is AND'd with the array value e.g. Array[SqoControl.POS]. The complement of the Mask value is AND'd with the current Dest value. The results of these two operations are then OR'd together and the result is stored to the Dest.

To reset .POS to the initial value (.POS = 0), us'e a RES instruction to clear the control structure. This example uses the status of the first-scan bit to clear the .POS value.

Ladder Diagram



Program Control Instructions

Use the program control instructions to change the flow of logic.

Available Instructions

Ladder Diagram

JMP on page 620	LBL on page 620	JSR on page 622	JXR on page 616	SBR on page 622	RET on page 622	TND on page 639	MCR on page 630

UID on page 644	UIE on page 644	SFR on page 637	SFP on page 635	EVENT on page 640	AFI on page 614	EOT on page 615	NOP on page 633

Function Block

JSR on page 622	RET on page 622	SBR on page 622

Structured Text

JSR on page 622	RET on page 622	SBR on page 622	TND on page 639	EVENT on page 640	UID on page 644	EOT on page 615	SFR on page 637

NOTE: SFP on page 635

If you want to:	Use this instruction:
Jump over a section of logic that does not always need to be executed.	JMP LBL
Jump to a separate routine, pass data to the routine, execute the routine, and return results.	JSR SBR RET
Jump to an external routine	JXR
Mark a temporary end that halts routine execution.	TND
Disable all the rungs in a section of logic	MCR
Disable user tasks.	UID
Enable user tasks.	UIE
Pause a sequential function chart	SFP
Reset a sequential function chart	SFR
End a transition for a sequential function chart	EOT
Trigger the execution of an event task	EVENT
Disable a rung	AFI
Insert a placeholder in the logic.	NOP

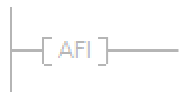
Always False (AFI)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The AFI instruction sets the EnableOut to false.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

Ladder Diagram

None

Description

The AFI instruction sets its EnableOut to false.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults

Execution

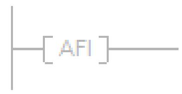
All conditions below the thick solid line can only occur during Normal Scan mode.

Condition	Action
Prescan	N/A
Rung-condition-in is false	Clear EnableOut to false.
Rung-condition-in is true	Clear EnableOut to false.
Postscan	N/A

Examples

Ladder Diagram

Use the AFI instruction to temporarily disable a rung while you are debugging a program. AFI disables all the instructions on this rung.



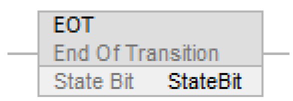
End of Transition (EOT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The EOT instruction is used to set the state of a transition. It typically occurs in a subroutine called from a transition (JSR). The state bit parameter used in EOT determines the state of the Transition. If the state bit is set to true, the SFC transitions to next state else EOT acts as NOP.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

EOT(StateBit);

Operands

Ladder Diagram

Operand	Type	Format	Description
State Bit	BOOL	tag	state of the transition (0=executing, 1=completed)

Structured Text

Operand	Type	Format	Description
State Bit	BOOL	tag	state of the transition (0=executing, 1=completed)

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

Because the EOT instruction returns a boolean state, multiple SFC routines can share the same routine that contains the EOT instruction. If the calling routine is not a transition, the EOT instruction acts as a NOP instruction.

In a Logix controller, the return parameter returns the transition state, since rung condition is not available in all Logix programming languages.

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction returns the data bit value to the calling routine.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	N/A
Normal execution	The instruction returns the data bit value to the calling routine.
Postscan	N/A

Example

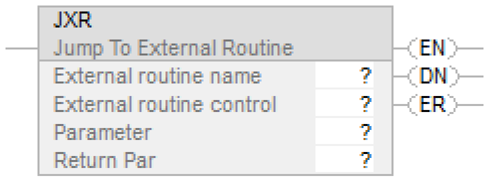
Jump to External Routine (JXR)

This information applies to the SoftLogix 5800 controller only.

The JXR instruction executes an external routine.

Available Languages

Ladder Diagram



Function Block

This instruction is not available for function block.

Structured Text

This instruction is not available for structured text.

Operands

Ladder Diagram

Operand	Type	Format	Description
External routine name	ROUTINE	Name	External routine to execute
External routine control	EXT_ROUTINE_CONTROL	Tag	Control structure
Parameter	BOOL SINT INT DINT REAL 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 SINT INT DINT REAL	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

EXT_ROUTINE_CONTROL Structure

Mnemonic	Data Type	Description	Implementation
ErrorCode	SINT	If an error occurs, this value identifies the error. Valid values are from 0-255.	There are no predefined error codes. The developer of the

			external routine must provide the error codes.
NumParams	SINT	This value indicates the number of parameters associated with this instruction.	Display only - this information is derived from the instruction entry.
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.	Display only - this information is derived from the instruction entry.
ReturnParamDef	EXT_ROUTIN_ PARAMETERS	This value contains definitions of the return parameter from the external routine. There is only one return parameter.	Display only - this information is derived from the instruction entry.
EN	BOOL	When set, the enable bit indicates that the JXR instruction is enabled.	The external routine sets this bit.
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.	Display only - this information is derived from the instruction entry.
DN	BOOL	The done bit is set when the external routine has executed once to completion.	The external routine sets this bit.
ER	BOOL	The error bit is set if an error occurs. The instruction stops executing until the program clears the error bit.	The external routine sets this 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.	The controller sets this bit to reflect scan status.
EnableOut	BOOL	Enable output.	The external routine sets this bit.
EnableIn	BOOL	Enable input.	The controller sets this bit to reflect rung-condition-in. The instruction executes regardless of rung condition.

			The developer of the external routine should monitor this status and act accordingly.
User1	BOOL	These bits are available for the user. The controller does not initialize these bits.	Either the external routine or the user program can set 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)	The controller sets these bits to reflect scan status.
ScanType0	BOOL		

Description

Use the Jump to External Routine (JXR) instruction to call the external routine from a ladder routine in your project. The JXR instruction supports multiple parameters so you can pass values between the ladder routine and the external routine.

The JXR instruction is similar to the Jump to Subroutine (JSR) instruction. The JXR instruction initiates the execution of the specified external routine:

- The external routine executes one time.
- After the external routine executes, logic execution returns to the routine that contains the JXR instruction.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if	Fault Type	Fault Code:
An exception occurs in the external routine DLL. The DLL could not be loaded. The entry point was not found in the DLL.	4	88

Execution

The JXR can be synchronous or asynchronous depending on the implementation of the DLL. The code in the DLL also determines how to respond to scan status, rung-condition-in status, and rung-condition-out status.

For more information on using the JXR instruction and creating external routines, see thehttps://literature.rockwellautomation.com/idc/groups/literature/documents/um/1789-um002_-en-p.pdf.

Jump to Label (JMP) and Label (LBL)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The JMP and LBL instructions skip portions of ladder logic.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

Ladder Diagram

Operand	Format	Description
JMP instruction		
Label name	label name	Enter the name for associated LBL instruction
LBL instruction		
Label name	label name	Execution jumps to the references LBL instruction

Description

When true, the JMP instruction skips to the referenced LBL instruction and the controller continues executing from there. When false, the JMP instruction does not affect ladder execution.

The JMP and LBL it references must be in the same routine.

The JMP instruction can move ladder execution forward or backward. Jumping forward to a label saves program scan time by omitting a logic segment until it is needed. Jumping backward lets the controller repeat iterations of logic.

IMPORTANT: Be careful not to jump backward an excessive number of times. The watchdog timer could time out because the scan does not complete in time.

IMPORTANT: Jumped logic is not scanned. Place critical logic outside the jumped zone.

A JMP instruction requires the associated label to exist before you:

- Download when working offline
- Accept edits when working online

The LBL instruction must be the first instruction on the rung.

A label name must be unique within a routine. The name can:

- Have as many as 40 characters
- Contain letters, numbers, and underscores (_)

Affects Math Status Flags

No.

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

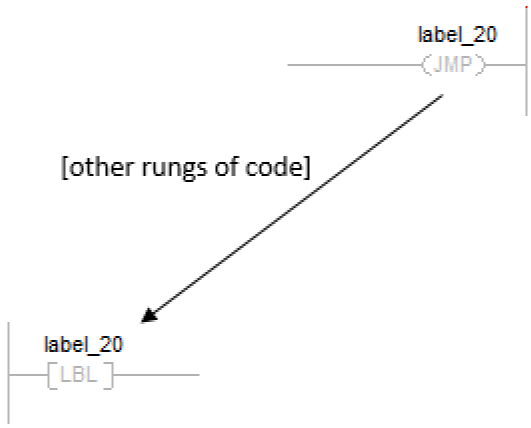
Condition	Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	(For JMP) Execution jumps to the rung that contains the LBL instruction with the referenced label name. (For LBL) no action taken
Postscan	N/A

Example

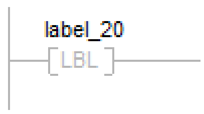
Ladder Diagram

JMP

When the JMP instruction is enabled, execution jumps over successive rungs of logic until it reaches the rung that contains the LBL instruction with label_20.



LBL



Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

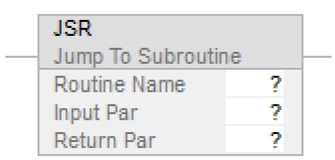
The JSR instruction invokes another routine. When that routine completes, the execution returns to the JSR instruction.

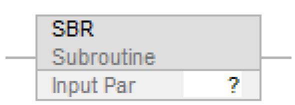
The SBR instruction receives the input parameters passed by the JSR.

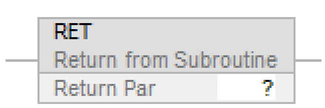
The RET instruction passes return parameters back to the JSR and ends the scan of the subroutine.

Available Languages

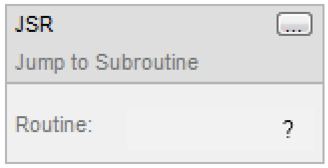
Ladder Diagram

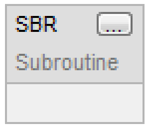


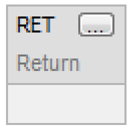




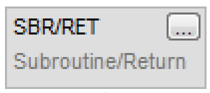
Function Block







Sequential Function Chart



Structured Text

JSR(RoutineName,InputCount,InputPar,ReturnPar);

SBR(InputPar);

RET(ReturnPar);

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

WARNING: For each parameter in an SBR or RET instruction, use the same data type (including any array dimensions) as the corresponding parameter in the JSR instruction. Using different data types may yield unexpected results.

Ladder Diagram

JSR Instruction

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Routine Name	ROUTINE	ROUTINE	name	Subroutine to execute
Input Par	BOOL SINT INT DINT REAL structure	BOOL SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL structure	immediate tag array tag	Data from this routine to copy to a tag in the subroutine. • Input parameters are optional • Enter a maximum of 40 input parameters, if needed.
Return Par	BOOL SINT INT DINT REAL structure	BOOL SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL structure	tag array tag	Tag in this routine to copy result from subroutine. • Return parameters are optional • Enter a maximum of 40 return parameters, if needed

SBR Instruction

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Input Par	BOOL SINT INT DINT REAL structure	BOOL SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL structure	tag array tag	• Tag in this routine into which to copy the corresponding input parameter (maximum 40) from the JSR instruction.

RET Instruction

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Return Par	BOOL SINT INT DINT REAL structure	BOOL SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL structure	immediate tag array tag	Data from this routine to copy to the corresponding return parameter (maximum 40) in the JSR instruction.

Affects Math Status Flags

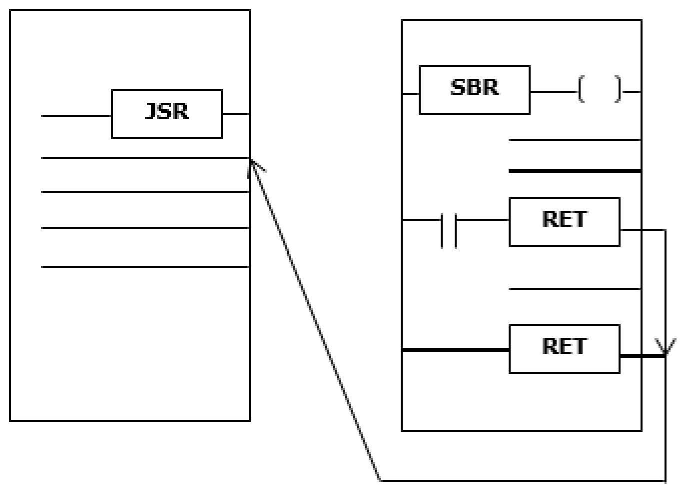
No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
JSR instruction has fewer input parameters than SBR instruction	4	31
JSR instruction jumps to a fault routine	4	990 or user-supplied
RET instruction has fewer return parameters than JSR instruction	4	31
Main routine contains a RET instruction	4	31

Operation

IMPORTANT: Any routine may contain a JSR instruction but a JSR instruction cannot call (execute) the main routine.



The JSR instruction initiates the execution of the specified routine, which is referred to as a subroutine:

- The subroutine executes each time it is scanned.
- After the subroutine executes, logic execution returns to the routine that contains the JSR instruction and continues with the instruction following the JSR.

To program a jump to a subroutine, follow these guidelines.

JSR

- To copy data to a tag in the subroutine enter an input parameter.
- To copy a result of the subroutine to a tag in this routine, enter a return parameter.
- Enter up to 40 inputs and enter up to 40 return parameters as needed.

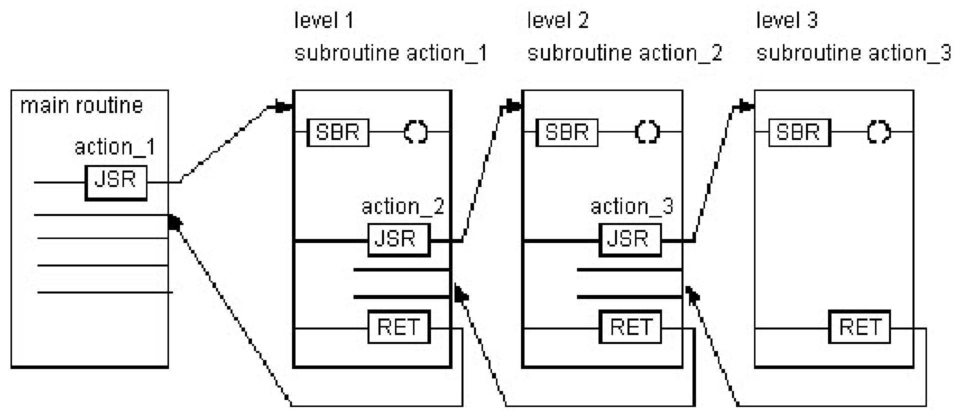
SBR

- If the JSR instruction has an input parameter enter an SBR instruction.
- Place SBR instruction as the first instruction in the routine.
- For each input Parameter in the JSR Instruction, enter the tag into which you want to copy the data.

RET

- If the JSR instruction has a return parameter, enter an RET instruction.
- Place the RET instruction as the last instruction in the routine.
- For each return parameter in the JSR instruction, enter a return parameter to send to the JSR instruction.
- In a ladder routine, place additional RET instructions to exit the subroutine based on different input conditions, if required (Function block routines only permit one RET instruction).

Invoke up to 25 nested subroutines, with a maximum of 40 parameters passed into a subroutine, and a maximum of 40 parameters returned from a subroutine.



Tip: Select the **Edit > Edit Ladder Element** menu to add and remove variable operands. For the JSR and SBR instructions, add Input Parameter. For JSR and RET instructions, add Output Parameter. For all three instructions, remove Instruction Parameter.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The rung is set to false. The controller executes all subroutines. To ensure that all rungs in the subroutine are prescanned, the controller ignores RET instructions (that is, RET instructions do not exit the subroutine). Input and return parameters are not passed. If the same subroutine is invoked multiple times, it will only be prescanned once.
Rung-condition-in is false (to the JSR instruction)	N/A
Rung-condition-in is true	Parameters are passed and the subroutine is executed.
Postscan	Same action as Prescan

Function Block

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
EnableIn is false	N/A
EnableIn is true	Parameters are passed and the subroutine is executed
Instruction first run	N/A
Instruction first scan	N/A
Postscan	See Postscan in the Ladder Diagram table.

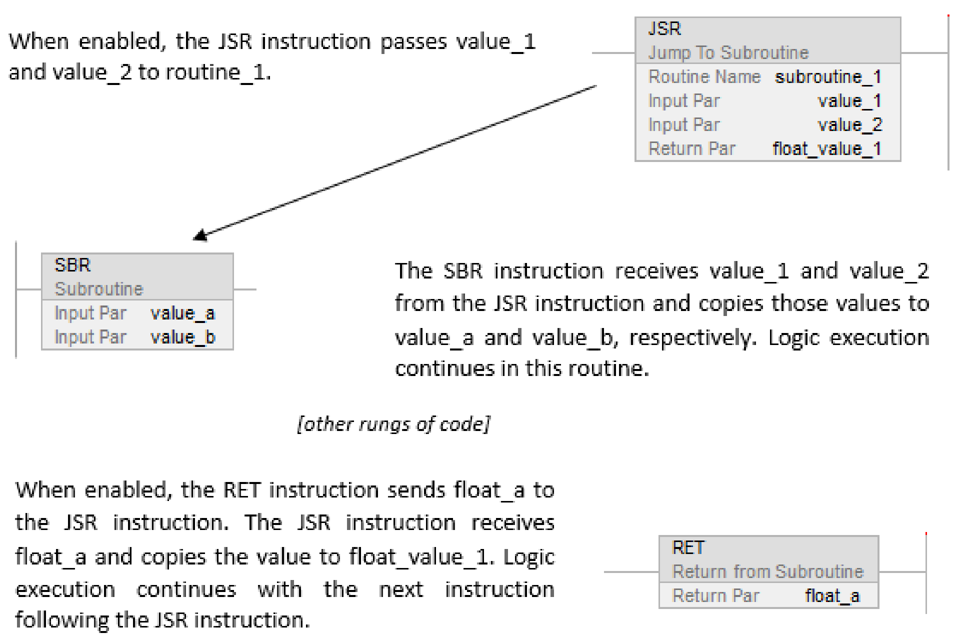
Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal Execution	Parameters are passed and the subroutine is executed.
Postscan	See Postscan in the Ladder Diagram table.

Examples

Example 1

Ladder Diagram



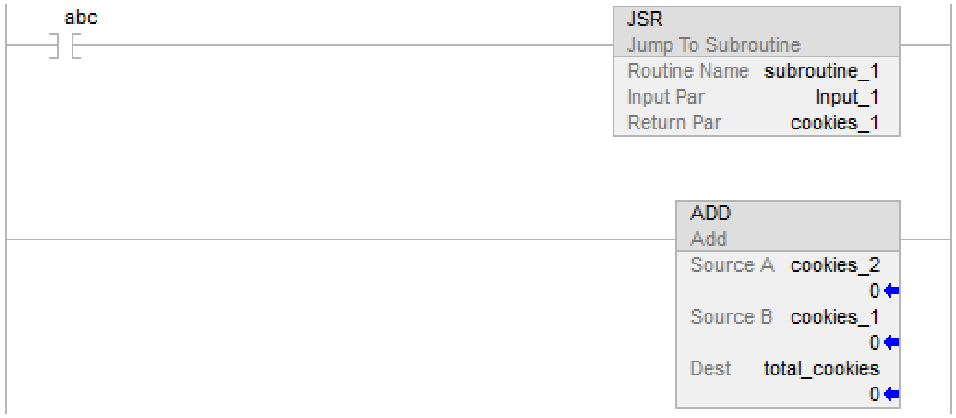
Structured Text

Routine	Program
Main routine	JSR(routine_1,2,value_1,value_2,float_value_1);
Subroutine	SBR(value_a,value_b); <statements>; RET(float_a);

Example 2

Ladder Diagram

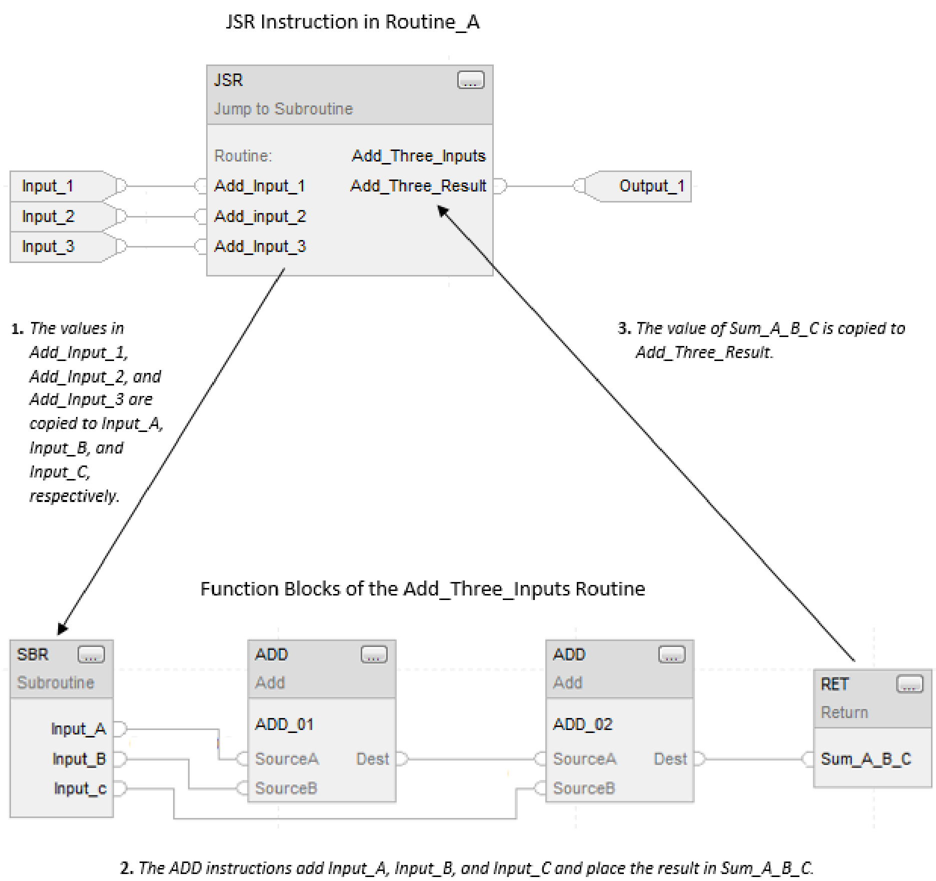
Main routine



subroutine_1

Example 3

Function Block



Add Input Parameter command

Choose this command to add an input operand to a JSR or SBR instruction.

Master Control Reset (MCR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The MCR instruction simulates a master control relay (a mandatory hard-wired relay that can be de-energized by any series-connected emergency stop switch). Whenever the relay is de-energized, its contacts open to de-energize all application I/O devices. The MCR instruction can selectively disable a section of rungs.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

Description

The MCR instruction is able to override the normal behavior of rungs; forcing every instruction to execute as if rung-condition-in is false. Typically, false execution of an instruction is faster than true so, selectively disabling unneeded sections of code could result in an overall improvement in scan time.

Each time the MCR instruction is executed with rung-condition-in false, the override behavior is toggled. Consequently, two MCR instructions are normally required: one to start the "zone" and a second to terminate it.

The starting MCR is typically conditioned by one or more input instructions. When the input conditions are false, the zone will be disabled. When the input conditions are true, the zone will operate normally.

The terminating MCR is normally unconditional. If the zone is enabled, the terminating MCR will be true so it will do nothing. If the zone is disabled, however, the terminating MCR will be false so it will toggle the override, re-enabling the rungs that follow it.

When you program an MCR zone, note that:

MCR instruction must be the last instruction of a rung.

- You should end the zone with an unconditional MCR instruction. If the terminating MCR is false, and the zone is enabled, the terminating MCR will disable all of the rungs that follow it.
- You cannot nest one MCR zone within another. There is only one override bit in each program. Each MCR instruction has the ability to toggle this override. Attempting to nest MCR zones will actually result in multiple smaller zones to be created.
- Do not jump into an MCR zone. If the starting MCR is not executed, the zone will not be disabled.
- The override bit is automatically reset at the end of the routine. If an MCR zone continues to the end of the routine, you do not have to program an MCR instruction to end the zone, however, to avoid confusion when online editing, it is recommended that the terminating MCR always be used.

If the MCR is disabled in a subroutine or an AOI, the override bit will be reset when the subroutine/AOI returns.

AOIs have their own override bit which is initialized when the AOI is invoked. If an AOI is invoked from within a disabled MCR zone, the false scan mode routine will execute normally. After the AOI returns, the state of the zone will be restored to what it was before the AOI was invoked.

IMPORTANT: The MCR instruction is not a substitute for a hard-wired master control relay that provides emergency-stop capability. You should still install a hard-wired master control relay to provide emergency I/O power shutdown.

IMPORTANT: Do not overlap or nest MCR zones. Each MCR zone must be separate and complete. If they overlap or nest, unpredictable machine operation could occur with possible damage to equipment or injury to personnel.

Place critical operations outside the MCR zone. If you start instructions such as timers in a MCR zone, instruction execution becomes false when the zone is disabled and the timer will be cleared.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	The override behavior is toggled enabling or disabling the rungs that follow.
Rung-condition-in is true	N/A
Postscan	N/A

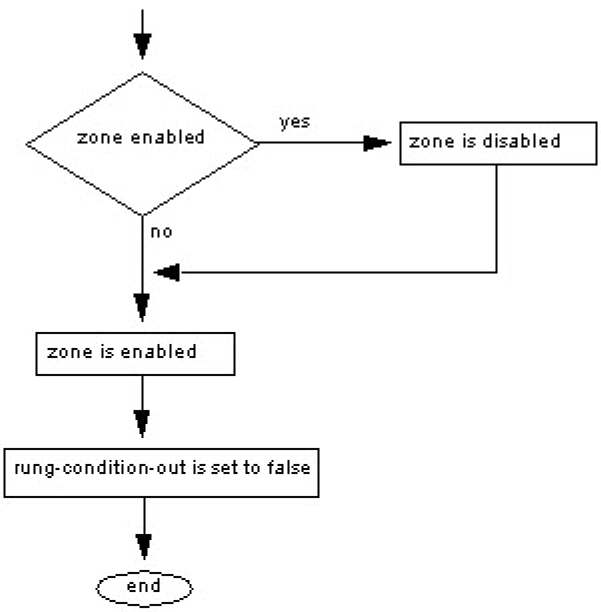
Example

Ladder Diagram

When the first MCR instruction is enabled (input_1, input_2, and input_3 are set), the controller executes the rungs in the MCR zone (between the two MCR instructions) and sets or clears outputs, depending on input conditions.

When the first MCR instruction is disabled (input_1, input_2, and input_3 are not all set), the controller executes the rungs in the MCR zone (between the two MCR instructions) and the EnableIn goes false for all the rungs in the MCR zone, regardless of input conditions.

MCR Flow Chart (False)



No Operation (NOP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The NOP instruction functions as a placeholder.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

Ladder Diagram

None

Description

You can place the NOP instruction anywhere on a rung. When enabled the NOP instruction performs no operation. When disabled, the NOP instruction performs no operation.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	N/A
Postscan	N/A

Examples

Ladder Diagram

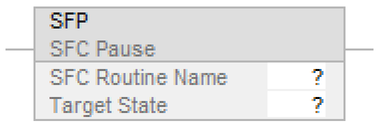
Pause SFC (SFP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SFP instruction pauses an SFC routine.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

SFP(SFCRoutineName,TargetState);

Operands

Ladder Diagram

Operand	Type	Format	Description
SFCRoutineName	ROUTINE	name	SFC routine to pause
TargetState	DINT	immediate	Select one: • Executing (or enter 0) • Paused (or enter 1)

Structured Text

Operand	Type	Format	Description
SFCRoutineName	ROUTINE	name	SFC routine to pause
TargetState	DINT	immediate	Select one: • Executing (or enter 0) • Paused (or enter 1)

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The SFP instruction lets you pause an executing SFC routine.

Affects Math Status Flags

No

Fault Conditions

A major fault will occur if:	Fault Type	Fault Code
The routine type is not an SFC routine	4	85

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

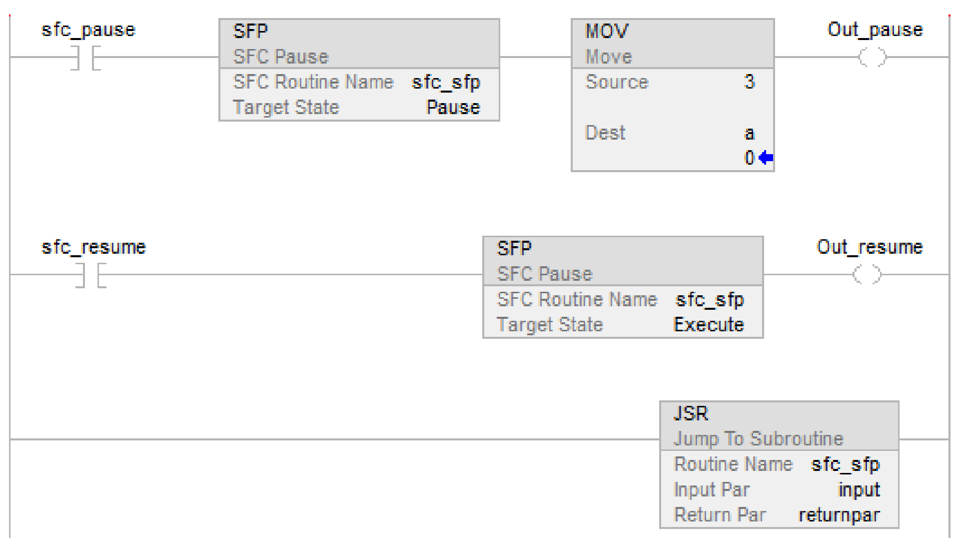
Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false.	N/A
Rung-condition-in is true	The instruction pauses or resumes execution of the specified SFC routine.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	N/A
Normal execution	The instruction pauses or resumes execution of the specified SFC routine.
Postscan	N/A

Example

Ladder Diagram



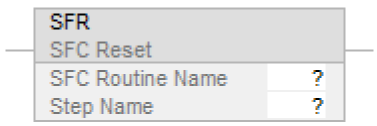
Reset SFC (SFR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The SFR instruction resets the execution of an SFC routine at a specified step.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

SFR(SFCRoutineName,StepName);

Operands

Ladder Diagram

Operand	Type	Format	Description
SFCRoutineName	ROUTINE	name	SFC routine to reset
StepName	SFC_STEP	tag	Target step where to resume execution

Structured Text

Operand	Type	Format	Description
SFCRoutineName	ROUTINE	name	SFC routine to reset
StepName	SFC_STEP	tag	Target step where to resume execution

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

When the SFR instruction is enabled:

- In the specified SFC routine, all stored actions stop executing (reset).
- The SFC begins executing at the specified step.
- If the target step is 0, the chart will be reset to is initial step.

The Logix implementation of the SFR instruction differs from that in the PLC-5 controller. In the PLC-5 controller, the SFR executes when the rung condition is true. After reset, the SFC would remain paused until the rung containing the SFR became false. This allowed the execution following a reset to be delayed. This pause/un-pause feature of the PLC-5 SFR instruction was decoupled from the rung condition and moved into the SFP instruction.

Affects Math Status Flags

No

Fault Conditions

A major fault will occur if:	Fault Type	Fault Code
The routine type is not an SFC routine	4	85
Specified target step does not exist in the SFC routine	4	89

See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

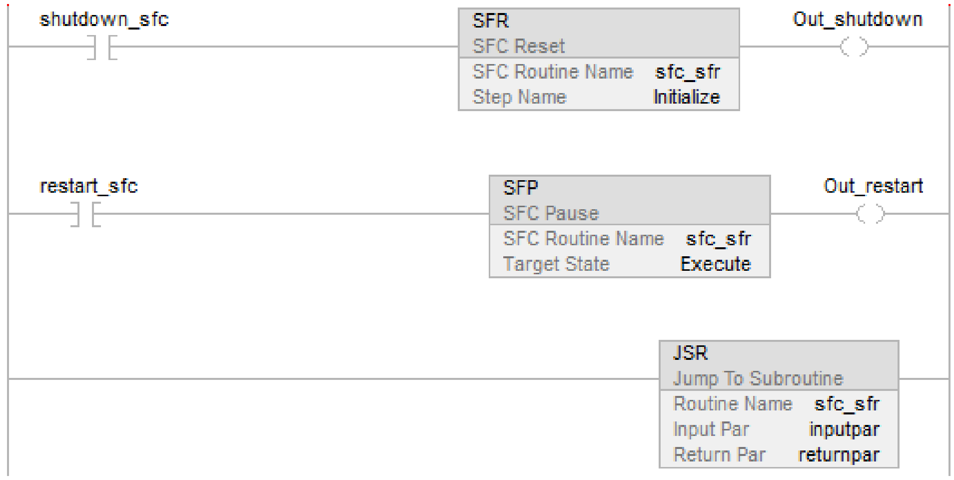
Condition/State	Action Taken
Prescan	N/A
Rung-condition-inis false	N/A
Rung-condition-in is true	The instruction reset the specified SFC routine execution to a particular step.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	N/A
Normal execution	The instruction reset the specified SFC routine execution to a particular step.
Postscan	N/A

Example

Ladder Diagram



Temporary End (TND)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The TND instruction conditionally ends a routine.

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

TND();

Operands

Ladder Diagram

None

Structured Text

None

Description

When enabled, the TND instruction acts as the end of the routine. If the TND instruction is in a subroutine, control returns to the calling routine. If the TND instruction is in a main routine, control returns to the next program within the current task.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true.	The routine ends
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See rung-condition-in is true in the Ladder Diagram table
Postscan	See Postscan in the Ladder Digram table.

Structured Text

```
InputA[:=] OutputB;

IF (InputA) THEN

  TND();

END_IF;


InputE [:=] OutputF;
```

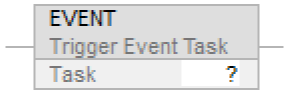
Trigger Event Task (EVENT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The EVENT instruction triggers one execution of an event task.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

EVENT(task_name);

Operands

Ladder Diagram

Operand	Type	Format	Description
Task	TASK	name	Event task to execute. If a task is specified that is not the Event task, the specified task will not be executed.

Structured Text

Operand	Type	Format	Description
Task	TASK	name	Event task to execute. If a task is specified that is not the Event task, the specified task will not be executed.

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

Use the EVENT instruction to programmatically execute an event task.

Each time the instruction executes, it trigger the specified event task.

Make sure that you give the event task enough time to complete its execution before you trigger it again. If not, an overlap occurs.

If you execute an EVENT instruction while the event task is already executing, the controller increments the overlap counter, but it does not trigger the event task.

EVENT instruction can be used to trigger Event Task with all the trigger types.

Programmatically Determine if an EVENT Instruction Triggered a Task

To determine if an EVENT instruction triggered an event task, use a Get System Value (GSV) instruction to monitor the Status attribute of the task.

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Instruction</th><th colspan="2">Description</th></tr>
<tr><td rowspan="5">Status</td><td rowspan="5">DINT</td><td rowspan="5">GSV
SSV</td><td colspan="2">Provides status information about the task. Once the controller sets a bit, you must manually clear the bit to determine if another fault of that type occurred.</td></tr>
<tr><td>To determine if</td><td>Examine this bit</td></tr>
<tr><td>An EVENT instruction triggered the task (event task only)</td><td>0</td></tr>
<tr><td>A timeout triggered the task (event task only)</td><td>1</td></tr>
<tr><td>An overlap occurred for this task</td><td>2</td></tr>
</table>

The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit. Use a Set System Value (SSV) instruction to set the attribute to a different value.

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	N/A
Normal execution	The instruction executes.
Postscan	N/A

Examples

Example 1

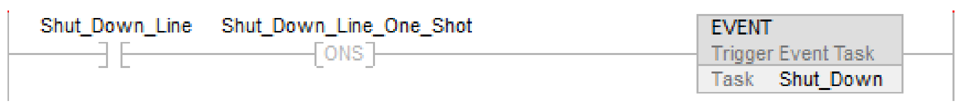
A controller uses multiple programs, but a common shut down procedure. Each program uses a program-scoped tag named Shut_Down_Line that turns on if the program detects a condition that requires a shut down. The logic in each program executes as follows.

If Shut_Down_Line = on (conditions require a shut down) then

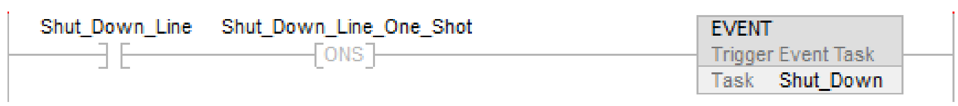
Execute the Shut_Down task one time

Ladder Diagram

Program A



Program B



Structured Text

Program A

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN

EVENT (Shut_Down);

END_IF;

Shut_Down_Line_One_Shot:=Shut_Down_Line;
```

Program B

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN

EVENT (Shut_Down);

END_IF;

Shut_Down_Line_One_Shot:=Shut_Down_Line;
```

Example 2

The following example uses an EVENT instruction to initialize an event task. Another type of event normally triggers the event task.

Continuous Task

IF Initialize_Task_1 = 1 THEN

The ONS instruction limits the execution of the EVENT instruction to 1 scan.

The EVENT instruction triggers an execution of Task_1 (event task).

Task_1 (event task)

The GSV instruction sets Task_Status (DINT tag) = Status attribute for the event task. In the Instance Name attribute, THIS means the TASK object for the task that the instruction is in (e.g., Task_1).

If Task_Status.0=1 then an EVENT instruction triggered the event task (i.e., when the continuous task executes its EVENT instruction to initialize the event task).

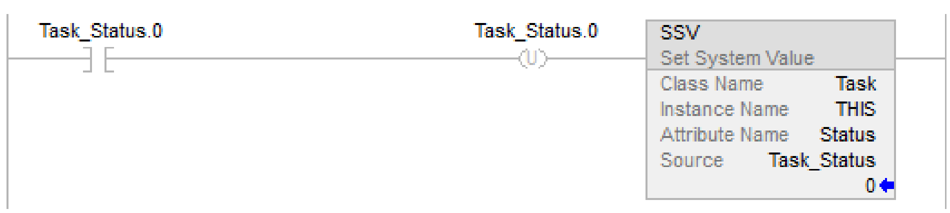
The RES instruction resets a counter the event task uses.

The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit.

If Task_Status.0 = 1 then clear that bit.

The OTU instruction sets Task_Status.0 = 0.

The SSV instruction sets the Status attribute of THIS task (Task_1) = Task_Status. This includes the cleared bit.



User Interrupt Disable (UID)/User Interrupt Enable (UIE)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The UID instruction and the UIE instruction work together to prevent a small number of critical rungs from being interrupted by other tasks.

vailable Languages

Ladder Diagrams

Function Block

This instruction is not available in function block.

Structured Text

UID();

UIE();

Operands

Ladder Diagram

This instruction is not available in ladder diagram.

Structured Text

This instruction is not available in structured text. You must enter the parentheses () after the instruction mnemonic, even though there are no operands.

Description

When the rung-condition-in is true, the:

- UID instruction prevents higher-priority tasks from interrupting the current task, but does not disable execution of a fault routine or the Controller Fault Handler.
- UIE instruction enables other tasks to interrupt the current task.

To prevent a series of rungs from being interrupted:

1. Limit the number of rungs that you do not want interrupted to as few as possible. Disabling interrupts for a prolonged period of time can produce communication loss.
2. Above the first rung that you do not want interrupted, enter a rung and a UID instruction.
3. After the last rung in the series that you do not want interrupted, enter a rung and a UIE instruction.
4. If required, you can nest pairs of UID/UIE instructions.

When the UID is called for the first time, it bumps priority, saves the old priority, and increments a nesting counter. Each subsequent call increments the count. The UIE will decrement the nesting counter. If the new value is 0, it will restore the saved priority.

Affects Math Status Flags

No.

Fault Conditions

None specific to this instruction. See Common Attributes on page 849 for operand-related faults.

Execution

Ladder Diagram

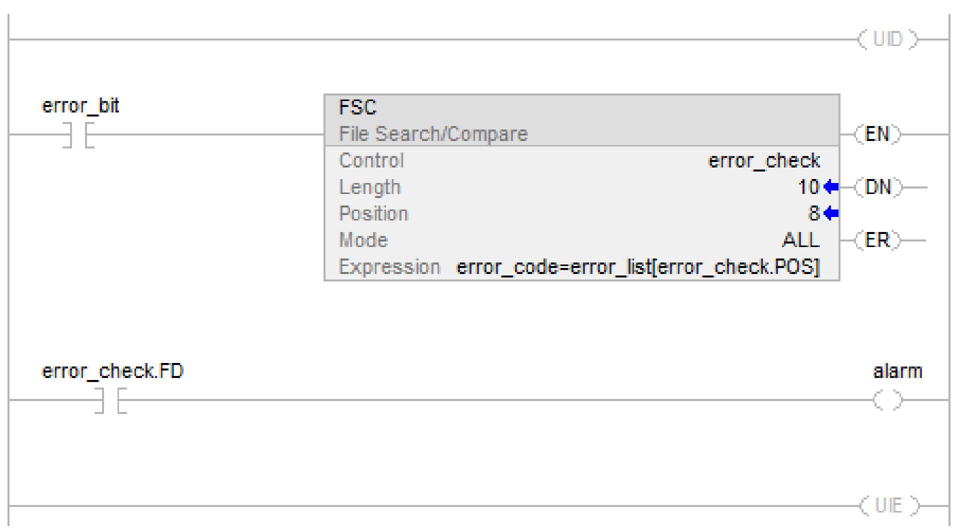
Condition/State	Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The UID instruction prevents the containing user task from being Interrupted. The UIE instruction enables the containing user task to be interrupted as is normally in the case.
Postscan	N/A

Structured Text

Condition/State	Action
Prescan	N/A
Normal execution	The UID instruction prevents the containing user task from being Interrupted. The UIE instruction enables the containing user task to be interrupted as is normally in the case.
Postscan	N/A

Example

Ladder Diagram



Structured Text

UID();

<statements>

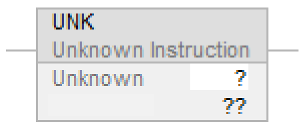
UIE();

Unknown Instruction (UNK)

The UNK instruction functions as an indication that you have entered an instruction type that is not defined within the Logix Designer instruction set.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block

Structured Text

This instruction is not available in function block.

Operands

Ladder Diagram

Operand	Type	Format	Description
Unknown	immediate	immediate	

For/Break Instructions

Use the FOR instruction to repeatedly call a subroutine. Use the BRK instruction to interrupt the execution of a subroutine.

Available Instructions

Ladder Diagram

Use the FOR instruction to repeatedly call a subroutine. Use the BRK instruction to interrupt the execution of the subroutine.

If you want to:	Use this instruction:
Repeatedly execute a routine.	For (FOR) on page 650
Terminate the repeated execution of a routine.	Break (BRK) on page 649

Break (BRK)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The BRK instruction interrupts the execution of a routine that was called by a FOR instruction.

When enabled, the BRK instruction exits the routine and returns control to the routine containing the most recently executed FOR instruction, resuming execution following that instruction. If no FOR instruction preceded this BRK instruction in its execution during this scan then BRK does not initiate.

If there are nested FOR instructions, a BRK instruction returns control to the innermost FOR instruction.

Available Languages

Ladder Diagram

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

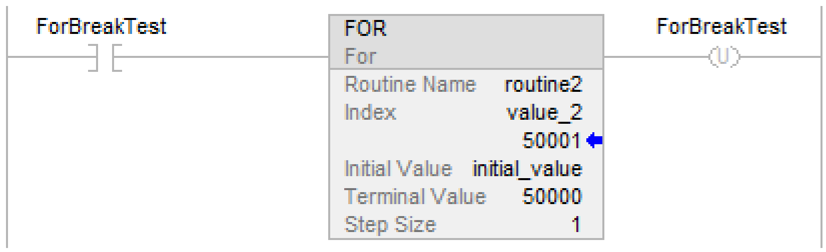
Ladder Diagram

Condition/State	Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

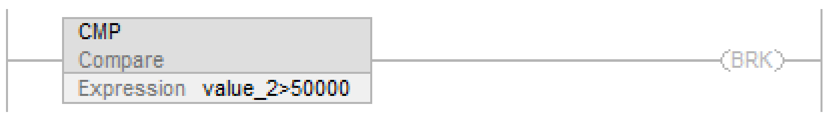
Example

When enabled, the BRK instruction stops executing the current routine and returns to the instruction that follows the calling FOR instruction.

Ladder Diagram



This is the routine2:



For (FOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The FOR instruction executes a routine repeatedly.

When enabled, the FOR instruction repeatedly executes the Routine until the Index value exceeds the Terminal value. This instruction does not pass parameters to the routine.

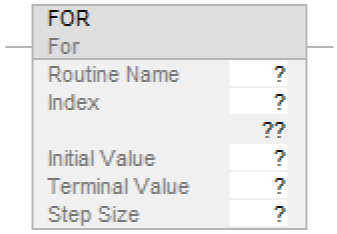
The step value can be positive or negative. If it is negative, the loop ends when the index is less than the terminal value. If it is positive, the loop ends when the index is greater than the terminal value.

Each time the FOR instruction executes the routine, it adds the Step size to the Index.

Be careful not to loop too many times in a single scan. An excessive number of repetitions can cause the controller's watchdog to timeout, which causes a major fault.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

Operand	Type	Format	Description
Routine name	ROUTINE	tag	Subroutine that is invoked each time the FOR loop executes.
Index	DINT	tag	Counts how many times the routine has been executed
Initial value	SINT INT DINT	immediate tag	Value at which to start the index
Terminal value	SINT INT DINT	immediate tag	Value at which to stop executing the routine
Step size	SINT INT DINT	immediate tag	Amount to add to the index each time the FOR instruction executes the routine

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
The nesting level limit > 25	4	94
the subroutine is an SFC and it is already executing (recursive call)	4	82

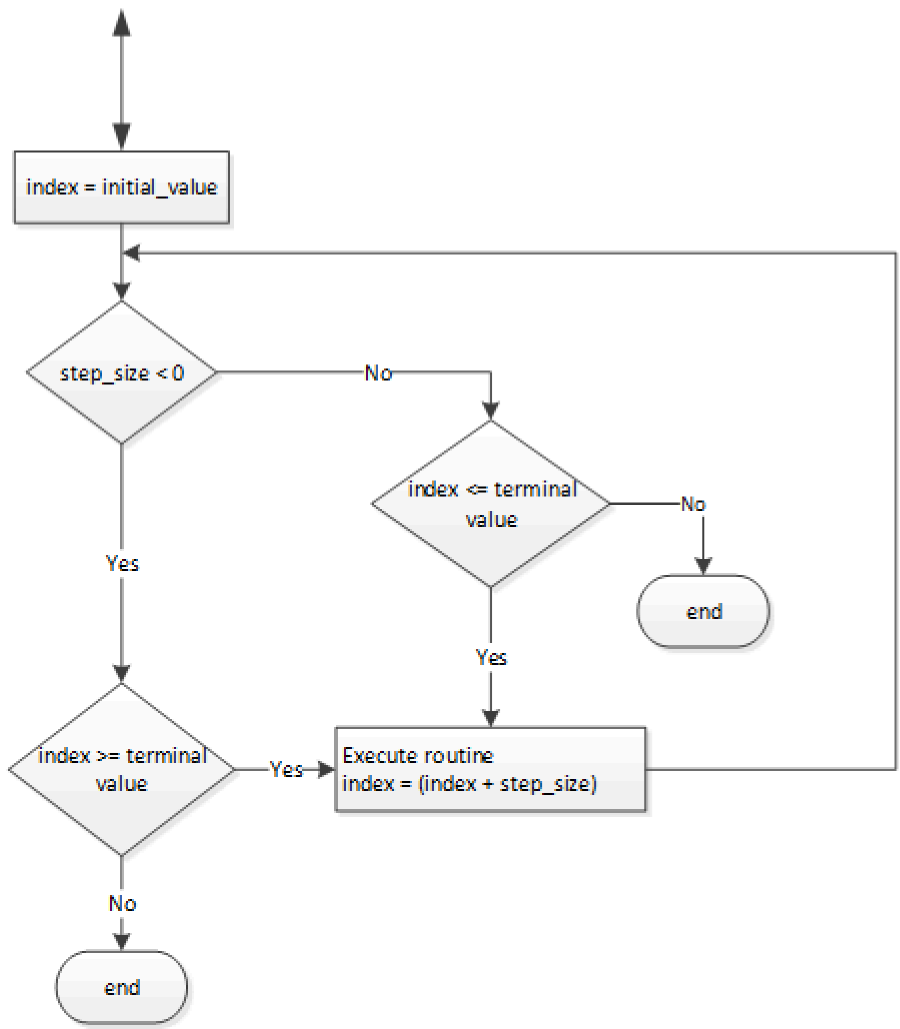
See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Condition/State	Action
Prescan	The instruction will prescan the named subroutine if it has never been prescanned before.

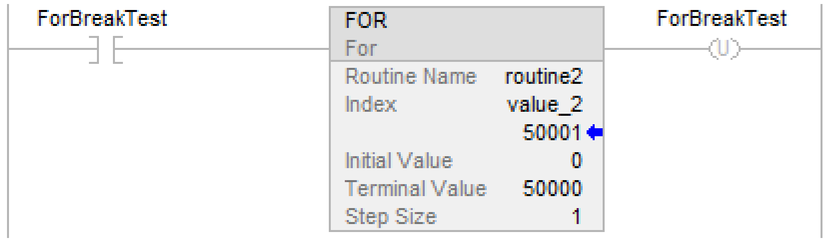
Condition/State	Action
	Tip: If recursive FOR instruction exist to the same subroutine, or multiple FOR instruction exist (non-recursive) to the same subroutine, the subroutine is pre-scanned only once. This is also true if the subordinate was prescanned by a JSR.
Rung-condition-in is false	N/A
Rung-condition-in is true	See the following FOR Flow Chart (True).
Postscan	The instruction will postscan the named subroutine exactly once.

FOR Flow Chart (True)



Examples

When enabled, the FOR instruction repeatedly executes routine_2 and increments value_2 by 1 each time. When value_2 is > 50000 or a BRK instruction is enabled, the FOR instruction no longer executes routine_2.



Special Instructions

The special instructions perform application-specific operations.

Available Instructions

If you want to:	Use this instruction:
Compare data against a known, good reference and record any mismatches.	FBC on page 665
Compare data against a known, good reference, record any mismatches, and update the reference to match the source.	DDT on page 658
Pass the source data through a mask and compare the result to reference data. Then write the source into the reference for the next comparison.	DTR on page 655
Control a PID loop.	PI on page 673

Data Transition (DTR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The DTR instruction passes the Source value through a Mask and compares the result with the Reference value.

The DTR instruction also writes the masked Source value into the Reference value for the next comparison. The Source remains unchanged.

A "1" in the mask means the data bit is passed. A "0" in the mask means the data bit is blocked.

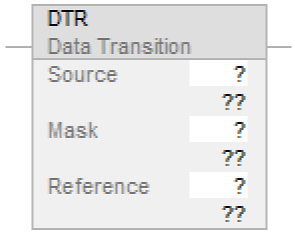
When enabled, the Mask passes data when the Mask bits are set; the Mask blocks data when the Mask bits are cleared.

When the masked Source differs from the Reference, the EnableOut goes true for one scan. When the masked Source is the same as the Reference, the EnableOut is false.

IMPORTANT: Online programming with this instruction can be dangerous. If the Reference value is different than the Source value, the EnableOut goes true. Use caution if you insert this instruction when the processor is in Run or Remote Run mode.

Available Languages

Ladder Diagram



Operands

Ladder Diagram

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

Entering an immediate mask value

When you enter a mask, the programming software defaults to decimal values. If you want to enter a mask using another format, precede the value with the correct prefix.

Prefix	Description
16#	hexadecimal (e.g., 16#0F0F)
8#	octal (e.g., 8#16)
2#	binary (e.g., 2#00110011)

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

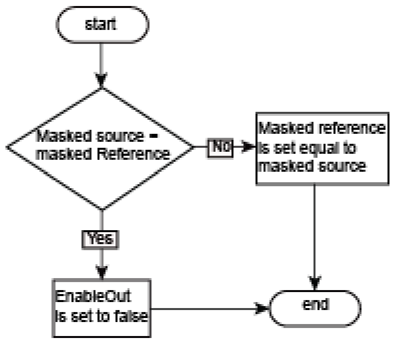
Execution

Ladder Diagram

Condition	Action
Prescan	The Reference = Source AND Mask.
Rung-condition-in is false	The Reference = Source AND Mask.
Rung-condition-in is true	See DTR Flow Chart (True)

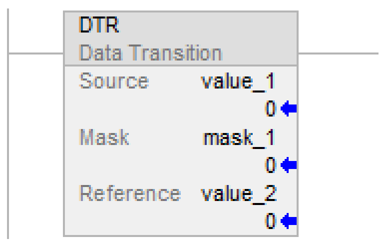
Condition	Action
Postscan	N/A

DTR Flow Chart (True)

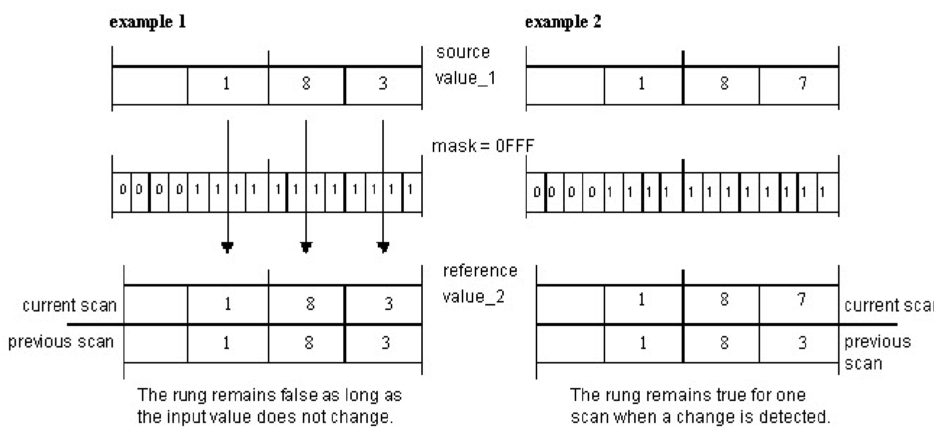


Example

Ladder Diagram



When enabled, the DTR instruction masks value_1. If there is a difference in the two masked values, the EnableOut is set to true.



In example 1, since reference value is equal to sourcevalue_1 AND mask, so the EnableOut will always set to false. In example 2, for some reason, the source value is changed, then reference_value is not equal to source_value AND mask, so in case of this, the EnableOut will be set to TRUE and the referencevalue will be updated based on the sourceValue and mask. That's why you see in previous scan the reference value is 183, but in current scan it is 187. The rung remains true only for one scan when a change is detected because in the next scan as long as source is not changed, the rung will remains false because the reference value will be equal to source value AND mask again.

Diagnostic Detect (DDT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The DDT instruction compares bits in a Source array with bits in a Reference array to find mismatch bit. The mismatch bit location is then recorded and the mismatch Reference bit is changed to match Source bit.

When enabled, the DDT instruction compares the bits in the Source array with the bits in the Reference array, records the bit number of each mismatch in the Result array, and changes the value of the Reference bit to match the value of the corresponding Source bit.

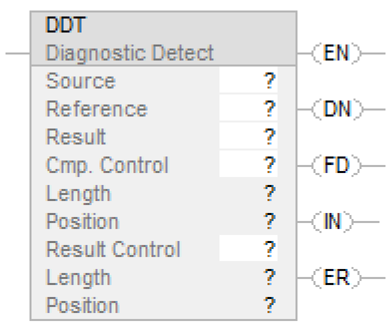
IMPORTANT: The DDT instruction operates on contiguous memory. You must test and confirm that the instruction does not change data that you don't want it to change.

The difference between the DDT and FBC instructions is that each time the DDT instruction finds a mismatch, the DDT instruction changes the reference bit to match the source bit. The FBC instruction does not change the reference bit.

If the instruction tries to read past the end of an array, the instruction sets the .ER bit and generates a major fault.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram

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

Operand	Type	Format	Description
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

IMPORTANT: Use different tags for the compare control structure and the result control structure. Using the same tag for both could result in unpredictable operation, possibly causing equipment damage and/or injury to personnel.

COMPARE Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the DDT instruction is enabled.
.DN	BOOL	The done bit is set when the DDT instruction compares the last bit in the Source and Reference arrays.
.FD	BOOL	The found bit is set each time the DDT 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 DDT search mode. 0 = all mode 1 = one mismatch at a time mode
.ER	BOOL	The error bit is either POS or LEN are invalid.
.LEN	DINT	The length value identifies the number of bits to compare.

Mnemonic	Data Type	Description
.POS	DINT	The position value identifies the current bit.

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.

Select the search mode

If you want to detect:	Select this mode:
One mismatch at a time	Set the .IN bit in the compare CONTROL structure. Each time the EnableIn goes from false to true, the DDT instruction searches for the next mismatch between the Source and Reference arrays. Upon finding a mismatch, the instruction stops, sets the .FD bit, and records the position of the mismatch.
All mismatches	Clear the .IN bit in the compare CONTROL structure. Each time the EnableIn goes from false to true, the DDT instruction searches for all mismatches between the Source and Reference arrays.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
result.POS > size of result array	4	20

See Common Attributes for General Instructions on page 849 for operand related faults.

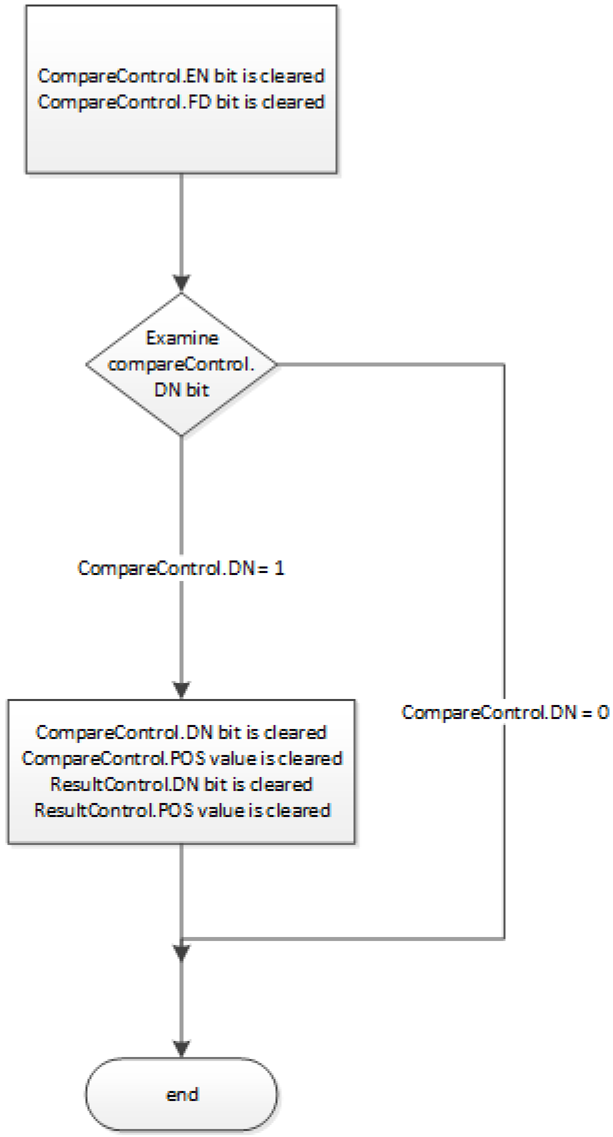
Execution

Ladder Diagram

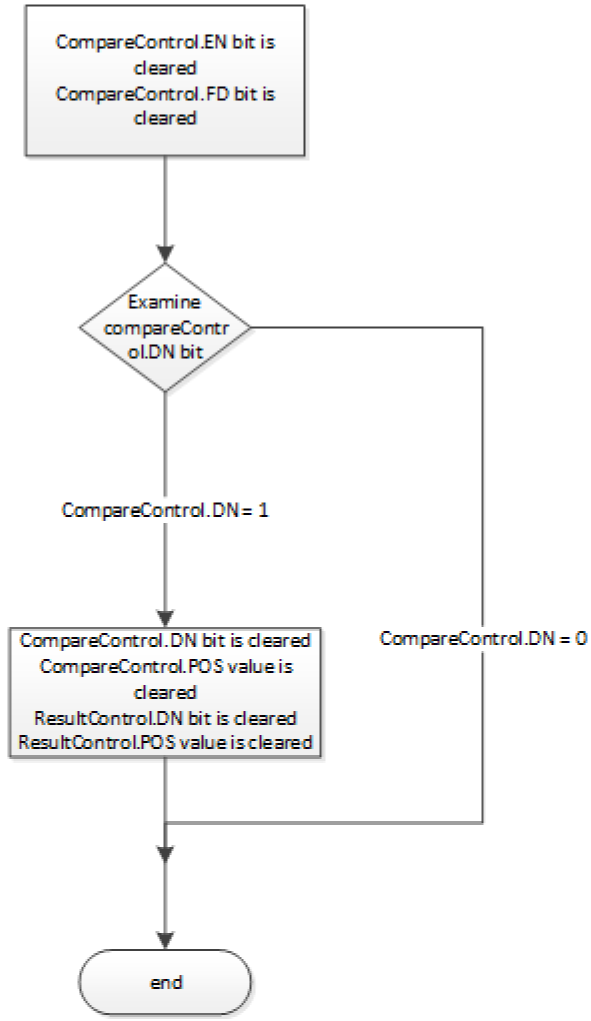
Condition/State	Action Taken
Prescan	See DDT Flow Chart (Prescan)
Rung-condition-in is false	See DDT Flow Chart (False)

Condition/State	Action Taken
Rung-condition-in is true	See DDT Flow Chart (True)
Postscan	N/A

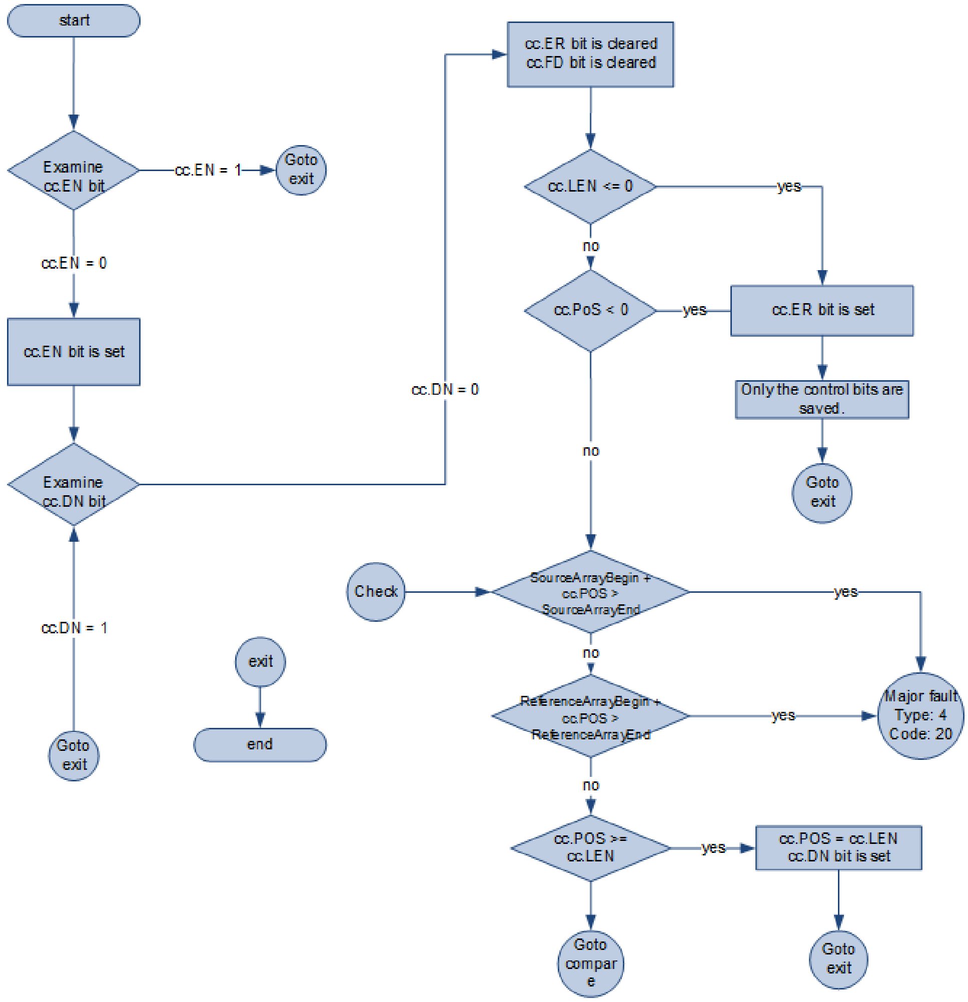
DDT Flow Chart (Prescan)



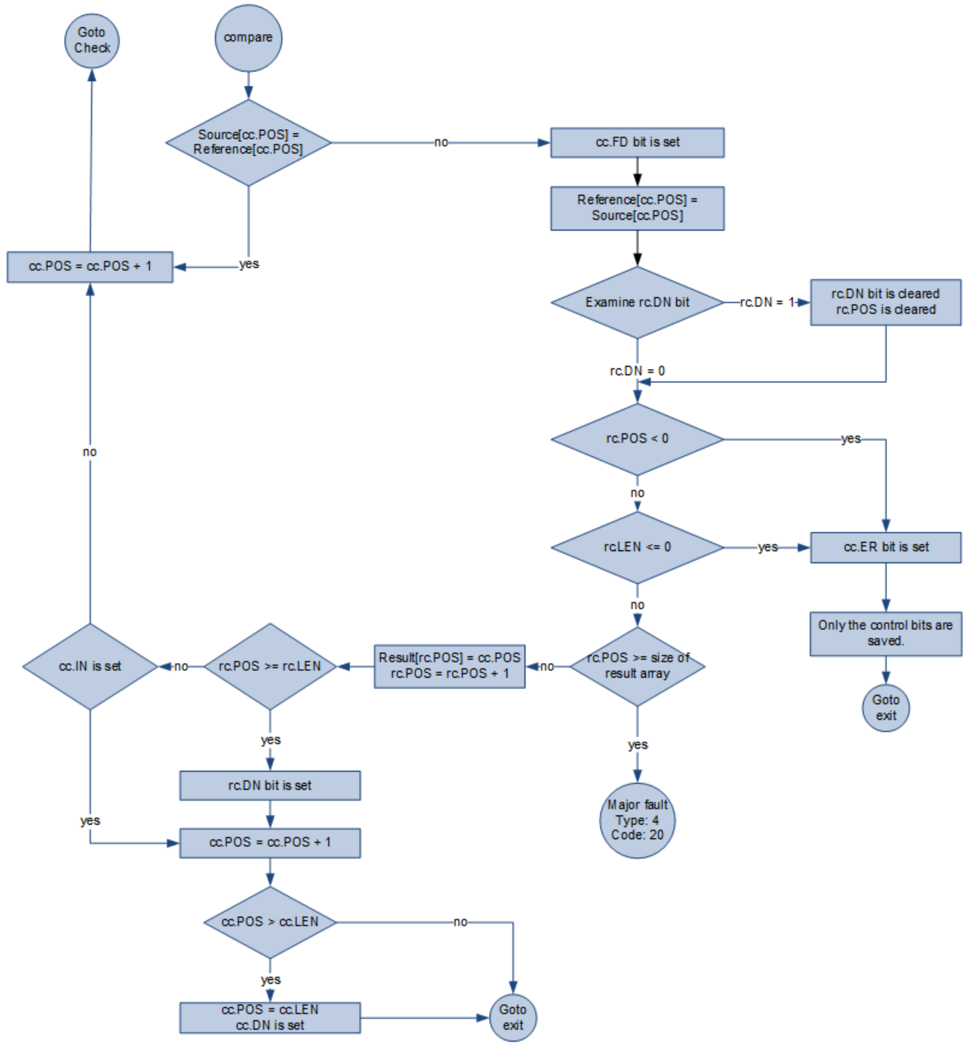
DDT Flow Chart (False)



DDT Flow Chart (True)

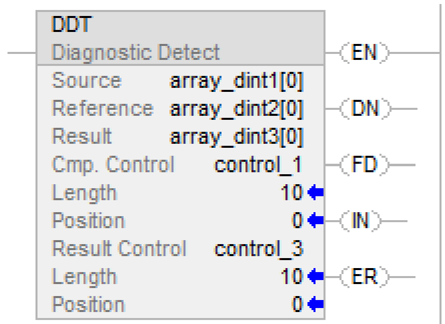


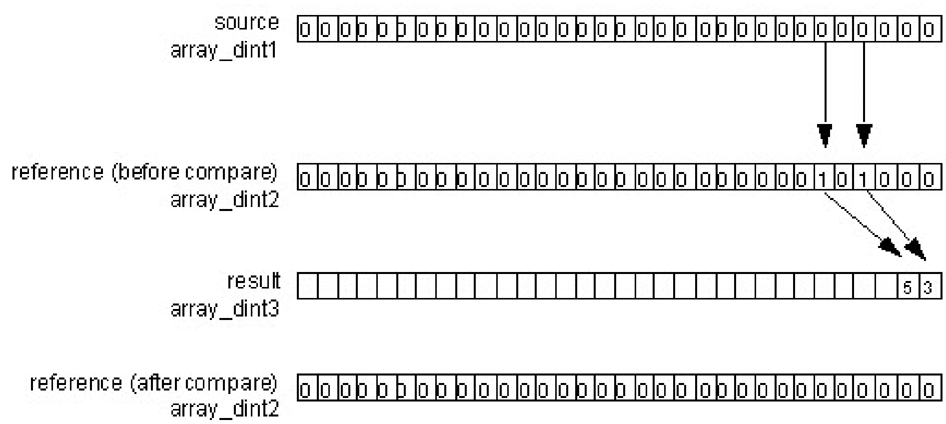
DDT Flow Chart (True) - Continued



Examples

Ladder Diagram





File Bit Comparison (FBC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The FBC instruction compares bits in a Source array with bits in a Reference array.

When enabled, the FBC instruction compares the bits in the Source array with the bits in the Reference array and records the bit number of each mismatch in the Result array.

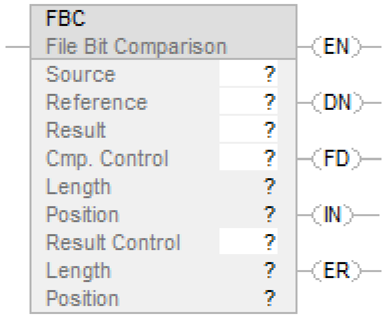
IMPORTANT: The FBC instruction operates on contiguous memory. You must test and confirm that the instruction doesn't change data that you don't want it to change.

The difference between the DDT and FBC instructions is that each time the DDT instruction finds a mismatch, the instruction changes the reference bit to match the source bit. The FBC instruction does not change the reference bit.

If the instruction tries to read past the end of an array, the instruction sets the .ER bit and generates a major fault.

Available Languages

Ladder Diagram



Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion on page 851.

Ladder Diagram

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

CAUTION: Use different tags for the compare control structure and the result control structure. Using the same tag for both could result in unpredictable operation, possibly causing equipment damage and injury to personnel.

COMPARE Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the FBC instruction is enabled.
.DN	BOOL	The done bit is set when the FBC instruction compares the last bit in the Source and Reference arrays.
.FD	BOOL	The found bit is set each time the FBC 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 FBC search mode. 0 = all mode 1 = one mismatch at a time mode
.ER	BOOL	The error bit is set either POS or LEN are invalid.
.LEN	DINT	The length value identifies the number of bits to compare.
.POS	DINT	The position value identifies the current bit.

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.

Select the search mode

If you want to detect:	Select this mode:
One mismatch at a time	Set the .IN bit in the compare CONTROL structure. Each time the EnableIn goes from false to true, the FBC instruction searches for the next mismatch between the Source and Reference arrays. Upon finding a mismatch, the instruction sets the .FD bit, records the position of the mismatch, and stops executing.
All mismatches	Clear the .IN bit in the compare CONTROL structure.

If you want to detect:	Select this mode:
	Each time EnableIn goes from false to true, the FBC instruction searches for all mismatches between the Source and Reference arrays.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if:	Fault type	Fault code
result.POS > size of result array	4	20

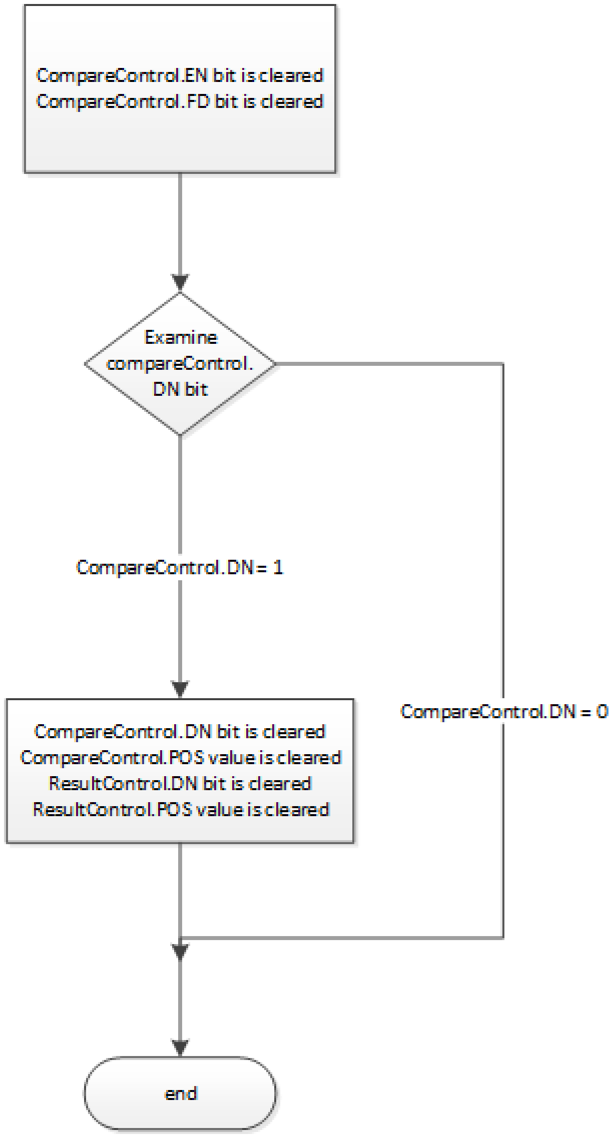
See Common Attributes for General Instructions on page 849 for operand related faults.

Execution

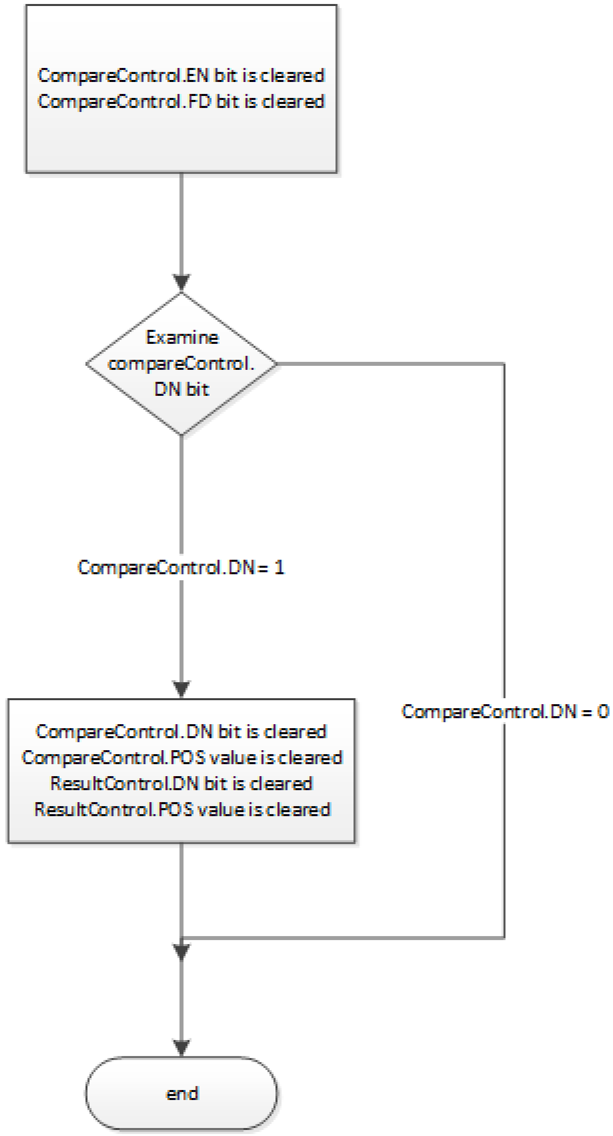
Ladder Diagram

Condition/State	Action Taken
Prescan	See FBC Flow Chart (Prescan)
Rung-condition-in is false	See FBC Flow Chart (False)
Rung-condition-in is true	See FBC Flow Chart (True)
Postscan	N/A

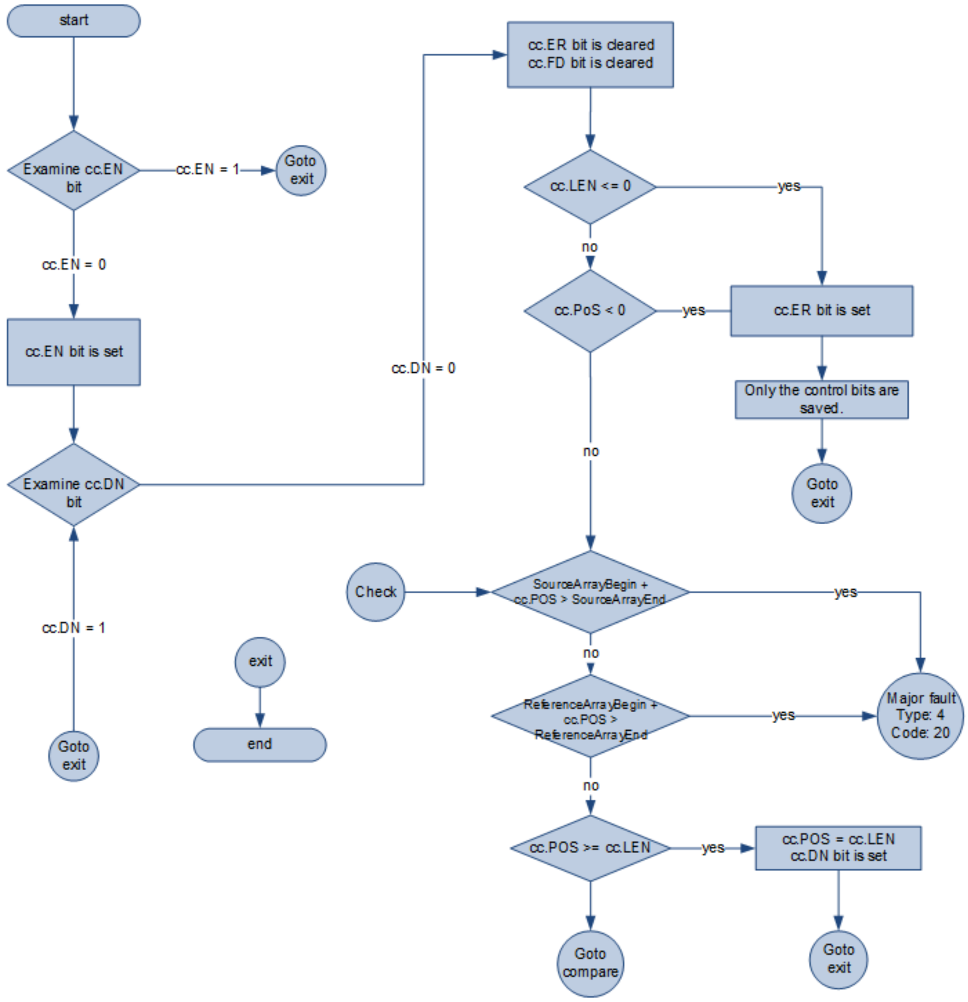
FBC Flow Chart (Prescan)



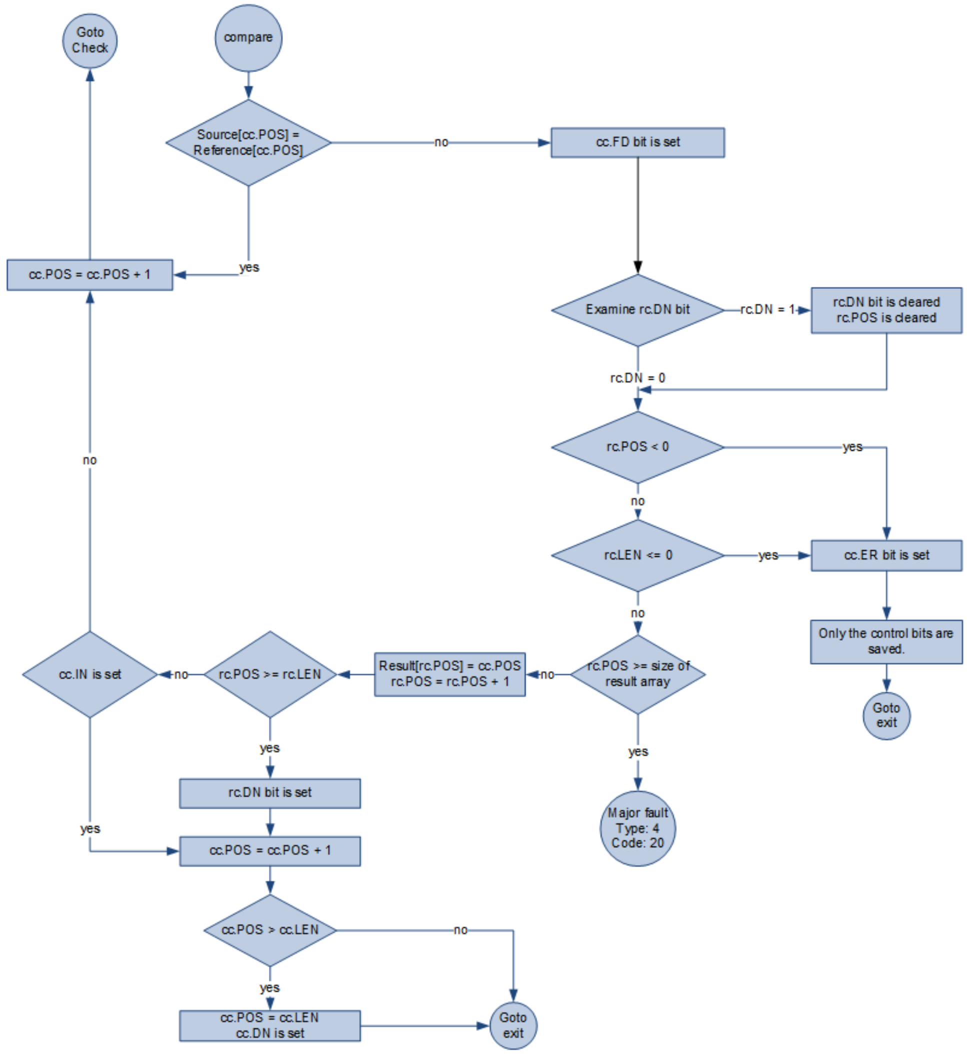
FBC Flow Chart (False)



FBC Flow Chart (True)

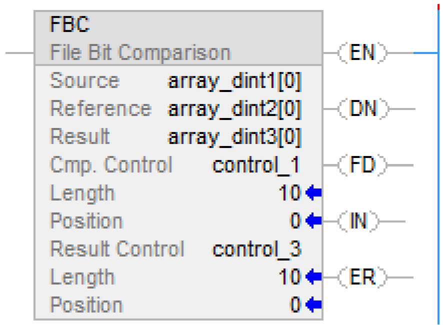


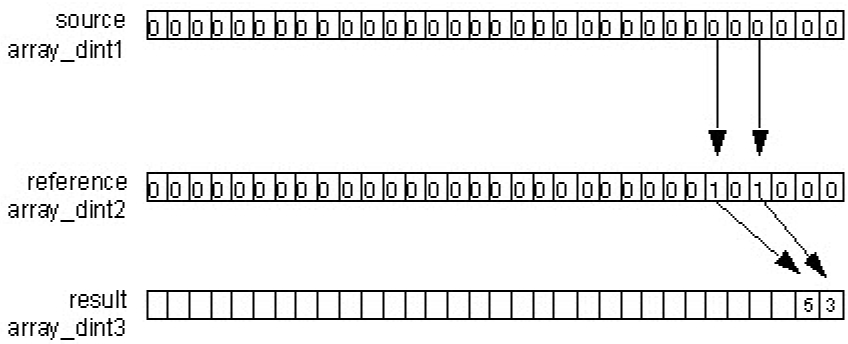
FBC Flow Chart (True) - continued



Example

Ladder Diagram





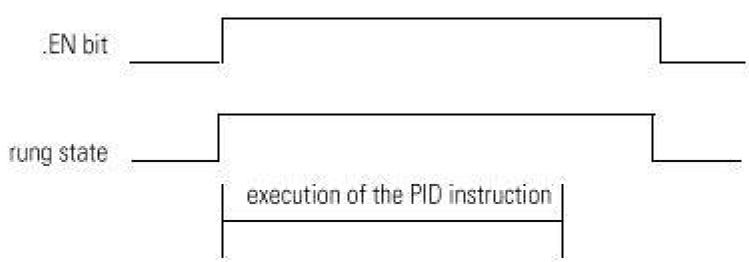
Proportional Integral Derivative (PID)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The PID instruction controls a process variable such as flow, pressure, temperature, or level.

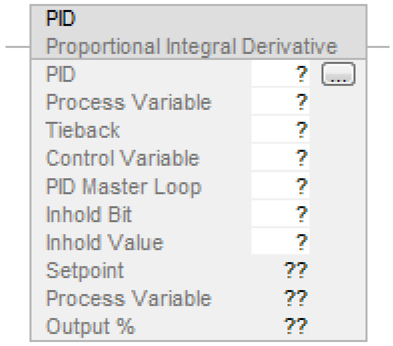
The PID instruction typically receives the process variable (PV) from an analog input module and modulates a control variable output (CV) on an analog output module in order to maintain the process variable at the desired setpoint.

The .EN bit indicates execution status. The .EN bit is set when the EnableIn transitions from false to true. The .EN bit is cleared when the EnableIn becomes false. The PID instruction does not use a .DN bit. The PID instruction executes every scan as long as the EnableIn is true.



Available Languages

Ladder Diagram



Structured Text

PID(PID,ProcessVariable,Tieback,ControlVariable,PIDMasterLoop,InHoldBit,InHoldValue);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Type	Format	Description
PID	PID	structure	PID structure
Process variable	SINT	tag	Value you want to control
	INT		
	DINT		
	REAL		
Tieback	SINT	immediate	(optional)
	INT	tag	
	DINT		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
	REAL		
Control variable	SINT	tag	Value which goes to the final control device (valve, damper, etc.)
	INT		
	DINT		If you are using the deadband, the Control variable must be REAL or it will be forced to 0

Operand	Type	Format	Description
			when the error is within the deadband.
	REAL		
PID master loop	PID	Structure	Optional
			PID tag for the master PID
			If you are performing cascade control and this PID is a slave loop, enter the name of the master PID
			Enter 0 if you do not 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
Inhold value	SINT	tag	Optional
	INT		Data readback value from a 1756 analog output
	DINT		Channel to support bumpless restart
	REAL		Enter 0 if you don't want to use this parameter
Setpoint			Display only
			Current value of the setpoint
Process variable			Display only
			Current value of the scaled Process_Variable
Output %			Display only
			Current output percentage value

PID structure

Specify a unique PID structure for each PID instruction.

Mnemonic	Data Type	Description
.CTL	DINT	The .CTL member provides access to the status members (bits) in one, 32-bit word. Bits 07-15 are set by the PID instruction. See .CTL member.

Mnemonic	Data Type	Description
.SP	REAL	setpoint
.KP	REAL	Independent - proportional gain (unitless)
		Dependent - controller gain (unitless)
.KI	REAL	Independent - integral gain (1/sec)
		Dependent - reset time (minutes per repeat)
.KD	REAL	Independent - derivative gain (seconds)
		Dependent - rate time (minutes)
.BIAS	REAL	feedforward or bias %
.MAXS	REAL	maximum engineering unit scaling value
.MINS	REAL	minimum engineering unit scaling value
.DB	REAL	deadband engineering units
.SO	REAL	set
.MAXO	REAL	maximum output limit (% of output)
.MINO	REAL	minimum output limit (% of output)
.UPD	REAL	loop update time (seconds)
.PV	REAL	scaled PV value
.ERR	REAL	scaled error value
.OUT	REAL	output %
.PVH	REAL	process variable high alarm limit
.PVL	REAL	process variable low alarm limit
.DVP	REAL	positive deviation alarm limit
.DVN	REAL	negative deviation alarm limit
.PVDB	REAL	process variable alarm deadband
.DVDB	REAL	deviation alarm deadband
.MAXI	REAL	maximum PV value (unscaled input)
.MINI	REAL	minimum PV value (unscaled input)
.TIE	REAL	tieback value for manual control
.MAXCV	REAL	maximum CV value (corresponding to 100%)
.MINCV	REAL	minimum CV value (corresponding to 0%)
.MINITIE	REAL	minimum tieback value (corresponding to 100%)
.MAXTIE	REAL	maximum tieback value (corresponding to 0%)

Mnemonic	Data Type	Description
.DATA[17]	REAL	The .DATA member stores: • .DATA[0] - integral accumulation • .DATA[1] - derivative smoothing temporary value • .DATA[2] - previous .PV value • .DATA[3] - previous .ERR value • .DATA[4] - previous valid .SP value • .DATA[5] - percent scaling constant • .DATA[6] - .PV scaling constant • .DATA[7] - derivative scaling constant • .DATA[8] - previous .KP value • .DATA[9] - previous .KI value • .DATA[10] - previous .KD value • .DATA[11] - dependend gain .KP • .DATA[12] - dependend gain .KI • .DATA[13] - dependend gain .KD • .DATA[14] - previous .CV value • .DATA[15] - .CV descaling constant • .DATA[16] - tieback descaling constant

The .CTL member

Bit	Number	Description
.EN	31	
.CT	30	cascade type (0=slave; 1=master)
.CL	29	cascade loop (0=no; 1=yes)
.PVT	28	process variable tracking (0=no; 1=yes)
.DOE	27	derivative of (0=PV; 1=error)
.SWM	26	software mode (0=no-auto); 1=yes- sw manual)
.CA	25	control action (0=reverse (SP-PV); 1=direct (PV- SP))
.MO	24	station mode (0=automatic; 1=manual)
.PE	23	PID equation (0=independent; 1=dependent)
.NDF	22	derivative smoothing (0=no; 1=yes)
.NOBC	21	bias calculation (0=no; 1=yes)
.NOZC	20	zero crossing (0=no; 1=for deadband)
.INI	15	PID initialized (0=no; 1=yes)

.SPOR	14	setpoint out of range (0=no; 1=yes)
.OLL	13	CV is below minimum output value (0=no; 1=yes)
.OLH	12	CV is above maximum output value (0=no; 1=yes)
.EWD	11	error is within deadband (0=no; 1=yes)
.DVNA	10	error is alarmed low (0=no; 1=yes)
.DVPA	9	error is alarmed high (0=no; 1=yes)
.PVLA	8	PV is alarmed low (0=no; 1=yes)
.PVHA	7	PV is alarmed high (0=no; 1=yes)

Affects Math Status Flags

No

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
UPD ≥ 0	4	35
setpoint out of range	4	36

See Common Attributes for General Instructions on page 849 for operand-related faults.

Using PID Instructions

After entering the PID instruction and specifying the PID structure, use the configuration tabs to specify how the PID instruction should function.

Specify Tuning

Select the **Tuning** tab. Changes take effect as soon as you select another field, select **OK**, select **Apply**, or press **Enter**.

In this field:	Do the following:
Setpoint (SP)	Enter a setpoint value (.SP).
Set output %	Enter a set output percentage (.SO). In software manual mode, this value is used for the output. In auto mode, this value displays the output %.
Output bias	Enter an output bias percentage (.BIAS).
Proportional gain (Kp)	Enter the proportional gain (.KP). For independent gains, it's the proportional gain (unitless). For dependent gains, it's the controller gain (unitless).
Integral gain (Ki)	Enter the integral gain (.KI). For independent gains, it's the integral gain (1/sec). For dependent gains, it's the reset time (minutes per repeat).

In this field:	Do the following:
Derivative time (Kd)	Enter the derivative gain (.KD). For independent gains, it's the derivative gain (seconds). For dependent gains, it's the rate time minutes).
Manual mode	Select either manual (.MO) or software manual (.SWM). Manual mode overrides software manual mode if both are selected.

Specify Configuration

Select the Configuration tab. You must select **OK** or **Apply** for any changes to take effect.

In this field:	Do the following:
PID equation	Select independent gains or dependent gains (.PE). Use independent when you want the three gains (P, I, and D) to operate independently. Use dependent when you want an overall controller gain that affects all three terms (P, I, and D).
Control action	Select either E=PV-SP or E=SP-PV for the control action (.CA).
Derivative of	Select PV or error (.DOE). Use the derivative of PV to reduce the risk of output spikes resulting from setpoint changes. Use the derivative of error for fast responses to setpoint changes when the algorithm can tolerate overshoots.
Loop update time	Enter the update time (.UPD) for the instruction.
CV high limit	Enter a high limit for the control variable (.MAXO).(1)
CV low limit	Enter a low limit for the control variable (.MINO).(1)
Deadband value	Enter a deadband value (.DB).
No derivative smoothing	Enable or disable this selection (.NDF).
No bias calculation	Enable or disable this selection (.NOBC).
No zero crossing in deadband	Enable or disable this selection (.NOZC).
PV tracking	Enable or disable this selection (.PVT).
Cascade loop	Enable or disable this selection (.CL).
Cascade type	If cascade loop is enabled, select either slave or master (.CT).

(1) When using the ladder-based PID instruction, if you set MAXO = MINO, the PID instruction resets these values to default. MAXO = 100.0 and MINO = 0.0

Specify Alarms

Select the **Alarms** tab. Select **OK** or **Apply** for any changes to take effect.

In this field:	Do the following:
PV high	Enter a PV high alarm value (.PVH).
PV low	Enter a PV low alarm value (.PVL).

In this field:	Do the following:
PV deadband	Enter a PV alarm deadband value (.PVDB).
Positive deviation	Enter a positive deviation value (.DVP).
Negative deviation	Enter a negative deviation value (.DVN).
Deviation deadband	Enter a deviation alarm deadband value (.DVDB).

Specify Scaling

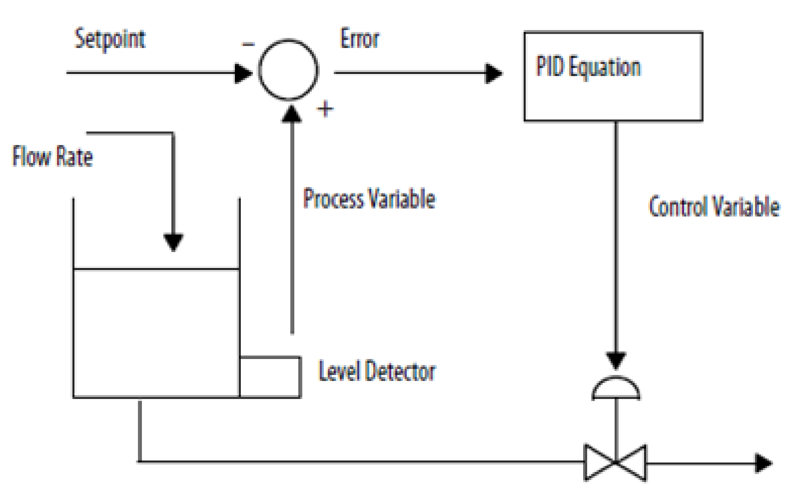
Select the Scaling tab. You must select **OK** or **Apply** for any changes to take effect.

In this field:	Do the following:
PV unscaled maximum	Enter a maximum PV value (.MAXI) that equals the maximum unscaled value received from the analog input channel for the PV value.
PV unscaled minimum	Enter a minimum PV value (.MINI) that equals the minimum unscaled value received from the analog input channel for the PV value.
PV engineering units maximum	Enter the maximum engineering units corresponding to .MAXI (.MAXS)
PV engineering units minimum	Enter the minimum engineering units corresponding to .MINI (.MINS)
CV maximum	Enter a maximum CV value corresponding to 100% (.MAXCV).
CV minimum	Enter a minimum CV value corresponding to 0% (.MINCV).
Tieback maximum	Enter a maximum tieback value (.MAXTIE) that equals the maximum unscaled value received from the analog input channel for the tieback value.
Tieback minimum	Enter a minimum tieback value (.MINTIE) that equals the minimum unscaled value received from the analog input channel for the tieback value.
PID Initialized	If you change scaling constants during Run mode, turn this off to reinitialize internal descaling values (.INI).

Tip: When using the ladder-based PID instruction, if you set MAXO = MINO, the PID instruction resets these values to default. MAXO = 100.0 and MINO = 0.0.

Use PID Instructions

PID closed-loop control holds a process variable at a desired set point. The illustration shows an example of a flow-rate/fluid level.



In the above example, the level in the tank is compared against the setpoint. If the level is higher than the setpoint, the PID equation increases the control variable and causes the outlet valve from the tank to open; thereby decreasing the level in the tank.

The PID equation used in the PID instruction is a positional form equation with the option of using either independent gains or dependent gains. When using independent gains, the proportional, integral, and derivative gains affect only their specific proportional, integral, or derivative terms respectively. When using dependent gains, the proportional gain is replaced with a controller gain that affects all three terms. You can use either form of equation to perform the same type of control. The two equation types are merely provided to let you use the equation type with which you are most familiar.

Gains Option	**Derivative Of**
Dependent gains (ISA standard)	Error (E)
	Process variable (PV)
Independent gains	Error (E)
	Process variable (PV)

Where:

Variable	**Description**
KP	Proportional gain (unitless) Kp = Kc unitless
Ki	Integral gain (seconds -1) To convert between Ki (integral gain) and Ti (reset time), see Conversion Formula:
Kd	Derivative gain (seconds) To convert between Kd (derivative gain) and Td (rate time), use: Kd = Kc (Td) 60
KC	Controller gain (unitless)
Ti	Reset time (minutes/repeat)

Variable	Description
Td	Rate time (minutes)
SP	Setpoint
PV	Process variable
E	Error [(SP-PV) or (PV-SP)]
BIAS	Feedforward or bias
CV	Control variable
dt	Loop update time

Conversion Formula

$$K_i = \frac{K_C}{60\,T_i}$$

If you do not want to use a particular term of the PID equation, just set its gain to zero. For example if you want no derivative action, set Kd or Td equal to zero.

Anti-reset Windup and Bumpless Transfer From Manual To Auto (PID)

The PID instruction automatically avoids reset windup by preventing the integral term from accumulating whenever the CV output reaches its maximum or minimum values, as set by .MAXO and .MINO. The accumulated integral term remains frozen until the CV output drops below its maximum limit or rises above its minimum limit. Then normal integral accumulation automatically resumes.

The PID instruction supports two manual modes of control.

Manual Mode of Control	Description
Software manual (.SWM)	This mode is also known as set output mode and allows the user to set the output % from the software. The set output (.SO) value is used as the output of the loop. The set output value typically comes from an operator input from an operator interface device.
Manual (.MO)	This mode takes the tieback value, as an input, and adjusts its internal variables to generate the same value at the output The tieback input to the PID instruction is scaled to 0-100% according to the values of .MINTIE and .MAXTIE and is used as the output of the loop. The tieback input typically comes from the output of a hardware hand/auto station that is bypassing the output from the controller. **Important:** Manual mode overrides software manual mode if both mode bits are set on.

The PID instruction automatically provides bumpless transfers from software manual mode to auto mode or from manual to auto mode. The PID instruction back-calculates the value of the integral accumulation term required to make the CV output track either the set output (.SO) value in software manual mode or the tieback input in manual mode. In this manner, when the loop switches to auto mode, the CV output starts off from the set output or tieback value and no 'bump' in output value occurs.

The PID instruction can also automatically provide a bumpless transfer from manual to auto even if integral control is not used (that is Ki = 0). In this case, the instruction modifies the .BIAS term to make the CV output track either the set output or tieback values. When automatic control is resumed, the .BIAS term maintains its last value. Disable back-calculation of the .BIAS term by setting the .NOBC bit in the PID data structure. If you set .NOBC true, the PID instruction no longer provides a bumpless transfer from manual to auto when integral control is not used.

Bumpless Restart (PID)

The PID instruction can interact with the 1756 analog output modules to support a bumpless restart when the controller changes from Program to Run mode or when the controller powers up.

When a 1756 analog output module loses communications with the controller or senses that the controller is in Program mode, the analog output module sets its outputs to the fault condition values you specified when you configured the module. When the controller then returns to Run mode or re-establishes communications with the analog output module, you can have the PID instruction automatically reset its control variable output equal to the analog output by using the Inhold bit and Inhold Value parameters on the PID instruction.

Instructions for setting a bumpless restart

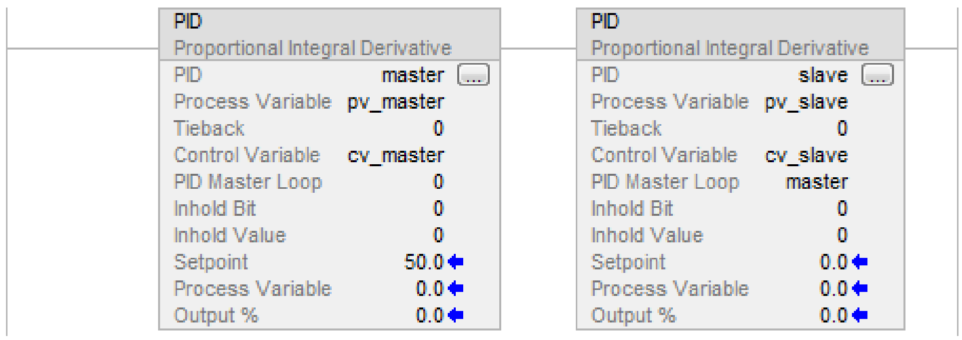
Do this	Details
Configure the channel of the 1756 analog output module that receives the control variable from the PID instruction	Select the **Hold for initialization** box on the properties page for the specific channel of the module. This tells the analog output module that when the controller returns to Run mode or re-establishes communications with the module, the module should hold the analog output at its current value until the value sent from the controller matches (within 0.1% of span) the current value used by the output channel. The output of of the channel ramps to the currently held output value by making use of the .BIAS term. This ramping is similar to auto bumpless transfer.
Enter the Inhold bit tag and Inhold Value tag in the PID instruction	The 1756 analog output module returns two values for each channel in its input data structure. The InHold status bit (.Ch2InHold, for example), when true, indicates that the analog output channel is holding its value. The Data readback value (.Ch2Data, for example) shows the current output value in engineering units. Enter the tag of the InHold status bit as the InHold bit parameter of the PID instruction. Enter the tag of the Data readback value as the Inhold Value parameter. When he Inhold bit is true, the PID instruction moves the Inhold Value into the Control variable output and re-initializes to support a bumpless restart at that value. When the analog output module receives this value back from the controller, it

Do this	Details
	turns off the InHold status bit, which allows the PID instruction to start controlling normally.

Cascading Loops (PID)

The PID cascades two loops by assigning the output in percent of the master loop to the setpoint of the slave loop. The slave loop automatically converts the output of the master loop into the correct engineering units for the setpoint of the slave loop, based on the slave loop's values for .MAXS and .MINS.

Ladder Diagram

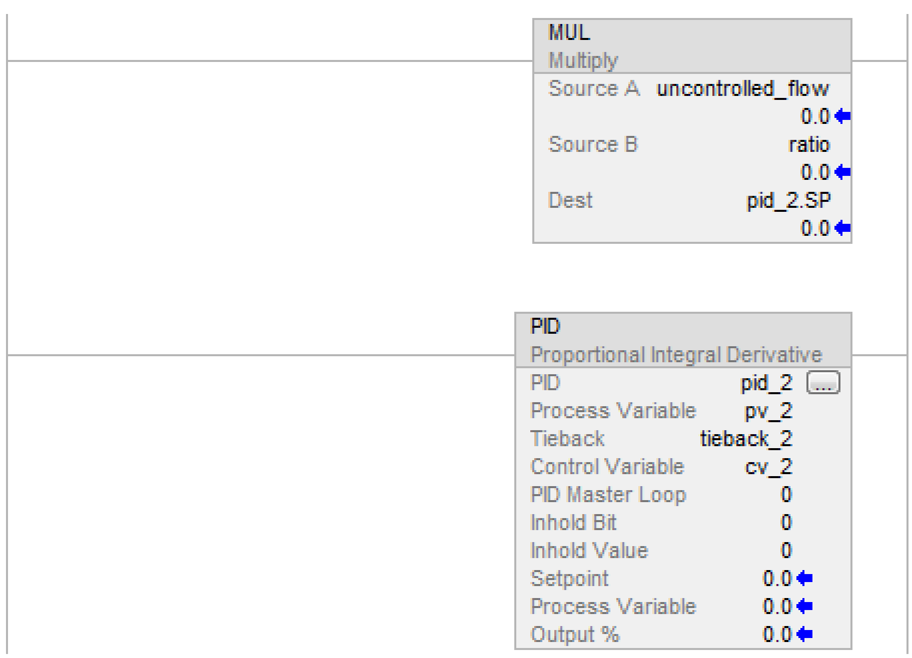


Controlling a Ratio (PID)

You can maintain two values in a ratio by using these parameters:

- Uncontrolled value
- Controlled value (the resultant setpoint to be used by the PID instruction)
- Ratio between these two values

Ladder Diagram



Tip: Tip: To avoid locking up the PID with invalid internal floating point values, ensure the PV is not INF or NAN before invoking the instruction such as:

XIC (PC_timer.DN)

MOV(Local:0:1.Ch0Data, Local:0:1.Ch0Data)

XIO(S:V)

PID(...)

Structured Text

pid_2.sp := uncontrolled_flow * ratio

PID(pid_2,pv_2,tieback_2,cv_2,0,0,0);

Tip: Tip: To avoid locking up the PID with invalid internal floating point values, ensure the PV is not INF or NAN before invoking the instruction such as:

XIC (PC_timer.DN)

MOV(Local:0:1.Ch0Data, Local:0:1.Ch0Data)

XIO(S:V)

PID(...)

For this multiplication	Enter this value
Destination	Controlled value
Source A	Uncontrolled value
Source B	Ratio

Derivative Smoothing (PID)

The derivative calculation is enhanced by a derivative smoothing filter. This first order, low pass, digital filter minimizes large derivative term spikes caused by noise in the PV. This smoothing becomes more aggressive with larger values of derivative gain. You can disable derivative smoothing if your process requires very large values of derivative gain (Kd > 10, for example).

To disable derivative smoothing:

- Select **No derivative smoothing** on the **Configuration** tab, or set the .NDF bit in the PID structure.

Feedforward or Output Biasing (PID)

Feedforward a disturbance from the system by feeding the .BIAS value into the PID instruction’s feedforward/bias value.

The feedforward value represents a disturbance fed into the PID instruction before the disturbance has a chance to change the process variable. Feedforward is often used to control processes with a transportation lag. For example, a feedforward value representing 'cold water poured into a warm mix' could boost the output value faster than waiting for the process variable to change as a result of the mixing.

A bias value is typically used when no integral control is used. In this case, the bias value can be adjusted to maintain the output in the range required to keep the PV near the setpoint.

PID Instruction Timing

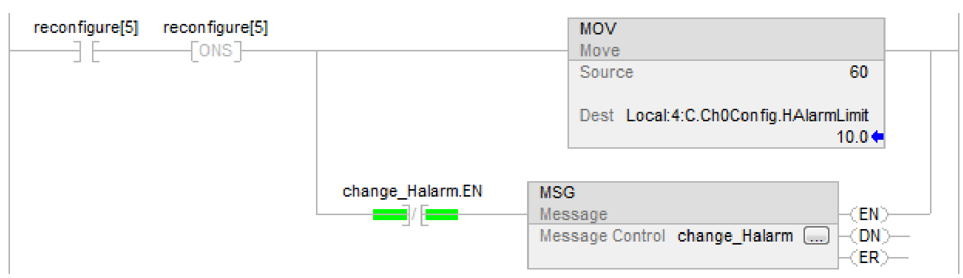
The PID instruction and the sampling of the process variable need to be updated at a periodic rate. This update time is related to the physical process you are controlling. For very slow loops, such as temperature loops, an update time of once per second or even longer is usually sufficient to obtain good control. Somewhat faster loops, such as pressure or flow loops, may require an update time such as once every 250 ms. Only rare cases, such as tension control on an unwinder spool, require loop updates as fast as every 10 ms or faster.

Because the PID instruction uses a time base in its calculation, you need to synchronize execution of this instruction with sampling of the process variable (PV).

The easiest way to execute the PID instruction is to put the PID instruction in a periodic task. Set the loop update time (.UPD) equal to the periodic task rate and make sure that the PID instruction is executed every scan of the periodic task.

The easiest way to execute the PID instruction is to put the PID instruction in a periodic task. Set the loop update time (.UPD) equal to the periodic task rate and make sure that the PID instruction is executed every scan of the periodic task.

Relay Ladder



Tip: To avoid locking up the PID with invalid internal floating point values, ensure the PV is not INF or NAN before invoking the instruction such as:

XIC (PC_timer.DN)

MOV(Local:0:1.Ch0Data, Local:0:1.Ch0Data)

XIO(S:V)

PID(...)

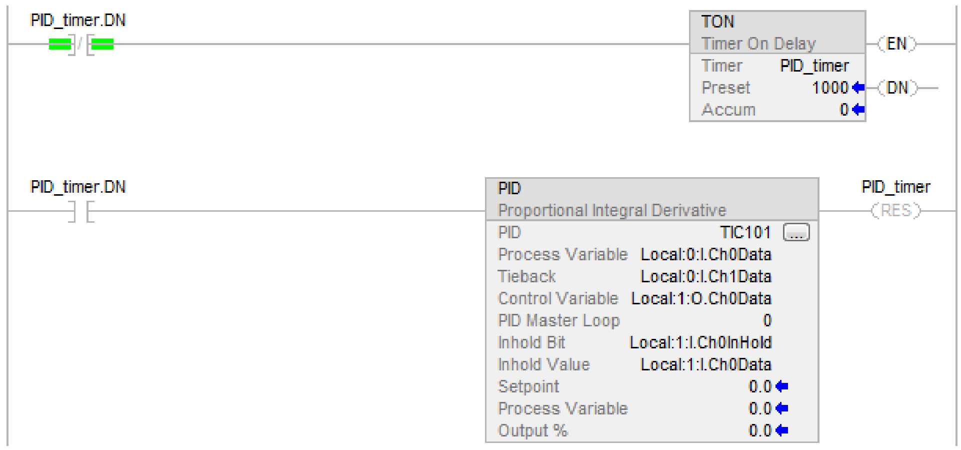
Structured Text

PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data, Local:1:O.Ch4Data,0,Local:1:I.Ch4InHold, Local:1:I.Ch4Data);

When using a periodic task, make sure that the analog input used for the process variable is updated to the processor at a rate that is significantly faster than the rate of the periodic task. Ideally, the process variable should be sent to the processor at least five to 10 times faster than the periodic task rate. This minimizes the time difference between actual samples of the process variable and execution of the PID loop. For example, if the PID loop is in a 250 ms periodic task, use a loop update time of 250 ms (.UPD = .25), and configure the analog input module to produce data at least about every 25 to 50 ms.

Another, somewhat less accurate, method of executing a PID instruction is to place the instruction in a continuous task and use a timer done bit to trigger execution of the PID instruction.

Relay Ladder



Tip: To avoid locking up the PID with invalid internal floating point values, ensure the PV is not INF or NAN before invoking the instruction such as:

XIC (PC_timer.DN)

MOV(Local:0:1.Ch0Data, Local:0:1.Ch0Data)

XIO(S:V)

PID(...)

Structured Text

PID_timer.pre := 1000

TONR(PID_timer);

IF PID_timer.DN THEN PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,

Local:1:O.Ch0Data,0,Local:1:I.Ch0InHold,

Local:1:I.Ch0Data);

END_IF;

Tip: To avoid locking up the PID with invalid internal floating point values, ensure the PV is not INF or NAN before invoking the instruction such as:

XIC (PC_timer.DN)

MOV(Local:0:1.Ch0Data, Local:0:1.Ch0Data)

XIO(S:V)

PID(...)

In this method, the loop update time of the PID instruction should be set equal to the timer preset. As in the case of using a periodic task, you should set the analog input module to produce the process variable at a significantly faster rate than the loop update time. You should only use the timer method of PID execution for loops with loop update times that are at least several times longer than the worst-case execution time for your continuous task.

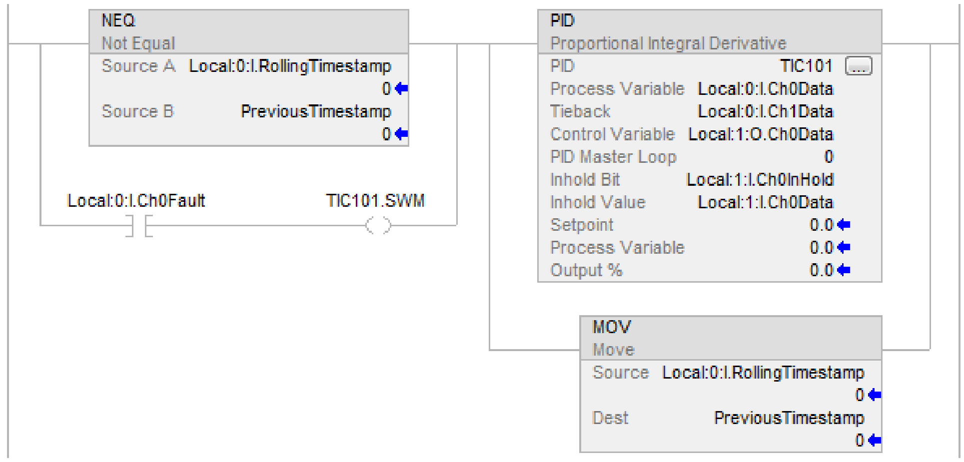
The most accurate way to execute a PID instruction is to use the real time sampling (RTS) feature of the 1756 analog input modules. The analog input module samples its inputs at the real time sampling rate you configure when you set up the module. When the real time sample period of the module expires, it updates its inputs and updates a rolling timestamp (represented by the .RollingTimestamp member of the analog input data structure) produced by the module.

The timestamp ranges from 0 to 32,767 ms. Monitor the timestamp. When it changes, a new process variable sample has been received. Every time a timestamp changes, execute the PID instruction once. Because the process variable sample is driven by the analog input module, the input sample time is very accurate, and the loop update time used by the PID instruction should be set equal to the RTS time of the analog input module.

To make sure that you do not miss samples of the process variable, execute your logic at a rate faster than the RTS time. For example, if the RTS time is 250 ms, you could put the PID logic in a periodic task that runs every100 ms to make sure that you never miss a sample. You could even place the PID logic in a continuous task, as long as you make sure that the logic would be updated more frequently than once every 250 ms.

An example of the RTS method of execution is shown below. The execution of the PID instruction depends on receiving new analog input data. If the analog input module fails or is removed, the controller stops receiving rolling timestamps and the PID loop stops executing. You should monitor the status bit of the PV analog input and, if it shows bad status, force the loop into software manual mode, and execute the loop every scan. This lets the operator still manually change the output of the PID loop.

Relay Ladder



Structured Text

IF (Local:0:I.Ch0Fault) THEN TIC101.SWM [:=] 1;

ELSE TIC101.SWM := 0; END_IF;

IF (Local:0:I.RollingTimestamp<>PreviousTimestamp) OR (Local:0:I.Ch0Fault) THEN

PreviousTimestamp := Local:0:I.RollingTimestamp; PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,

Local:1:0.Ch0Data,0,Local:1:I.Ch0InHold,

Local:1:I.Ch0Data);

END_IF;

Setting the Deadband (PID)

The adjustable deadband lets you select an error range above and below the setpoint where output does not change as long as the error remains within this range. This deadband allows you to control how closely the process variable matches the setpoint without changing the output. The deadband also helps to minimize wear and tear on your final control device.

Zero-crossing is deadband control that lets the instruction use the error for computational purposes as the process variable crosses into the deadband until the process variable crosses the setpoint. Once the process variable crosses the setpoint (error crosses zero and changes sign) and as long as the process variable remains in the deadband, the output does not change.

The deadband extends above and below the setpoint by the value you specify. Enter zero to inhibit the deadband. The deadband has the same scaled units as the setpoint. Use the deadband without the zero-crossing feature by selecting **No zero crossing for deadband** on the **Configuration** tab or set the .NOZC bit in the PID structure.

If you are using the deadband, the Control variable must be REAL or it is forced to zero when the error is within the deadband.

To inhibit the deadband:

- Enter zero (0).

The deadband has the same scaled units as the setpoint.

To use the deadband without the zero-crossing feature:

- Select **No zero crossing for deadband** on the **Configuration** tab or set the .NOZC bit in the PID structure.

If you are using the deadband, the Control variable must be REAL or it is forced to 0 when the error is within the deadband.

Using Output Limiting (PID)

Set an output limit (percentage of output) on the control output. When the instruction detects that the output has reached a limit, it sets an alarm bit and prevents the output from exceeding either the lower or upper limit.

Trigonometric Instructions

The trigonometric instructions evaluate arithmetic operations using trigonometric operations.

Available Instructions

Ladder Diagram, Function Block, and Structured Text

SIN on page 716	ATAN on page 702, ATAN2 on page 702	COS on page 711	TAN on page 721	ASIN on page 697	ACOS on page 691

If you want to:	Use this instruction:
Take the sine of a value.	SIN
Take the cosine of a value.	COS
Take the tangent of a value.	TAN
Take the arc sine of a value.	ASIN
Take the arc cosine of a value.	ACOS
Take the arc tangent of a value.	ATAN
Take the two-argument arc tangent of a value.	ATAN2

You can mix data types, but loss of accuracy and rounding error might occur and the instruction takes more time to execute. Check the S:V bit to see whether the result was truncated.

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

A trigonometric instruction executes once each time the instruction is scanned as long as the rung-condition-in is true. If you want the instruction evaluated only once, use an ONS instruction to trigger the trigonometric instruction.

Arc Cosine (ACOS)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

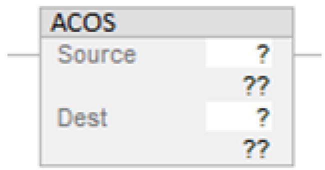
When enabled, the ACOS instruction takes the arc cosine of the Source value and stores the result in the Destination (in radians).

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from ACS to ACOS.

Available Languages

These are the available languages for Arc Cosine (ACOS).

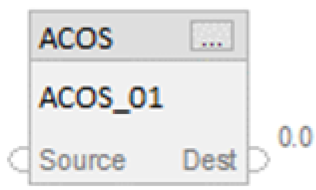
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use ACOS as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structured operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to arc cosine.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Block

Operand	Type	Format	Description
ACOS	FBD_MATH_ADVANCED	tag	ACS structure

FBD_MATH_ADVANCED Structure

Input Member	Data Type	Description

EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the trigonometric instruction.

Output Member	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to arc cosine.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Operator Aspects

The ACOS operator can be used in various RLL expressions. Similarly, the ACOS function is invoked in Structured Text statements. ACOS returns a floating point result containing the arc cosine of the Source. Depending on the context this value may then be type converted if appropriate.

Description

The ACOS instruction takes the arc cosine of the Source value and stores the result in the Destination (in radians). The ACOS operator/function computes the arc cosine of the Source and returns the floating point result. The Source must be greater than or equal to -1 and less than or equal to 1. The resulting value in the Destination is greater than or equal to 0 or less than or equal to pi (where pi = 3.141593). If Source is smaller than -1 or greater than 1 then Destination is set to NAN.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = arc cosine value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false.	Set EnableOut to EnableIn.
EnableIn is true	Dest = arc cosine value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.

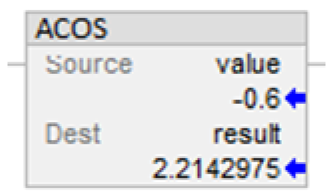
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

FBD Function

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = arc cosine value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

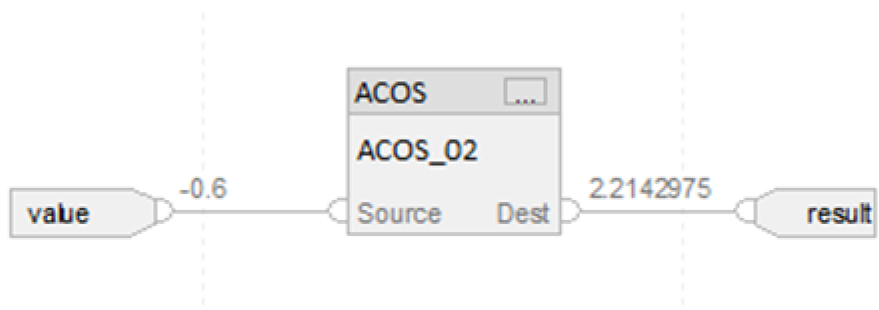
Example

Ladder Diagram

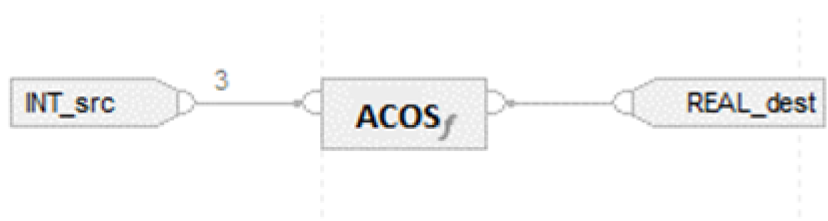


Function Block Diagram

FBD Block



FBD Function



Structured Text

REAL_dest := ACOS(REAL_src);

Arc Sine (ASIN)

This table lists the controllers and applications that support this instruction.

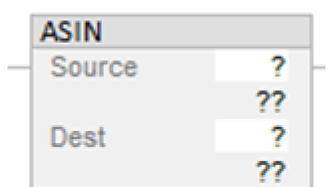
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the Arc Sine (ASIN) instruction takes the arc sine of the Source value and stores the result in the Destination (in radians). The ASIN operator/function computes the arc sine of the Source and returns the floating point result. The Source must be greater than or equal to -1 and less than or equal to 1. The resulting value in the Destination is greater than or equal to -pi /2 and less than or equal to pi /2 (where pi = 3.141593). If Source is smaller than -1 or greater than 1 then Destination is set to NAN.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from ASN to ASIN.

Available Languages

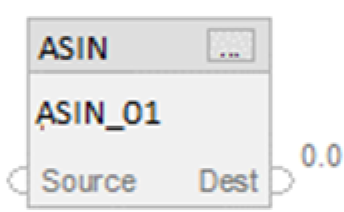
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use ASIN as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structured operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to arc sine
Destination	SINT INT DINT	SINT INT DINT	tag	Tag to store result of the instruction

	REAL	LINT USINT UINT UDINT ULINT REAL LREAL		

Function Block Diagram

FBD Block

Operand	Type	Format	Description
ASN	FBD_MATH_ADVANCED	tag	ASN structure

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to arc sine.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

FBD_MATH_ADVANCED Structure

Input Member	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the trigonometric instruction

Output Member	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

Operator Aspects

The ASIN operator can be used in various RLL expressions. Similarly, the ASIN function is invoked in Structured Text statements. ASIN returns a floating point result containing the arc sine of the Source. Depending on the context this value may then be type converted if appropriate.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = arc sine value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = arc sine value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

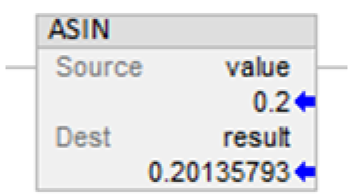
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = arc since value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

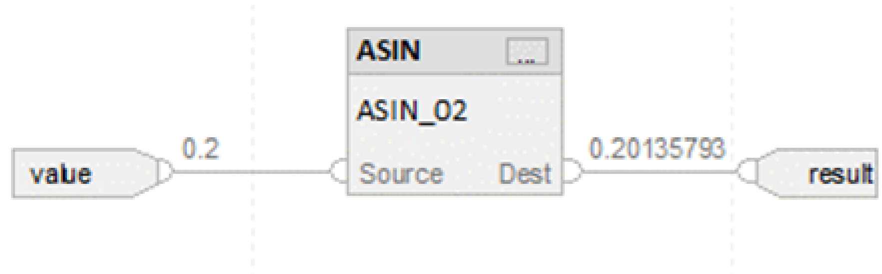
Example

Ladder Diagram

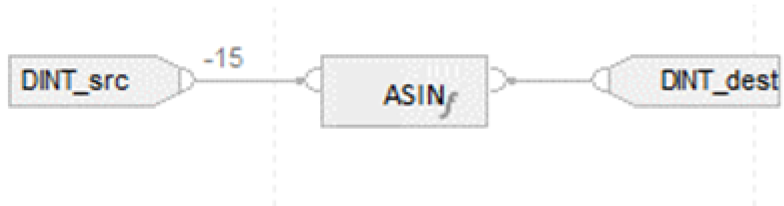


Function Block Diagram

FBD Block



FBD Function



Structured Text

REAL_dest := ASIN(REAL_src);

Arc Tangent (ATAN)

This table lists the controllers and applications that support this instruction.

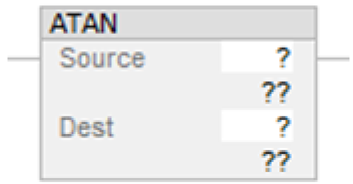
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the ATAN instruction computes the arc tangent of the Source value and stores the result in the Destination (in radians). The ATAN operator/function computes the arc tangent of the Source and returns the floating point result. The resulting value in the Destination is greater than or equal to -pi/2 and less than or equal to pi/2 (where pi = 3.141593).

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from ATN to ATAN.

Available Languages

Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block

FBD Function

Structured Text

This instruction is not available in structured text.

Tip: Use ATAN as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description

Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to arc tangent.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

Function Block

Operand	Type	Format	Description
ATAN tag	FBD_MATH_ADVANCED	tag	ATAN structure

FBD_MATH_ADVANCED Structure

Input Member	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the trigonometric instruction.

Output Member	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to arc tangent.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Operator Aspects

The ATAN operator can be used in various RLL expressions. Similarly, the ATAN function is invoked in Structured Text statements. ATAN returns a floating point result containing the arc tangent of the Source. Depending on the context this value may then be type converted if appropriate.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = arc tangent value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = arc tangent value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

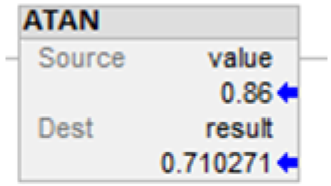
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = arc tangent value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

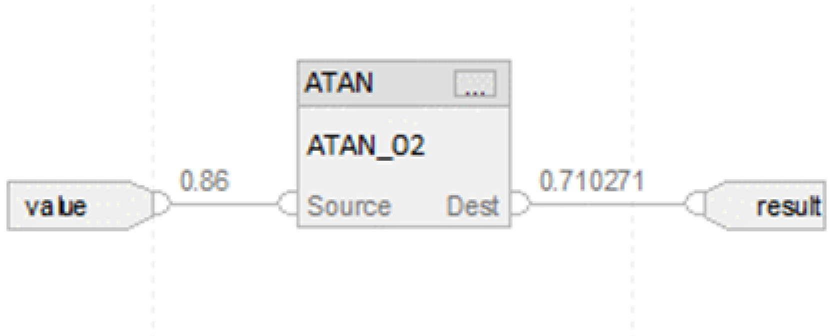
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := ATAN(REAL_src);

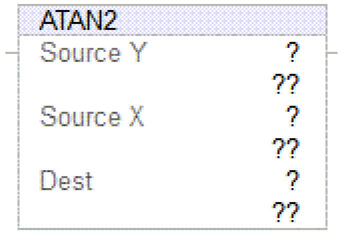
Two-Argument Arctangent (ATAN2)

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The ATAN2 instruction takes the two-argument arc tangent of the Source values and stores the result in the Destination (in radians). The ATAN2 operator/function computes the arc tangent of the Source and returns the FLOAT result. The resulting value in the Destination is greater than or equal to -p and less than or equal to p (where p = 3.141593).

Available Languages

Ladder Diagram



Function Block Diagram

Function Block Diagram supports only the FBD function:

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use ATAN2 as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structured operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

Ladder Diagram

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source Y	SINT	immediate	Source Y of ATAN2 Input

	INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	
Source X	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	Source X of ATAN2 Input
Dest	REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Function

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source Y	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Source Y of ATAN2 Input
Source Y	SINT INT DINT LINT USINT	Source X of ATAN2 Input

	UINT UDINT ULINT REAL LREAL	

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Operator Aspects

The ATAN2 operator can be used in various RLL expressions. Similarly, the ATAN2 function is invoked in Structured Text statements. ATAN2 returns a FLOAT result containing the result of two argument arc tangent of the Source Y and Source X. Depending on the context, this value can then be type converted if appropriate.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = ATAN2(Source Y, Source X)
Postscan	N/A

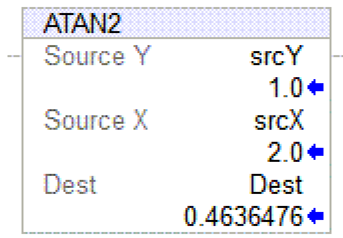
Function Block Diagram

FBD Function

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = ATAN2(Source Y, Source X)
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Examples

Ladder Diagram



Function Block Diagram

FBD Function

Structured Text

REAL_dest := ATAN2(REAL_srcY, REAL_srcX);

Cosine (COS)

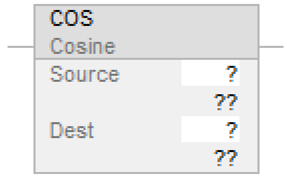
This table lists the controllers and applications that support this instruction.

Architecture	**Standard applications**	**Safety applications**
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The COS instruction takes the cosine of the Source value (in radians) and stores the result in the Destination. The COS operator/function computes the cosine of Source and returns the floating point result. The resulting value is always greater than or equal to -1 and less than or equal to 1.

Available Languages

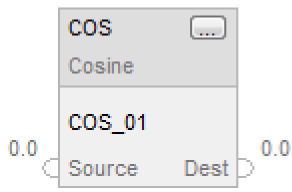
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use COS as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT	SINT INT	Immediate tag	Find the cosine of this value.

	DINT REAL	DINT LINT USINT UINT UDINT ULINT REAL LREAL		
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result.

Function Block Diagram

FBD Block

Operand	Type	Format	Description
COS tag	FBD_MATH_ADVANCED	tag	COS structure

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type	Description
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to cosine.

Output Operand (Right Pin)	Data Type	Description

	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Operator Aspects

The COS operator can be used in various expressions. Similarly, the COS function is invoked in Structured Text statements. Both applications of COS return a floating point result containing the cosine of the Source. Depending on the context this value may then be type converted if appropriate.

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is set.
Source	REAL	Input to the trigonometric instruction.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if instruction is enabled.
Dest	REAL	Result of the math instruction.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in.

	Dest = cosine value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = cosine value of the Source. If overflow occurs Clear EnableOut to false else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

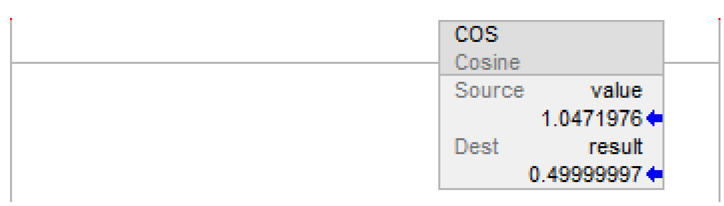
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = cosine value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

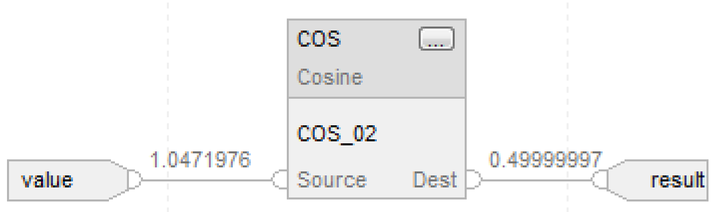
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := COS(REAL_src);

Sine (SIN)

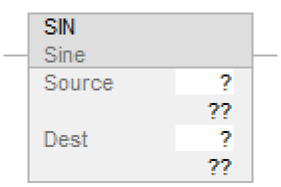
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the SIN instruction takes the sine of the Source value (in radians) and stores the result in the Destination.

Available Languages

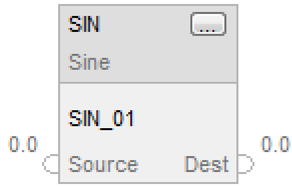
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use SIN as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to sine.

Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Type	Format	Description
SIN tag	FBD_MATH_ADVANCED	tag	SIN structure

FBD_MATH_ADVANCED Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the trigonometric instruction.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT	Value to convert to sine.

	UINT UDINT ULINT REAL LREAL	

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Operator Aspects

The SIN operator can be used in various expressions. Similarly, the SIN function is invoked in Structured Text statements. Both applications of SIN return a floating point result containing the sine of the Source. Depending on the context, this value may then be type converted if appropriate.

Description

The SIN instruction takes the sine of the Source value (in radians) and stores the result in the Destination.

The SIN operator or function computes the sine of Source and returns the floating point result. The resulting value is always greater than or equal to -1 and less than or equal to 1.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in.

	Dest = sine value of the Source.
Postscan	N/A

Function Block

Condition/State	Action Taken
Prescan	N/A
Tag.EnableIn is false.	Set EnableOut to EnableIn.
Tag.EnableIn is true	Dest = sine value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

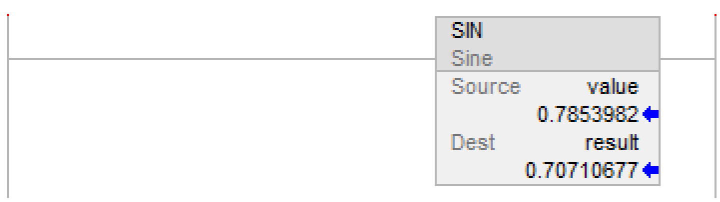
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = sine value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

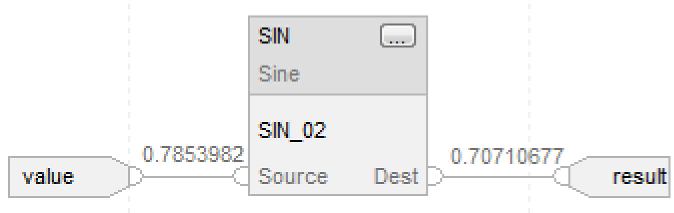
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := SIN(REAL_src);

Tangent (TAN)

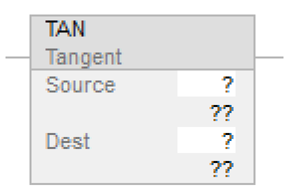
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The TAN instruction takes the tangent of the Source value (in radians) and stores the result in the Destination. The TAN operator or function computes the tangent of Source and returns the floating point result.

Available Languages

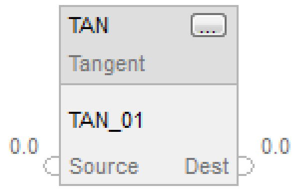
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use TAN as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to tangent.

Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
TAN tag	FBD_MATH_ADVANCED	Structure	TAN structure

FBD_MATH_ADVANCED Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the trigonometric instruction.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Operand	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT INT DINT LINT USINT	Value to convert to tangent.

	UINT UDINT ULINT REAL LREAL	

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

Operator Aspects

The TAN operator can be used in various expressions. Similarly, the TAN function is invoked in Structured Text statements. Both applications of TAN return a floating point result containing the tangent of the Source. Depending on the context, this value may then be type converted if appropriate.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = tangent value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
Tag.EnableIn is false	Set EnableOut to EnableIn.
Tag.EnableIn is true	Dest = tangent value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

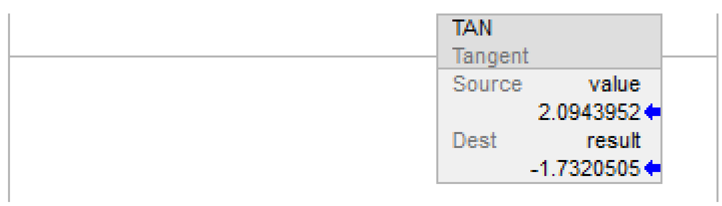
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = tangent value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

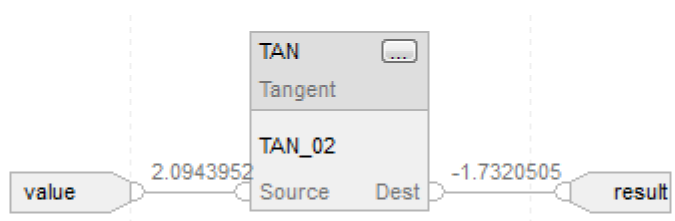
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

```
REAL_dest := TAN(REAL_src);
```

Advanced Math Instructions

The advanced math instructions include these instructions:

Ladder Diagram and Function Block

Natural Log (LN) on page 732	Log Base 10 (LOG) on page 727	X to the Power of Y (EXPT) on page 737

Structured Text

Natural Log (LN) on page 732	Log Base 10 (LOG) on page 727	X to the Power of Y (EXPT) on page 737

If you want to:	Use this instruction:
Take the natural log of a value	LN
Take the log base 10 of a value	LOG
Raise a value to the power of another value	EXPT

Mixing data types can cause accuracy and rounding errors and cause the instruction to take longer to execute. Check the S:V bit to see whether the result was truncated.

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

An advanced math instruction executes once each time the instruction is scanned as long as the rung-condition-in is true. If you want the instruction evaluated only once, use an ONS instruction to trigger the math instruction.

Log Base 10 (LOG)

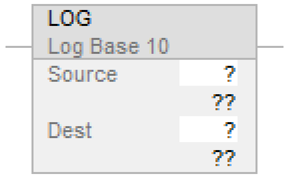
This table lists the controllers and applications that support this instruction.

Architecture	**Standard applications**	**Safety applications**
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The Log Base 10 (LOG) instruction takes the log base 10 of the Source and stores the result in the Destination.

Available Languages

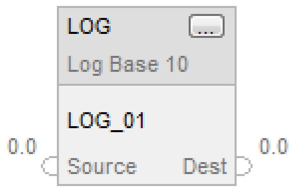
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use LOG as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value for which the instruction finds the log.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Type	Format	Description
LOG	FBD_MATH_ADVANCED	tag	LOG structure

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Value for which the instruction finds the log.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of which to find the log of this value

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

Description

The LOG instruction takes the log base 10 of the Source and stores the result in the Destination. The Source must be greater than zero or a minor fault will occur.

Affects Math Status Flags

Controllers	Affects Math Status Flag
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional

Controllers	Affects Math Status Flag
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = value of log base 10 of the Source.
Postscan	N/A.

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false.	Set EnableOut to EnableIn.
EnableIn is true	Dest = value of natural log of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

FBD Function

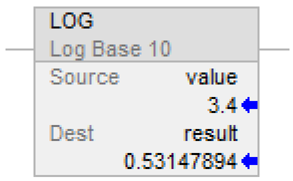
Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = value of log base 10 of the Source.

Condition/State	Action Taken
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

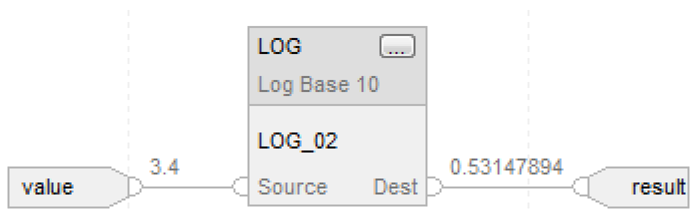
Examples

Ladder Diagram

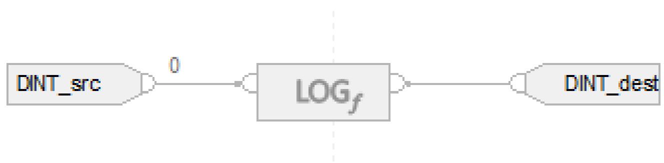


Function Block Diagram

FBD Block



FBD Function



Structured Text

result := LOG(value);

Natural Log (LN)

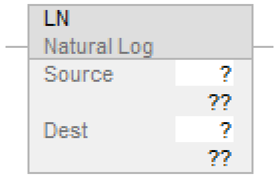
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The Natural Log (LN) instruction takes the natural log of the Source and stores the result in the Destination. The Source must be greater than zero or a minor fault will occur.

Available Languages

Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use LN as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Find the natural log of this value.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
LN	FBD_MATH_ADVANCED	tag	LN structure

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.

Input Members	Data Type	Description
Source	REAL	Value for which the instruction finds the natural log.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value of which to find the natural log of this value.

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	REAL LREAL	Result of the function.

See FBD Functions on page 862.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

See Math status flags on page 849.

Major/Minor Faults

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = value of natural log of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = value of natural log of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

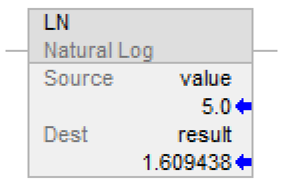
FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

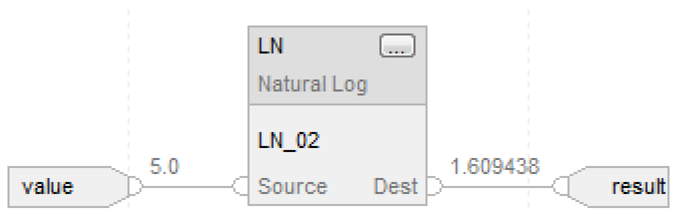
Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = value of natural log of the Source
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

Examples

Ladder Diagram



Function Block



FBD Function

Structured Text

result := LN(value);

X to the Power of Y (EXPT)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No

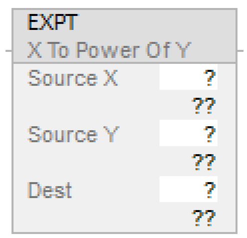
Architecture	Standard applications	Safety applications
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The X to the Power of Y (EXPT) instruction takes Source A (X) to the power of Source B (Y) and stores the result in the Destination.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from XPY to EXPT.

Available Languages

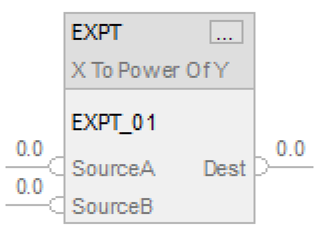
Ladder Diagram



Function Block Diagram

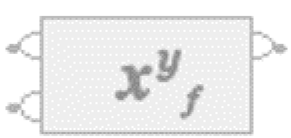
Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.



Structured Text

This instruction is not available in structured text.

Tip: Use ** as an operator in an expression to compute the same result. Refer to *Structured Text Syntax* for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source A	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	value to exponentiate
Source B	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	immediate tag	exponent
Dest	SINT INT	SINT INT	tag	Tag to store the result of the instruction.

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
	DINT REAL	DINT LINT USINT UINT UDINT ULINT REAL LREAL		

Function Block Diagram

FBD Block

Operand	Data Type	Format	Description
EXPT	FBD_MATH	tag	EXPT structure

FBD_MATH Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
SourceA	REAL	Value added to SourceB.
SourceB	REAL	Value added to SourceA.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

FBD Function

Input Operands (Left Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Source A (top)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	Value to exponentiate
Source B (bottom)	SINT USINT INT UINT DINT UDINT LINT ULINT REAL LREAL	exponent

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	DINT UDINT LINT ULINT REAL LREAL	Result of the function.

Description

The XPY instruction takes Source A (X) to the power of Source B (Y) and stores the result in the Destination. If Source A (X) is negative, Source B (Y) must be a non-fractional value or a minor fault will occur.

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A.
Rung-condition-in is false.	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true.	Set Rung-condition-out to Rung-condition-in. Dest = value of Source X to the power of Source Y.
Postscan	N/A.

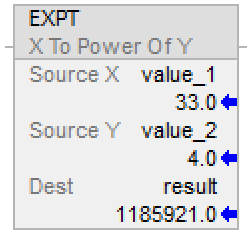
Function Block Diagram

FBD Block

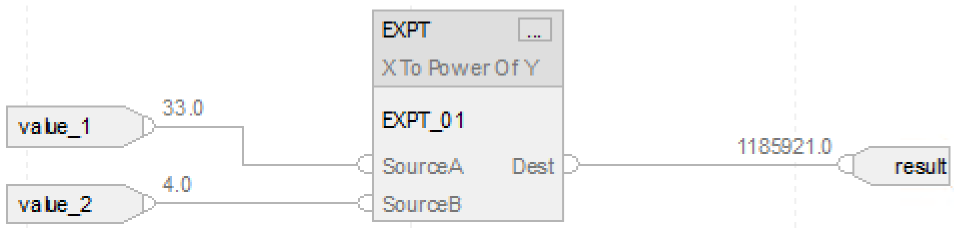
Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = value of Source X to the power of Source Y. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

Examples

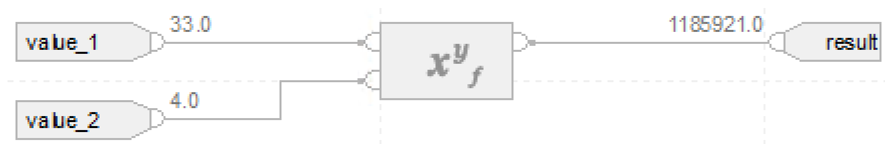
Ladder Diagram



Function Block



FBD Function



Structured Text

```
result := value_1 ** value_2;
```

Math Conversion Instructions

The math conversion instructions convert values.

Available Instructions

Ladder Diagram and Function Bock

DEG on page 753	RAD on page 758	TO_BCD on page 745	BCD_TO on page 749	TRUNC on page 763

Structured Text

DEG on page 753	RAD on page 758	TRUNC on page 763

If you want to	Use this instruction
Convert radians into degrees.	DEG
Convert degrees into radians.	RAD
Convert an integer value to a BCD value.	TO_BCD
Convert a BCD value to an integer value.	BCD_TO
Remove the fractional part of a value.	TRUNC

You can mix data types, but loss of accuracy and rounding error might occur and the instruction takes more time to execute. Check the S:V bit to see whether the result was truncated.

The **bold** data types indicate optimal data types. An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

A math conversion instruction executes once each time the instruction is scanned as long as the rung-condition-in is true. If you want the instruction evaluated only once, use an ONS instruction to trigger the conversion instruction.

Convert to BCD (TO_BCD)

This table lists the controllers and applications that support this instruction.

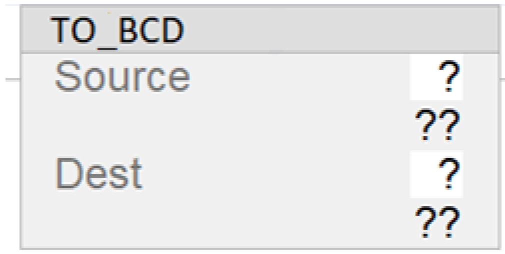
Architecture	**Standard applications**	**Safety applications**
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The TO_BCD instruction converts a decimal value ($0 \leq$ Source $\leq$ 99,999,999) to a BCD value and stores the result in the Destination.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from TOD to TO_BCD.

Available Languages

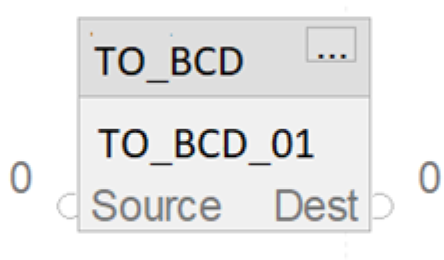
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



Structured Text

This instruction is not available in structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description

Source	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	Immediate tag	Value to convert to BCD 0 $\leq$ Source $\leq$ 99,999,999
Destination	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	tag	Tag to store the result

Function Block Diagram

FBD Block

Operand	Type	Format	Description
TO_BCD tag	FBD_CONVERT	Structure	TO_BCD structure

FBD_CONVERT Structure

Input Member	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Source	DINT	Input to the conversion instruction. Valid = any integer

Output Member	Data Type	Description
EnableOut	BOOL	Enable output.
Dest	DINT	Result of the conversion instruction. Math status flags are set for this output.

Description

BCD is the Binary Coded Decimal number system that expresses individual decimal digits (0-9) in a 4-bit binary notation.

Source	Destination	Destination Type
Negative source < 0	0	
Source > 9,999, 999,999, 999,999	16#9999_9999_9999_9999	ULINT
Source > 9,999, 999,999, 999,999	16#9999_9999_9999_9999	LINT
Source > 99,999,999	16#9999_9999	UDINT
Source > 99,999,999	16#9999_9999	DINT
Source > 99,999,999	16#9999	UINT
Source > 99,999,999	16#9999	INT
Source > 99,999,999	16#99	USINT
Source > 99,999,999	16#99	SINT

Affects Math Status Flags

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see *Math status flags* on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

See *Common Attributes for General Instructions* on page 849 for operand related faults.

A minor fault will occur if:	Fault type	Fault code
feature is enabled and overflow detected and Source < 0	4	4
feature is enabled and overflow detected and Source > 99,999,999 / 9,999, 999,999, 999,999	4	4
feature is enabled and overflow detected	4	4

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A.
Rung-condition-in is false	N/A.
Rung-condition-in is true	The controller converts the Source to BCD and places the result in the Destination.
Postscan	N/A.

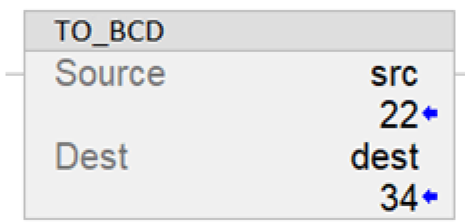
Function Block Diagram

FBD Block

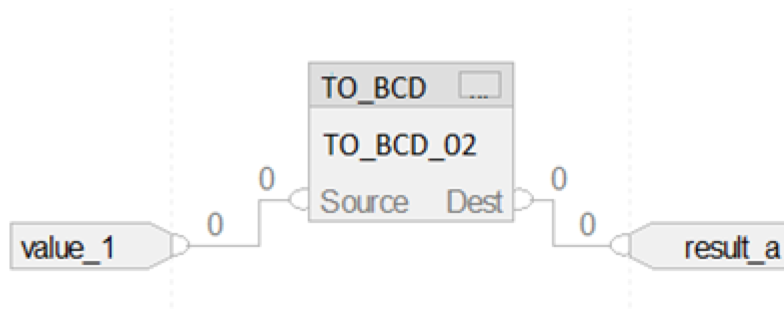
Condition/State	Action Taken
Prescan	N/A
Tag.EnableIn is false.	EnableOut is cleared to false
Tag.EnableIn is true	Dest = the result of computation in BCD value. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

Example

Ladder Diagram



Function Block



Convert to Integer (BCD_TO)

This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No

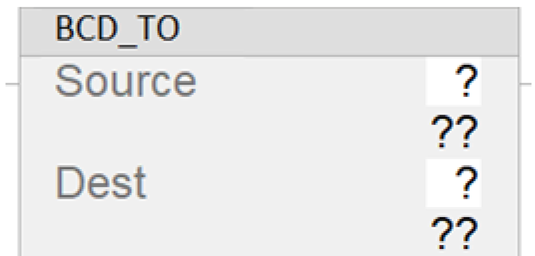
Architecture	Standard applications	Safety applications
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

The BCD_TO instruction converts a BCD value (Source) to a decimal value and stores the result in the Destination.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from FRD to BCD_TO.

Available Languages

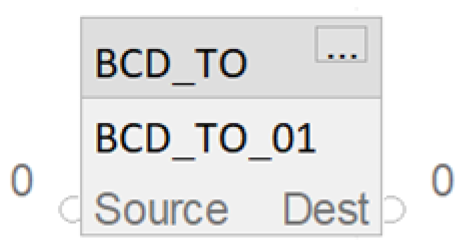
Ladder Diagram



Function Block Diagram

Function Block Diagram supports this element:

FBD Block



Structured Text

This instruction is not available in structured text.

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	Immediate tag	value to convert to decimal
Destination	SINT INT DINT	SINT INT DINT LINT USINT UINT UDINT ULINT	tag	tag to store the result

Function Block Diagram

FBD Block

Operand	Type	Format	Description
FRD tag	FBD_CONVERT	Structure	FRD structure

FBD_CONVERT Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If cleared, the instruction does not execute and outputs are not updated. Default is set.
Source	DINT	Input to the conversion instruction. Valid = any integer

Output Parameters	Data Type	Description
EnableOut	BOOL	Enable output.

Dest	DINT	Result of the conversion instruction.

Description

The BCD_TO instruction converts a BCD value (Source) to a decimal value and stores the result in the Destination.

This formula is used for calculations when source is 32bits:

$$Destination = (16\#Source8*10^7) + (16\#Source7*10^6) + (16\#Source6*10^5) + (16\#Source5*10^4) + (16\#Source4*10^3) + (16\#Source3*10^2) + (16\#Source2*10^1) + (16\#Source1*10^0)$$

For example:

Source = 16#1234_567E

$$Destination = (1*10^7)+(2*10^6)+(3*10^5)+(4*10^4)+(5*10^3)+(6*10^2)+(7*10^1)+(14*10^0)=12345684$$

Affects Math Status Flags

Major/Minor Faults

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see *Math status flags* on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = decimal value of source with BCD value.
Postscan	N/A

Function Block Diagram

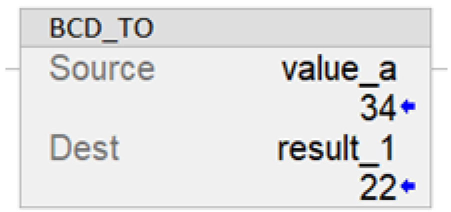
FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.

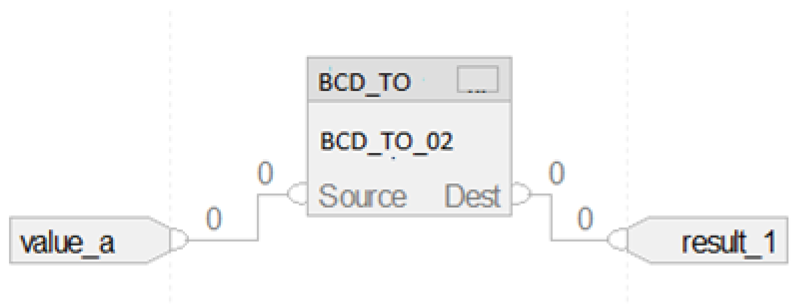
EnableIn is true	Dest = decimal value of source with BCD value If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

Examples

Ladder Diagram



Function Block



Degrees (DEG)

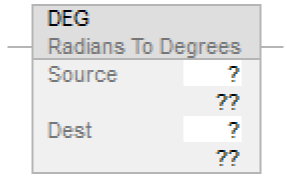
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the DEG instruction converts the Source (in radians) to degrees and stores the result in the Destination.

Available Languages

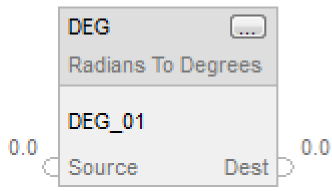
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

Function Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use DEG as an operator in an expression to compute the same result. Refer to Structured Text Syntax on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to degrees
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result

Function Block Diagram

FBD Block

Operand	Type	Format	Description
DEG	FBD_MATH_ADVANCED	tag	DEG structure

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description

	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to degrees

Output Operand (Right Pin)	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Description
Dest	REAL LREAL	Result of the function

See *FBD Functions* on page 862.

FBD_MATH_ADVANCED Structure

Input Parameter	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is True.
Source	REAL	Input to the conversion instruction.

Output Parameter	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

Description

The DEG instruction uses this algorithm:

Source*180/pi = Source*57.29578

Affects Math Status Flags

Major/Minor Faults

Controllers	Affects Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see *Math status flags* on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

None specific to this instruction. See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = degree value of the Source.
Postscan	N/A

Function Block Diagram

Function Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = degree value of the Source. If overflow occurs Clear EnableOut to false else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

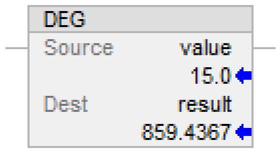
FBD Function

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = DEG(Source)
Instruction first run	N/A
Instruction first scan	N/A

Postscan	N/A

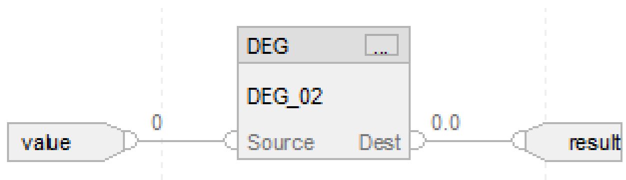
Examples

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := DEG(REAL_src);

Radian (RAD)

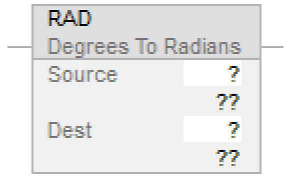
This table lists the controllers and applications that support this instruction.

Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the RAD instruction converts the Source (in degrees) to radians and stores the result in the Destination.

Available Languages

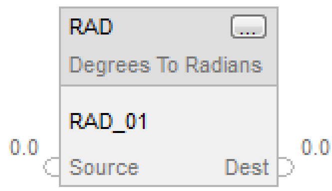
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use RAD as an operator in an expression to compute the same result. Refer to *Structured Text Syntax* on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Immediate tag	Value to convert to radians
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result

Function Block Diagram

FBD Block

Operand	Type	Format	Description
RAD	FBD_MATH_ADVANCED	tag	RAD structure

FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Input Operands (Left Pins)	Data Type	Description

	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	
Source	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	Value to convert to radians

Output Operand (Right Pin)	CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers Data Type	Description
Dest	REAL LREAL	Result of the function.

FBD_MATH_ADVANCED Structure

Input Members	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the conversion instruction.

Output Members	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	REAL	Result of the instruction.

Description

The RAD instruction uses this algorithm:

Deg2RadConvFactor=pi/180= 0.017453292

Destination = Source * Deg2RadConvFactor

Affects Math Status Flags

Controllers	Affected Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see Math Status Flags on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See Index Through Arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action taken
Prescan	N/A
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = radian value of the Source.
Postscan	N/A

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = radian value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

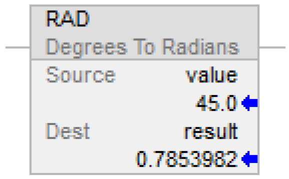
FBD Function

Condition/State	Action Taken
Prescan	N/A
Normal Scan	Dest = RAD(Source)
Instruction first run	N/A
Instruction first scan	N/A

Postscan	N/A

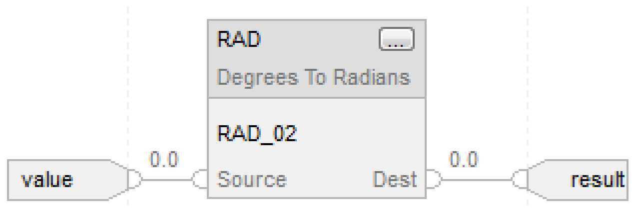
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := RAD(REAL_src);

Truncate (TRUNC)

This table lists the controllers and applications that support this instruction.

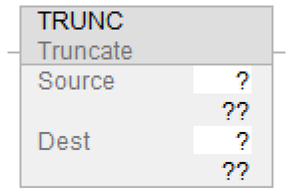
Architecture	Standard applications	Safety applications
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Yes	Yes

When enabled, the TRUNC instruction removes (truncates) the fractional part of the Source and stores the result in the Destination.

Tip: In Logix Designer version 36, the mnemonic for this instruction changed from TRN to TRUNC.

Available Languages

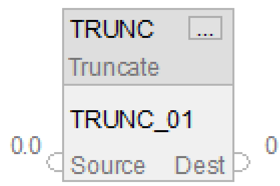
Ladder Diagram



Function Block Diagram

Function Block Diagram supports these elements:

FBD Block



FBD Function

Tip: FBD Function is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.

Structured Text

This instruction is not available in structured text.

Tip: Use TRUNC as an operator in an expression to compute the same result. Refer to *Structured Text Syntax* on page 879 for more information on the syntax of expressions and assignments within structured text.

Operands

IMPORTANT: Unexpected operation may occur if:

- Output tag operands are overwritten.
- Members of a structure operand are overwritten.
- Except when specified, structure operands are shared by multiple instructions.

There are data conversion rules for mixing numeric data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Data Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Format	Description
Source	SINT INT DINT REAL	REAL LREAL	immediate tag	Value to truncate.
Destination	SINT INT DINT REAL	SINT INT DINT LINT USINT UINT UDINT ULINT REAL LREAL	tag	Tag to store the result of the instruction.

Function Block Diagram

FBD Block

Operand	Type	Format	Description
TRUNC	FBD_TRUNCATE	tag	TRN structure

FBD Function

Input Operands (Left Pins)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description

Source	REAL LREAL	Value to truncate

Output Operand (Right Pin)	Data Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Description
Dest	LINT	Result of the function.

See *FBD Functions* on page 862.

FBD_TRUNCATE Structure

Input Member	Data Type	Description
EnableIn	BOOL	Enable input. If false, the instruction does not execute and outputs are not updated. Default is true.
Source	REAL	Input to the conversion instruction. Input also takes SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT and LREAL through input tag. But the integer type will be converted to REAL type first. Converting SINT or INT or USINT or UINT to REAL, there is no data precision lost. Converting 32-bit types (DINT, UDINT) to REAL, data precision could be lost. Both data types store data in 32 bits, but the REAL type uses some of its 32 bits to store the exponent value. If precision is lost, the controller takes it from the least-significant portion of the 32 bit types (DINT, UDINT). Converting 64 bit types (LINT, ULINT and LREAL) to REAL, data precision could be lost.

Output Member	Data Type	Description
EnableOut	BOOL	Indicates if the instruction executed without fault when it was enabled.
Dest	DINT	Result of the instruction.

Description

Truncating does not round the value; rather, the non-fractional part remains the same, regardless of the value of the fractional part.

Affects Math Status Flags

Controllers	Affected Math Status Flags
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Conditional, see *Math status flags* on page 849.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes

Major/Minor Faults

None specific to this instruction. See *Index through arrays* on page 863 for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A.
Rung-condition-in is false	Set Rung-condition-out to Rung-condition-in.
Rung-condition-in is true	Set Rung-condition-out to Rung-condition-in. Dest = Truncated value of the Source.
Postscan	N/A.

Function Block Diagram

FBD Block

Condition/State	Action Taken
Prescan	N/A
EnableIn is false	Set EnableOut to EnableIn.
EnableIn is true	Dest = Truncated value of the Source. If overflow occurs Clear EnableOut to false. else Set EnableOut to true.
Instruction first scan	N/A
Instruction first run	N/A
Postscan	N/A

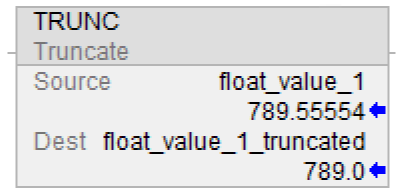
FBD Function

Condition/State	Action Taken
Prescan	N/A

Normal Scan	Dest = Truncated value of the Source.
Instruction first run	N/A
Instruction first scan	N/A
Postscan	N/A

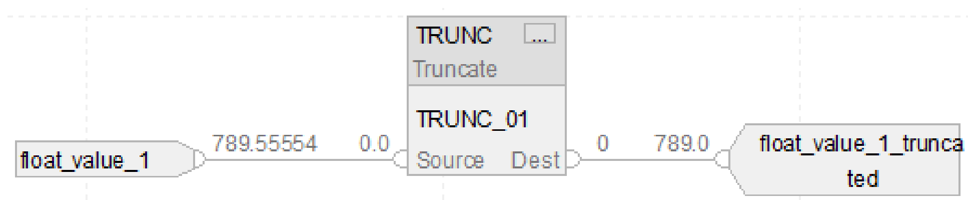
Example

Ladder Diagram



Function Block Diagram

FBD Block



FBD Function

Structured Text

REAL_dest := TRUNC(REAL_src);

ASCII Serial Port Instructions

Use the ASCII serial port instructions to read and write ASCII characters.

IMPORTANT: The ASCII serial port instructions are included in Logix Designer versions 36 and earlier. They are not included in versions 37 and later.

IMPORTANT: To use the ASCII serial port instructions, you must configure the serial port of the controller. Refer to the Logix 5000 Controller Common Procedures manual (publication 1756-PM001) for more information.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for projects using controllers that do not have serial ports.

Available Instructions

Ladder Diagram and Structured Text

ASCII Test for Buffer Line (ABL) on page 790	ASCII Chars in Buffer (ACB) on page 771	ASCII Clear Buffer (ACL) on page 774	ASCII Handshake Lines (AHL) on page 776	ASCII Read (ARD) on page 781	ASCII Read Line (ARL) on page 785	ASCII Write Append (AWA) on page 798	ASCII Write (AWT) on page 793

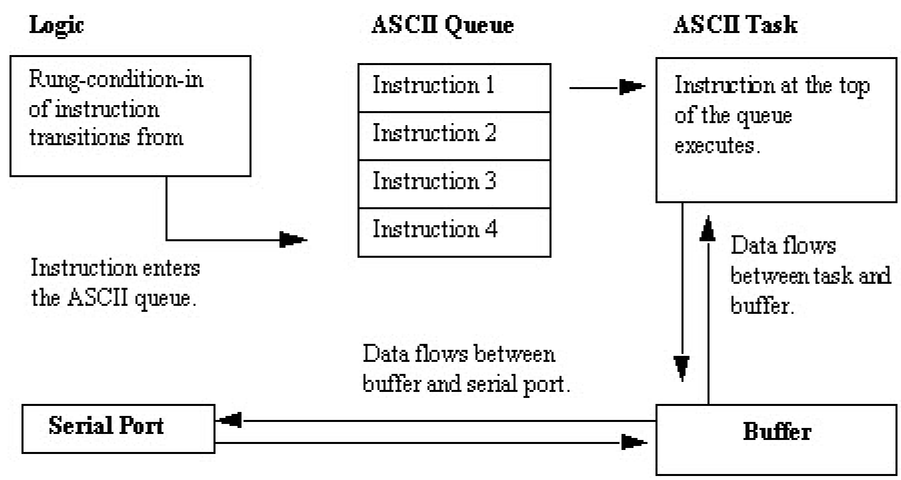
Function Block

Not available

If you want to:	Use this instruction:
Check for data that contains termination characters	ABL
Check for the required number of characters before reading the buffer	ACB
Clear the buffer. For example, remove old data from the buffer at start-up, or synchronize the buffer with a device. Clear out ASCII serial port instructions that are currently executing or are in the queue.	ACL
Obtain the status of the serial port control lines. For example, cause a modem to hang up. Turn the DTR signal on or off Turn the RTS signal on or off	AHL
Read a fixed number of characters. For example, read data from a device that sends the same number of characters with every transmission)	ARD

Read a varying number of characters, up to and including the first set of termination characters. For example, read data from a device that sends a varying number of characters with every transmission.	ARL
Send characters and automatically append one or two additional characters to mark the end of the data. For example, send messages that always use the same termination character(s).	AWA
Send characters. For example, send messages that use a variety of termination characters.	AWT

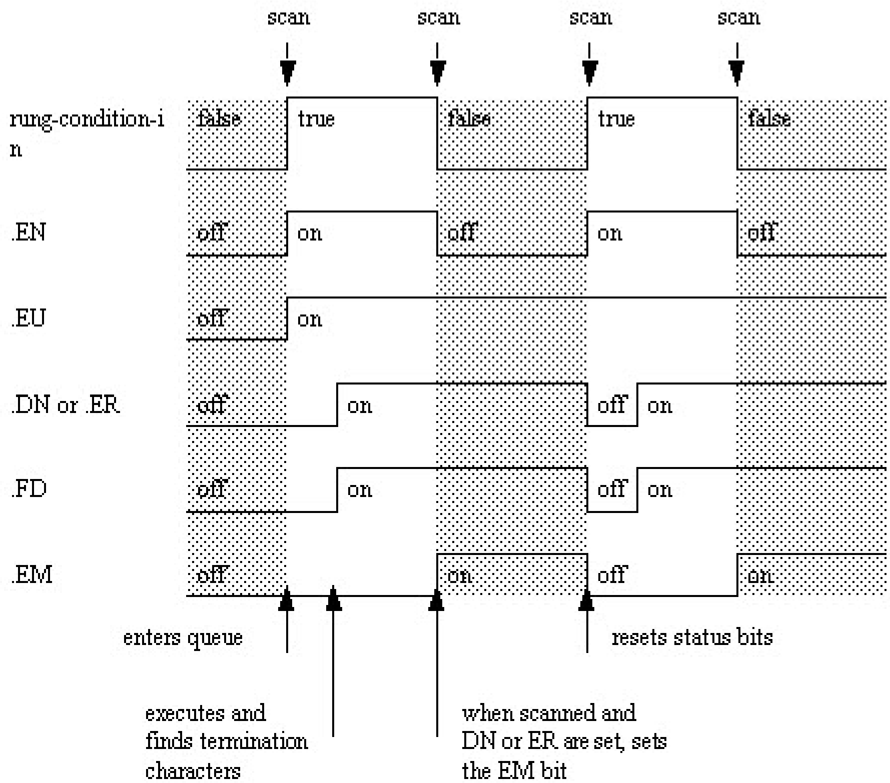
ASCII serial port instructions execute asynchronous to the scan of the logic:



Each ASCII instruction, except for the ACL instruction, uses a SERIAL_PORT_CONTROL structure. The SerialPort Control operand:

- controls the execution of the instruction
- provides status information about the instruction ASCII instructions execute asynchronous to the scan of the logic:

The bits of the SerialPort Control operand provide status information:



ASCII Chars in Buffer (ACB)

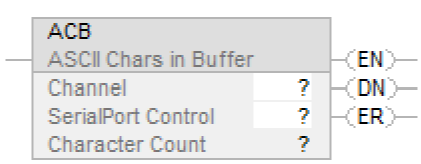
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Chars in Buffer (ACB) instruction counts the characters in the buffer.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

ACB(Channel,SerialPortControl);

Operands

Ladder Diagram

Operand	Type	Format	Description
Channel	DINT	immediate tag	0
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	0 During execution, displays the number of characters in the buffer, including the first set of termination characters.

Structured Text

Operand	Type	Format	Description
Channel	DINT	immediate tag	0
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	0 During execution, displays the number of characters in the buffer, including the first set of termination characters.

You can specify the Character Count value by accessing the .POS member of the SERIAL_PORT_CONTROL structure, rather than by including the value in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue indicates 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 the instruction is executing.

.EM	BOOL	The empty bit indicates 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 the instruction found a character.
.POS	DINT	The position determines the number of characters in the buffer, up to and including the first set of termination characters.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The ACB instruction counts the characters in the buffer.

To program the ACB instruction, follow these guidelines:

- Configure the serial port of the controller for User mode.

This is a transitional instruction:

- In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition

Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes when EnableIn toggles from cleared to set.
Postscan	N/A

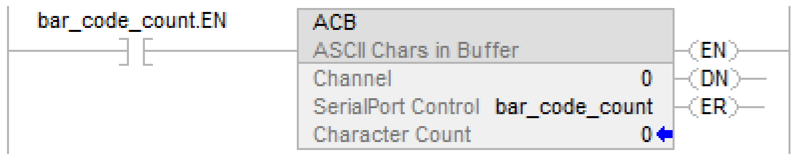
Structured Text

Condition	Structured Text Action

Prescan	N/A
Normal execution	The instruction executes when EnableIn toggles from cleared to set.
Postscan	N/A

Example

Ladder Diagram



Structured Text

ACB(0,bar_code_count);

ASCII Clear Buffer (ACL)

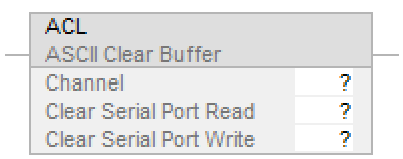
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Clear Buffer (ACL) instruction immediately clears the buffer and ASCII queue.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

ACL(Channel,ClearSerialPortRead,ClearSerialPortWrite);

Operands

Ladder Diagram

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 1.
Clear Serial Port Write	BOOL	immediate tag	To remove AWA and AWT instructions from the queue, enter 1.

Structured Text

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 1.
Clear Serial Port Write	BOOL	immediate tag	To remove AWA and AWT instructions from the queue, enter 1.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

The ACL instruction immediately performs one or both of the following actions:

- Clears the buffer or characters and clears the ASCII queue of read instructions
- Clears the ASCII queue of write instructions To program the ACL instructions, follow these guidelines:

Configure the serial port of the controller:

If your application:	Then:
Uses ARD or ARL instruction	Select User mode
Does not use ARD or ARL instructions	Select either System or User mode

To determine if an instruction was removed from the queue or cancelled, examine the following of the appropriate instruction:

- .ER bit is set
- .ERROR member is 16#E

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

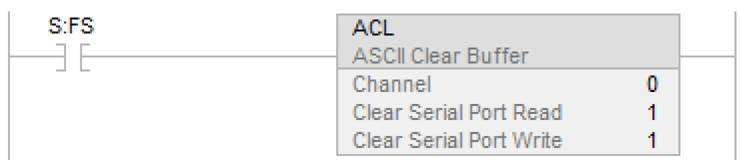
Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction clears the specified instruction and buffer(s)
Postscan	N/A

Example

Ladder Diagram



Structured Text

IF (osri_1.OutputBit THEN

ACL(0,0,1);

END_IF;

ASCII Handshake Lines (AHL)

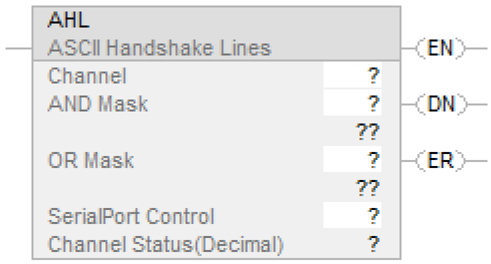
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Handshake Lines (AHL) instruction obtains the status of control lines and turns on or off the DTR and RTS signals.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

AHL(Channel,ANDMask,ORMask,SerialPortControl);

Operands

Ladder Diagram

<table>
<tr><th>Operand</th><th>Type</th><th>Format</th><th colspan="2">Description</th></tr>
<tr><td>Channel</td><td>DINT</td><td>immediate tag</td><td colspan="2">0</td></tr>
<tr><td>ANDMask</td><td>DINT</td><td>immediate tag</td><td colspan="2" rowspan="2">Refer to the Description</td></tr>
<tr><td>ORMask</td><td>DINT</td><td>immediate tag</td></tr>
<tr><td>SerialPort Control</td><td>SERIAL_PORT_CONTROL</td><td>tag</td><td colspan="2">Tag that controls the operation</td></tr>
<tr><td rowspan="8">Channel Status (Decimal)</td><td rowspan="8">DINT</td><td rowspan="8">immediate</td><td colspan="2">0
During execution, displays the status of the control lines.</td></tr>
<tr><td>For the status of this control line:</td><td>Examine this bit:</td></tr>
<tr><td>CTS</td><td>0</td></tr>
<tr><td>RTS</td><td>1</td></tr>
<tr><td>DSR</td><td>2</td></tr>
<tr><td>DCD</td><td>3</td></tr>
<tr><td>DTR</td><td>4</td></tr>
<tr><td>Received the XOFF character</td><td>5</td></tr>
</table>

Structured Text

<table>
<tr><th>Operand</th><th>Type</th><th>Format</th><th colspan="2">Description</th></tr>
<tr><td>Channel</td><td>DINT</td><td>immediate tag</td><td colspan="2">0</td></tr>
<tr><td>ANDMask</td><td>DINT</td><td>immediate tag</td><td colspan="2" rowspan="2">Refer to the Description</td></tr>
<tr><td>ORMask</td><td>DINT</td><td>immediate tag</td></tr>
<tr><td>SerialPort Control</td><td>SERIAL_PORT_CONTROL</td><td>tag</td><td colspan="2">Tag that controls the operation</td></tr>
<tr><td rowspan="8">Channel Status
(Decimal)</td><td rowspan="8">DINT</td><td rowspan="8">immediate</td><td colspan="2">0
During execution, displays the status of the control lines.</td></tr>
<tr><td>For the status of this control line:</td><td>Examine this bit:</td></tr>
<tr><td>CTS</td><td>0</td></tr>
<tr><td>RTS</td><td>1</td></tr>
<tr><td>DSR</td><td>2</td></tr>
<tr><td>DCD</td><td>3</td></tr>
<tr><td>DTR</td><td>4</td></tr>
<tr><td>Received the XOFF character</td><td>5</td></tr>
</table>

You can specify the Channel Status value by accessing the .POS member of the SERIAL_PORT_CONTROL structure, rather than by including the value in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates the instruction enter the ASCII queue.
.DN	BOOL	The done bit indicates the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates the instruction is executing.
.EM	BOOL	The empty bit indicates 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 does not apply to this instruction.
.POS	DINT	The position determines the number of characters in the buffer, up to and including the first set of termination characters.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The AHL instruction can:

- Obtain the status of the control lines of the serial port
- Turn the Data Terminal Ready (DTR) signal on or off
- Turn the Request to Send (RTS) signal on or off

To program the AHL instruction, follow these guidelines:

Configure the serial port of the controller:

If your application:	Then:
Uses ARD or ARL instruction	Select User mode
Does not use ARD or ARL instructions	Select either System or User mode

Use the following table to select the correct values for the ANDMask and ORMask operands:

To turn DTR:	And turn RTS:	Enter this ANDMask value:	And enter this ORMask value:
Off	Off	3	0
		on	1
		unchanged	1
On	Off	2	1
		on	0
		unchanged	0
Unchanged	Off	2	0
		on	0
		unchanged	0

This is a transitional instruction:

- In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition

Affects Math Status Flags

No

Fault Conditions

Type	Code	Cause	Recovery Method
4	57	The AHL instruction failed to execute because the serial port is set to no handshaking	Change the Control Line setting of the serial port or Delete the AHL instruction

Execution

Ladder Diagram

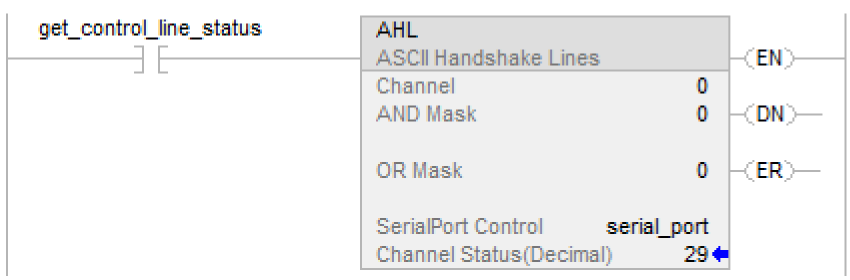
Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes when rung condition in toggles from cleared to set.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes when rung condition in toggles from cleared to set.
Postscan	N/A

Example

Ladder Diagram



Structured Text

osri_1.InputBit := get_control_line_status;

OSRI(osri_1);

IF (osri_1.OutputBit) THEN

AHL(0,0,0,serial_port);

END_IF;

ASCII Read (ARD)

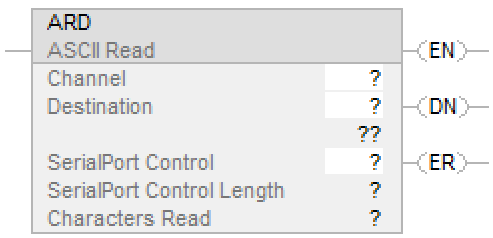
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Read (ARD) instruction removes characters from the buffer and stores them in the Destination.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

ARD(Channel,Destination,SerialPortControl);

Operands

Ladder Diagram

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Destination	String type SINT INT DINT	tag	tag into which the characters are moved (i.e., read): For a string type, enter the name of the tag. For a SINT, INT, or DINT array, enter the first element of the array.	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are: default STRING data type

				any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	number of characters to move to the destination (read)	The Serial Port Control Length must be less than or equal to the size of the Destination. If you want to set the Serial Port Control Length equal to the size of the Destination, enter 0.
Characters Read	DINT	immediate	0	During execution, displays the number of characters in the buffer, including the first set of termination characters.

Structured Text

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Destination	String type SINT INT DINT	tag	tag into which the characters are moved (i.e., read): For a string type, enter the name of the tag. For a SINT, INT, or DINT array, enter the first element of the array.	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are: default STRING data type any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	number of characters to move to the destination (read)	The Serial Port Control Length must be less than or equal to the size of the Destination. If you want to set the Serial Port Control Length equal to the size of the Destination, enter 0.

Characters Read	DINT	immediate	0	During execution, displays the number of characters in the buffer, including the first set of termination characters.

You can specify the Serial Port Control Length and the Characters Read values by accessing the .LEN and .POS members of the SERIAL_PORT_CONTROL structure, rather than by including the values in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates the instruction entered the ASCII queue.
.DN	BOOL	The done bit indicates the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates the instruction is executing.
.EM	BOOL	The empty bit indicates 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 does not apply to this instruction.
.LEN	DINT	The length indicates the number of characters to move to the destination (i.e., read).
.POS	DINT	The position displays the number of characters that were read.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The ARD instruction removes the specified number of characters from the buffer and stores them in the Destination.

- The ARD instruction continues to execute until it removes the specified number of characters (Serial Port Control Length operand).
- While the ARD instruction is executing, no other ASCII serial port instruction executes.

To program the ARD instruction, follow these guidelines:

1. Configure the serial port of the controller for User mode.
2. Use the result of an ACB instruction to trigger the ARD instruction.
   This prevents the ARD instruction from holding up the queue while it waits for the required number of characters. Refer to the ARD example below for more information.
3. This is a transitional instruction:
   In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.
   In structured text, condition the instruction so that it only executes on a transition
4. To trigger a subsequent action when the instruction is done, examine the .EM bit.

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

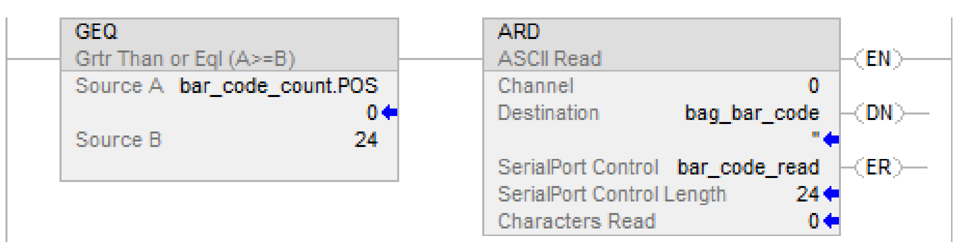
Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Examples

Ladder Diagram



Structured Text

```
ACB(o,bar_code_count);

IF bar_code_count.POS >= 24 THEN

bar_code_read.LEN := 24;

ARD(0,bag_bar_code,bar_code_read);

END_IF;
```

ASCII Read Line (ARL)

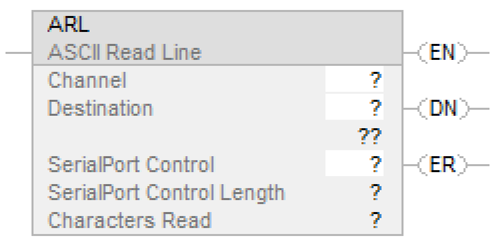
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Read Line (ARL) instruction removes characters from the buffer and stores them in the Destination.

> **Tip:** ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

```
ARL(Channel,Destination,SerialPortControl);
```

Operands

Ladder Diagram

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Destination	String type SINT	tag	tag into which the characters are moved (i.e., read)	If you want to compare, convert, or manipulate

	INT DINT		For a string type, enter the name of the tag. For a SINT, INT, or DINT array, enter the first element of the array.	the characters, enter a string type tag. String types are: default STRING data type any new string type you create
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	maximum number of characters to read if no termination characters are found.	Enter the maximum number of characters that any message will contain (i.e., when to stop reading if no termination characters are found). For example, if messages range from 3 to 6 characters in length, enter 6. The Serial Port Control Length must be less than or equal to the size of the Destination. If you want to set the Serial Port Control Length equal to the size of the Destination, enter 0.
Characters Read	DINT	immediate	0	During execution, displays the number of characters that were read

Structured Text

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Destination	String type SINT INT DINT	tag	tag into which the characters are moved (i.e., read) For a string type, enter the name of the tag.	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are:

			For a SINT, INT, or DINT array, enter the first element of the array.	default STRING data type any new string type you create
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	maximum number of characters to read if no termination characters are found.	Enter the maximum number of characters that any message will contain (i.e., when to stop reading if no termination characters are found). For example, if messages range from 3 to 6 characters in length, enter 6. The Serial Port Control Length must be less than or equal to the size of the Destination. If you want to set the Serial Port Control Length equal to the size of the Destination, enter 0.
Characters Read	DINT	immediate	0	During execution, displays the number of characters that were read

However, you specify the Serial Port Control Length and the Characters Read values by accessing the .LEN and .POS members of the SERIAL_PORT_CONTROL structure, rather than by including the values in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates the instruction entered the ASCII queue.

.DN	BOOL	The done bit indicates the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates the instruction is executing.
.EM	BOOL	The empty bit indicates 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 does not apply to this instruction.
.LEN	DINT	The length indicates the maximum number of characters to move to the destination (i.e., when to stop reading if no termination characters are found).
.POS	DINT	The position displays the number of characters that were read.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The ARL instruction removes characters from the buffer and stores them in the Destination, as follows:

- The ARL instruction continues to execute until it removes either the:
  - First set of termination characters
  - Secified number of characters (String Length operand)

While the ARL instruction is executing, no other ASCII instruction executes. To program the ARL instruction, follow these guidelines:

1. Configure the serial port of the controller for User mode and define the characters that serve as the termination characters.
2. Use the results of an ABL instruction to trigger the ARL instruction.
   This prevents the ARL instruction from holding up the queue while it waits for the termination characters. Refer to the ARL example below for more information.
3. This is a transitional instruction:
   In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.In structured text, condition the instruction so that it only executes on a transition
4. To trigger a subsequent action when the instruction is done, examine the .EM bit.

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

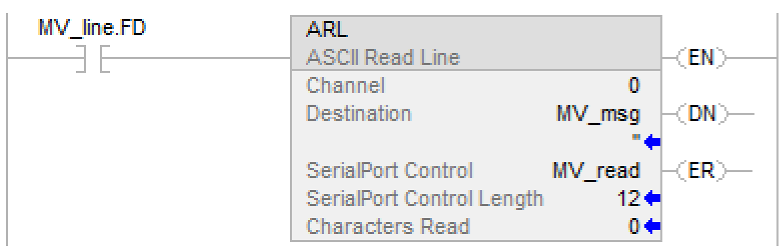
Example

Continuously tests the buffer for a message from the MessageView terminal. Since each message ends in a carriage return ($r), the carriage return is configured as the termination character on the User Protocol tab of the Controller Properties dialog.

When the ABL finds a carriage return, it sets the .FD bit. When the ABL instruction finds the carriage return (MV_line.FD is set), the controller has received a complete message.

The ARL instruction removes the characters from the buffer, up to and including the carriage return, and places them in the DATA member of the MV_msg tag, which is a string type.

Ladder Diagram



Structured Text

ABL(0,MV_line);

osri_1.InputBit :=MVLine.FD

OSRI(osri_1);

IF osri_1.OutputBit) THEN

```
mv_read.LEN := 12;

ARL(0,MV_msg,MV_read);

END_IF;
```

ASCII Test for Buffer Line (ABL)

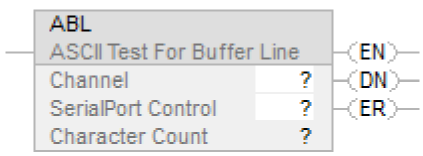
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Test for Buffer Line (ABL) instruction counts the characters in the buffer up to and including the first termination character.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

ABL(Channel,SerialPortControl);

Operands

Ladder Diagram

Operand	Type	Format	Description
Channel	DINT	immediate	0
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	0 During execution, displays the number of characters in the buffer, including the first set of termination characters.

Structured Text

Operand	Type	Format	Description
Channel	DINT	immediate	0
SerialPort Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation
Character Count	DINT	immediate	0 During execution, displays the number of characters in the buffer, including the first set of termination characters.

You access the Character Count value via the .POS member of the SERIAL_PORT_CONTROL structure.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates 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 the instruction is executing.
.EM	BOOL	The empty bit indicates 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 the instruction found the termination character(s).
.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(s).
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The ABL instruction searches the buffer for the first set of termination characters. If the instruction finds the termination characters, it:

- sets the .FD bit
- counts the characters in the buffer up to and including the first set of termination characters

The **User Protocol** tab of the **Controller Properties** dialog box defines the ASCII characters that the instruction considers as the termination characters.

To program the ABL instruction, follow these guidelines:

- Configure the serial port of the controller for User mode and define the characters that serve as the termination characters.

This is a transitional instruction:

- In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

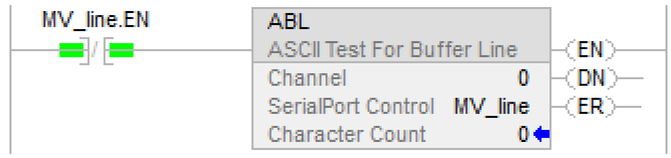
Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Example

Ladder Diagram



Structured Text

ABL(0,MV_line);

ASCII Write (AWT)

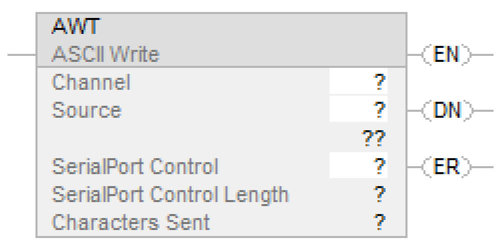
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Write (AWT) instruction sends characters of the Source array to a serial device.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

AWT(Channel,Source,SerialPortControl);

Operands

Ladder Diagram

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Source	String type	tag	Tag that contains the characters to send	If you want to compare, convert, or manipulate

	SINT INT DINT		For a string type, enter the name of the tag. For a SINT, INT, or DINT array, enter the first element of the array.	the characters, enter a string type tag. String types are: default STRING data type any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	Tag that controls the operation	
Serial Port Control Length	DINT	immediate	Number of characters to send	The Serial Port Control Length must be less than or equal to the size of the Source. If you want to set the Serial Port Control Length equal to the number of characters in the Source, enter 0.
Characters Sent	DINT	immediate	0	During execution, displays the number of characters that were sent

Structured Text

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Source	String type SINT INT DINT	tag	Tag that contains the characters to send For a string type, enter the name of the tag. For a SINT, INT, or DINT array, enter the first element of the array.	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are: default STRING data type any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	Tag that controls the operation	
Serial Port Control Length	DINT	immediate	Number of characters to send	The Serial Port Control Length must be less than or equal to the size of the Source.

				If you want to set the Serial Port Control Length equal to the number of characters in the Source, enter 0.
Characters Sent	DINT	immediate	0	During execution, displays the number of characters that were sent

You can specify the Serial Port Control Length and the Characters Sent values by accessing the .LEN and .POS members of the SERIAL_PORT_CONTROL structure, rather than by including the values in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates the instruction entered the ASCII queue.
.DN	BOOL	The done bit indicates the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates the instruction is executing.
.EM	BOOL	The empty bit indicates 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 does not apply to this instruction.
.LEN	DINT	The length indicates the number of characters to send.
.POS	DINT	The position displays the number of characters that were sent.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The AWT instruction sends the specified number of characters (i.e., serial port control length) of the Source tag to the device that is connected to the serial port of the controller.

To program the AWT instruction, follow these guidelines:

1. Configure the serial port of the controller:

If your application:	Then:
Uses ARD or ARL instruction	Select User mode
Does not use ARD or ARL instructions	Select System or User mode

2. This is a transitional instruction: In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute. In structured text, condition the instruction so that it only executes on a transition
3. Each time the instruction executes, do you always send the same number of characters?

If:	Then:
Yes	In the Serial Port Control Length, enter the number of characters to send.
No	Before the instruction executes, move the LEN member of the Source tag to the LEN member of the Serial Port Control tag. Refer to example 2 below.

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Structured Text

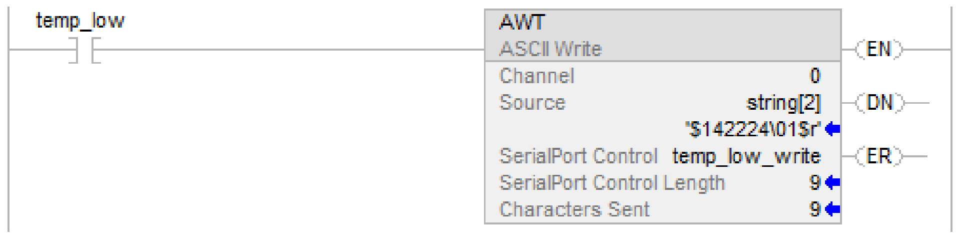
Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Examples

Example 1

When the temperature reaches the low limit (i.e., temp_low is on), the AWT instruction sends a message to the MessageView terminal that is connected to the serial port of the controller. The message nine characters from the DATA member of the string[2] tag, which is a string type. (The $14 counts as one character; it is a hex code for the Ctrl-T character.) The last character is a carriage return ($r), which marks the end of the message.

Ladder Diagram



Structured Text

osri_1.InputBit := temp_low;

OSRI(osri_1);

IF (osri_1.OutputBit) THEN

temp_low_write.LEN := 9;

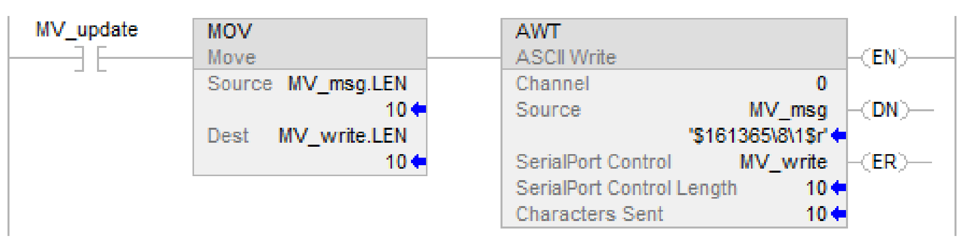
AWT(0,string[2],temp_low_write);

END_IF;

Example 2

When MV_update is on, the AWT instruction sends the characters in MV_msg. Because the number of characters in MV_msg varies, the rung first moves the length of the string (MV_msg.LEN) to the Serial Port Control Length of the AWT instruction (MV_write.LEN). (In MV_msg, the $16 counts as one character; it is the hex code for the Ctrl-V character.)

Ladder Diagram



Structured Text

osri_1.InputBit := MV_update;

OSRI(osri_1);

IF (osri_1.OutputBit) THEN

```
MV_write.LEN := Mv_msg.LEN;

AWT(0.MV_msg,MV_write);

END_IF;
```

ASCII Write Append (AWA)

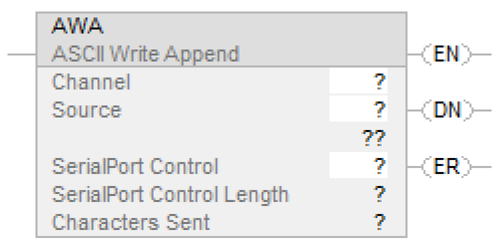
This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

The ASCII Write Append (AWA) instruction sends characters of the Source array to a serial device and appends either one or two predefined characters.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, ACL) are not available for controllers that do not have serial ports.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

AWA(Channel,Source,SerialPortControl);

Operands

Ladder Diagram

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Source	String type SINT INT	tag	tag that contains the characters to send For a string type, enter the name of the tag	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are:

	DINT		For a SINT, INT, or DINT array, enter the first element of the array.	default STRING data type any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	number of characters to send	The Serial Port Control Length must be less than or equal to the size of the Source. If you want to set the Serial Port Control Length equal to the number of characters in the Source, enter 0.
Characters Sent	DINT	immediate	0	During execution, displays the number of characters that were sent.

Structured Text

Operand	Type	Format	Description	Notes
Channel	DINT	immediate tag	0	
Source	String type SINT INT DINT	tag	tag that contains the characters to send For a string type, enter the name of the tag For a SINT, INT, or DINT array, enter the first element of the array.	If you want to compare, convert, or manipulate the characters, enter a string type tag. String types are: default STRING data type any new string type you create
Serial Port Control	SERIAL_PORT_CONTROL	tag	tag that controls the operation	
Serial Port Control Length	DINT	immediate	number of characters to send	The Serial Port Control Length must be less than or equal to the size of the Source. If you want to set the Serial Port Control Length equal to the

				number of characters in the Source, enter 0.
Characters Sent	DINT	immediate	0	During execution, displays the number of characters that were sent.

You can specify the Serial Port Control Length and the Characters Sent values by accessing the .LEN and .POS members of the SERIAL_PORT_CONTROL structure, rather than by including the values in the operand list.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

SERIAL_PORT_CONTROL Structure

Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates the instruction is enabled.
.EU	BOOL	The queue bit indicates the instruction entered the ASCII queue.
.DN	BOOL	The done bit indicates the instruction is done, but it is asynchronous to the logic scan.
.RN	BOOL	The run bit indicates the instruction is executing.
.EM	BOOL	The empty bit indicates 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 does not apply to this instruction.
.LEN	DINT	The length indicates the number of characters to send.
.POS	DINT	The position displays the number of characters that were sent.
.ERROR	DINT	The error contains a hexadecimal value that identifies the cause of an error.

Description

The AWA instruction:

- Sends the specified number of characters (i.e., serial port control length) of the Source tag to the device that is connected to the serial port of the controller
- Adds to the end of the characters (i.e., appends) either one or two characters that are defined on the User Protocol tab of the Controller Properties dialog.

To program the AWA instruction, follow these guidelines:

1. Configure the serial port of the controller:

If your application:	Then:
Uses ARD or ARL instruction	Select User mode
Does not use ARD or ARL instructions	Select either System or User mode

2. This is a transitional instruction: In ladder diagram, toggle the EnableIn from cleared to set each time the instruction should execute.
   In structured text, condition the instruction so that it only executes on a transition
3. Each time the instruction executes, do you always send the same number of characters?

If:	Then:
Yes	In the Serial Port Control Length, enter the number of characters to send.
No	Before the instruction executes, move the LEN member of the Source tag to the LEN member of the Serial Port Control tag. (Refer to example 2 below.)

Affects Math Status Flags

No

Fault Conditions

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Structured Text

Condition	Structured Text Action
Prescan	N/A
Normal execution	The instruction executes. EnableIn toggles from cleared to set.
Postscan	N/A

Examples

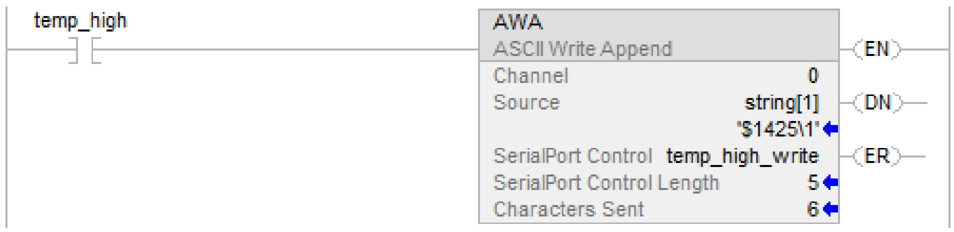
Example 1

When the temperature exceeds the high limit (temp_high is on), the AWA instruction sends a message to the MessageView terminal that is connected to the serial port of the controller.

The message contains five characters from the DATA member of the string[1] tag, which is a string type. (The $14 counts as one character; it is a hex code for the Ctrl-T character.)

The instruction also sends (appends) the characters defined in the controller properties. In this example, the AWA instruction sends a carriage return ($0D), which marks the end of the message.

Ladder Diagram



Structured Text

IF temp_high THEN

temp_high_write.LEN := 5;

AWA(o,string[1],temp_high_write);

temp_high := 0;

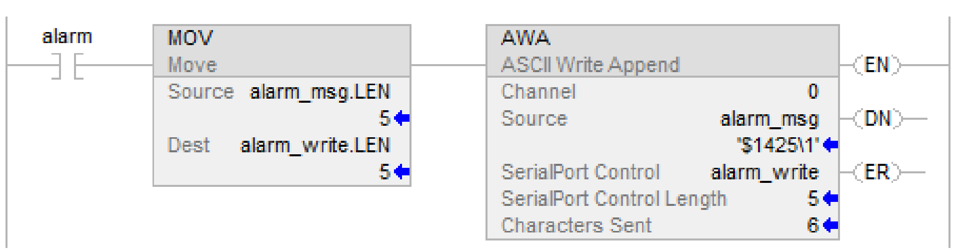
END_IF;

Example 2

When alarms is on, the AWA instruction sends the specified number of characters in alarm_msg and appends a termination character(s). Because the number of characters in alarm_msg varies, the rung first moves the length of the string (alarm_msg.LEN)

to the Serial Port Control Length of the AWA instruction (alarm_write.LEN). In alarm_msg, the $14 counts as one character; it is the hex code for the Ctrl-T character.

Ladder Diagram



Structured Text

```
osri_1.InputBit := alarm;

OSRI(osri_1);

IF(osri_1.OutputBit) THEN

alarm_write.LEN := alarm_msg.LEN;

AWA(0,alarm_msg,alarm_write);

END_IF;
```

String Types

Store ASCII characters in tags that use a string type data type to:

- Use the default STRING data type, which stores up to 82 characters
- Create a new string type that stores less or more characters

To create a new string type, refer to the Logix 5000 Controllers ASCII Strings Programming Manual publication 1756-PM013.

Each string type contains the following 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 using: • The String Browser to enter characters • 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. For example, to access the characters of the string_1 tag, enter string_1. Each element of the DATA array contains one character. Create new string types that store less or more characters.

ASCII Error Codes

If an ASCII serial port instruction fails to execute, the ERROR member of its SERIAL_PORT_CONTROL structure will contain one of the following hexadecimal error codes:

Hex code	Indicates:
16#2	The modem went offline.
16#3	The CTS signal was lost during communication.
16#4	The serial port was in System mode.
16#5	Instructions could not be sent or received because the channel configuration has been shutdown via the channel configuration menu.
16#6	Bad Parameters were passed to the ASCII driver.
16#7	Instructions could not be sent or received because the channel configuration has been shut down via the channel configuration menu.
16#8	Transmission already in progress. This will cause the instruction in progress to error.
16#9	The ASCII Communication requested is not supported by the current channel configuration.
16#10	Attempted to execute an AHL instruction while the Channel was in System Mode.
16#A	Before the instruction executed, the UL bit was set. This stops the execution of the instruction.
16#B	The Port this instruction was requested to operate on does not exist.
16#C	The controller changed from Run mode to Program mode. This stops the execution of an ASCII serial port instruction and clears the queue.
16#D	On the User Protocol tab of the Controller Properties dialog, the buffer size or echo mode parameters were changed and applied. This stops the execution of an ASCII serial port instruction and clears the queue.
16#E	The ACL instruction executed and stopped or removed this type of instruction.
16#F	The serial port configuration changed from User mode to System mode. This stops the execution of an ASCII serial port instruction and clears the queue.

16#51	The LEN value of the string tag is either negative or greater than the DATA size of the string tag.
16#54	The Serial Port Control length is greater than the size of the buffer.
16#55	The Serial Port Control length is either negative or greater than the size of the Source or Destination.

ASCII String Instructions

Use the ASCII string instructions to modify and create strings of ASCII characters.

Available Instructions

Ladder Diagram and Structured Text

FIND on page 808	INSERT on page 810	MID on page 812	CONCAT on page 815	DELETE on page 819

Function Block

Not available

If you want to:	Use this instruction:
Add termination characters or delimiters to a string	CONCAT
Delete characters from a string (e.g., remove header or control characters from a string)	DELETE
Determine the starting character of a sub-string	FIND
Insert characters into a string	INSERT
Extract characters from a string	MID

You can also use the following instructions to compare or convert ASCII characters:

If you want to:	Use this instruction:
Compare a string to another string	CMP
See if the characters are equal to specific characters	EQ
See if the characters are not equal to specific characters	NE
See if the characters are equal to or greater than specific characters	GE
See if the characters are greater than specific characters	GT
See if the characters are equal to or less than specific characters	LE
See if the characters are less than specific characters	LT
Rearrange the bytes of an INT, DINT, or REAL tag	SWPB
Find a string in an array of strings	FSC
Convert characters to a SINT, INT, DINT, or REAL value	STOD
Convert characters to a REAL value	STOR
Convert a SINT, INT, DINT, or REAL value to a string of ASCII characters	DTOS
Convert a REAL value to a string of ASCII characters	RTOS

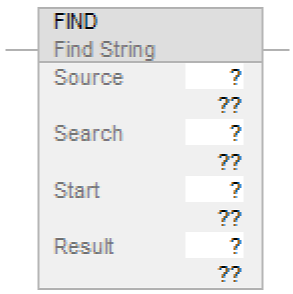
Find String (FIND)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The FIND instruction locates the starting position of a specified string within another string.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

FIND (Source,Search,Start,Result);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	ANY_STRING	Tag	The string to search in	String types are: default STRING data type with max 82 length of characters for the string. any new string type you created with configurable length of characters for the string.
Search	ANY_STRING	Tag	The string to find	
Start	SINT INT DINT	Immediate tag	The position in Source to start the search	Enter a number between 1 and the DATA size of the Source.

Operand	Type	Format	Description	Notes
Result	DINT SINT INT	Tag	The position in Source where search string was found	

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.DescriptionThe FIND instruction searches the Source string for the Search string. If the instruction finds the Search string, the Result shows the starting position of the Search string within the Source string. Otherwise the Results is zero.Affects Math Status FlagsNoMajor/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
The LEN value of the string tag is greater than the DATA size of the string tag.	4	51
The Start value is invalid, or the Source string is empty.	4	56

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand related faults.

Execution

Ladder Diagram

Condition	Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

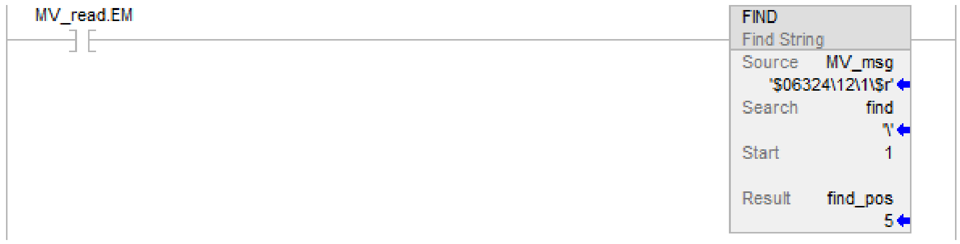
Structured Text

Condition	Action
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table

Example

A message from a MessageView terminal contains several pieces of information. The backslash character (\) separates each piece of information. To locate a piece of information, the FIND instruction searches for the backslash character and records its position in find_pos.

Ladder Diagram



Structured Text

IF MV_read.EM THEN

FIND(MV_msg,find,1,find_pos);

MV_read.EM := 0;

END_IF;

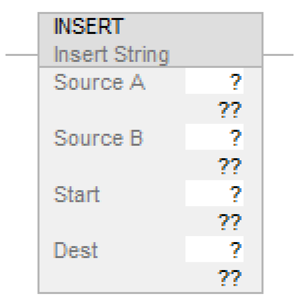
Insert String (INSERT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the INSERT instruction to add ASCII characters to a specified location within a string.

Available Languages

Ladder Diagram



Function Block

Structured Text

INSERT (SourceA,SourceB,Start,Dest);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851. The INSERT instruction uses the following operands.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source A	String type	Tag	String to add the characters to	String types are default STRING data types or any new string types you create
Source B	String type	Tag	String containing the characters to add	
Start	SINT DINT	Immediate tag	Position in Source A to add the characters	Enter a number between 1 and the DATA size of the Source.
Destination	String type	Tag	String to store the result	

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Description

The INSERT instruction adds the characters in Source B to a designated position within Source A and places the result in the Destination.

- Start defines where in Source A that Source B is added.
- Unless Source A and the Destination are the same tag, Source A remains unchanged.

Affects Math Status Flags

No

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag. 2. In the LEN value, enter 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.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	The rung-condition-out is set to false.
Rung-condition-in is false	The rung-condition-out is set to false.

Condition	Ladder Diagram Action
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.
Postscan	The rung-condition-out is set to false.

Execution

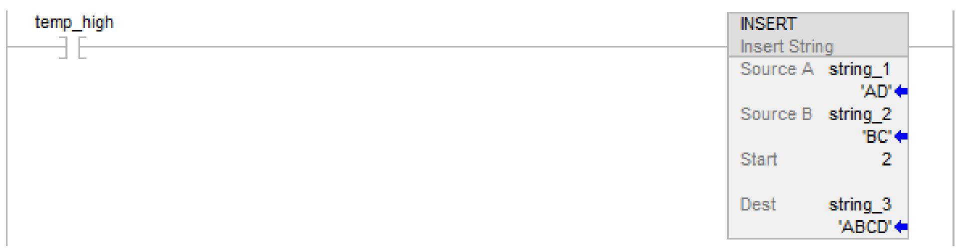
Structured Text

Condition	Action
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table

Example

When *temp_high* is set, the INSERT instruction adds the characters in *string_2* to position 2 within *string_1* and places the result in *string_3*.

Ladder Diagram



Structured Text

IF temp_high THEN

INSERT(string_1,string_2,2,string_3);

temp_high := 0;

END_IF;

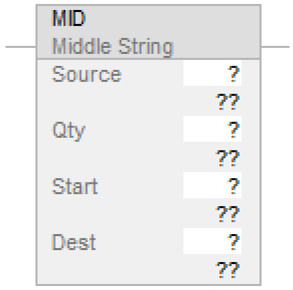
Middle String (MID)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The MID instruction copies a specified number of ASCII characters from a string and stores them in another string.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MID(Source,Qty,Start,Dest);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	ANY_STRING	Tag	The string to copy characters from	String types are: default STRING data type with max 82 length of characters for the string. any new string type you created with configurable length of characters for the string.
Quantity	SINT INT DINT	Immediate tag	The number of characters to copy	The Start plus the Quantity must be less than or equal to the length size of the Source plus 1.
Start	SINT INT DINT	Immediate tag	The position of the first character to copy	Enter a number between 1 and the DATA size of the Source.
Destination	ANY_STRING	Tag	The string to copy the characters to	

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The MID instruction copies a group of characters from the Source and places the result in the Destination.

- The Start position and Quantity define the characters to copy.
- Unless the Source and the Destination are the same tag, the Source remains unchanged.

Affects Math Status Flags

No

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
The LEN value of the Source string tag is greater than the DATA size of the Source string tag.	4	51
The length of output string is larger than the DATA size of the destination string tag.	4	52
The Start or Quantity value is invalid.	4	56

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

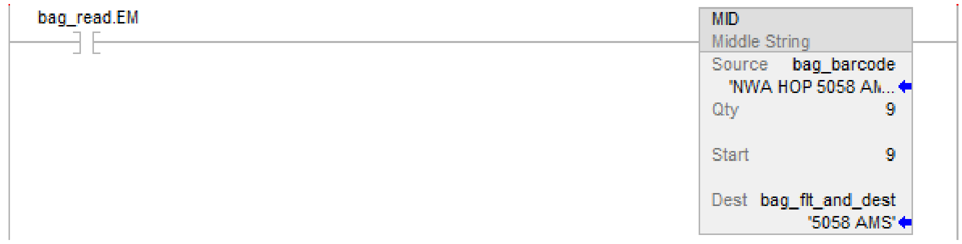
Structured Text

Condition	Action
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table

Example

In the baggage handling conveyor of an airport, each bag gets a bar code. Characters 9 through 17 of the bar code are the flight number and destination airport of the bag. After the bar code is read (bag_read.EM is on), the MID instruction copies the flight number and destination airport to the bag_flt_and_dest string. Subsequent rungs use bag_flt_and_dest to determine where to route the bag.

Ladder Diagram



Structured Text

IF bag_read.EM THEN

MID(bag_barcode,9,9,bag_flt_and_dest);

bag_read.EM := 0;

END_IF;

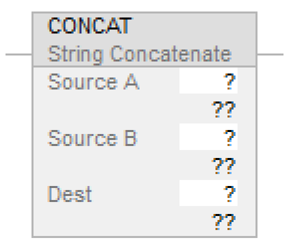
String Concatenate (CONCAT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The CONCAT instruction adds ASCII characters to the end of a string.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

CONCAT(SourceA,SourceB,Dest);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851 for more information on Data Conversion.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source A	ANY_STRING	tag	Tag that contains the initial characters	String types are: • Default STRING data type with maximum 82 length of characters for the string. • Any new string type you created with configurable length of characters for the string.
Source B	ANY_STRING	tag	Tag that contains the end characters	
Destination	ANY_STRING	tag	Tag to store the result	

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The CONCAT instruction combines the characters in Source A with the characters in Source B and places the result in the Destination.

The characters from Source A are first, followed by the characters from Source B.

Unless Source A and the Destination are the same tag, Source A remains unchanged.

Affects Math Status Flags

No

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
The LEN value of the string tag is greater than the DATA size of the string tag.	4	51
The sum length of Source A and Source B is greater than the DATA size of the string tag.	4	51

See Index through arrays on page 863 for array-indexing faults.

Execution

Ladder Diagram

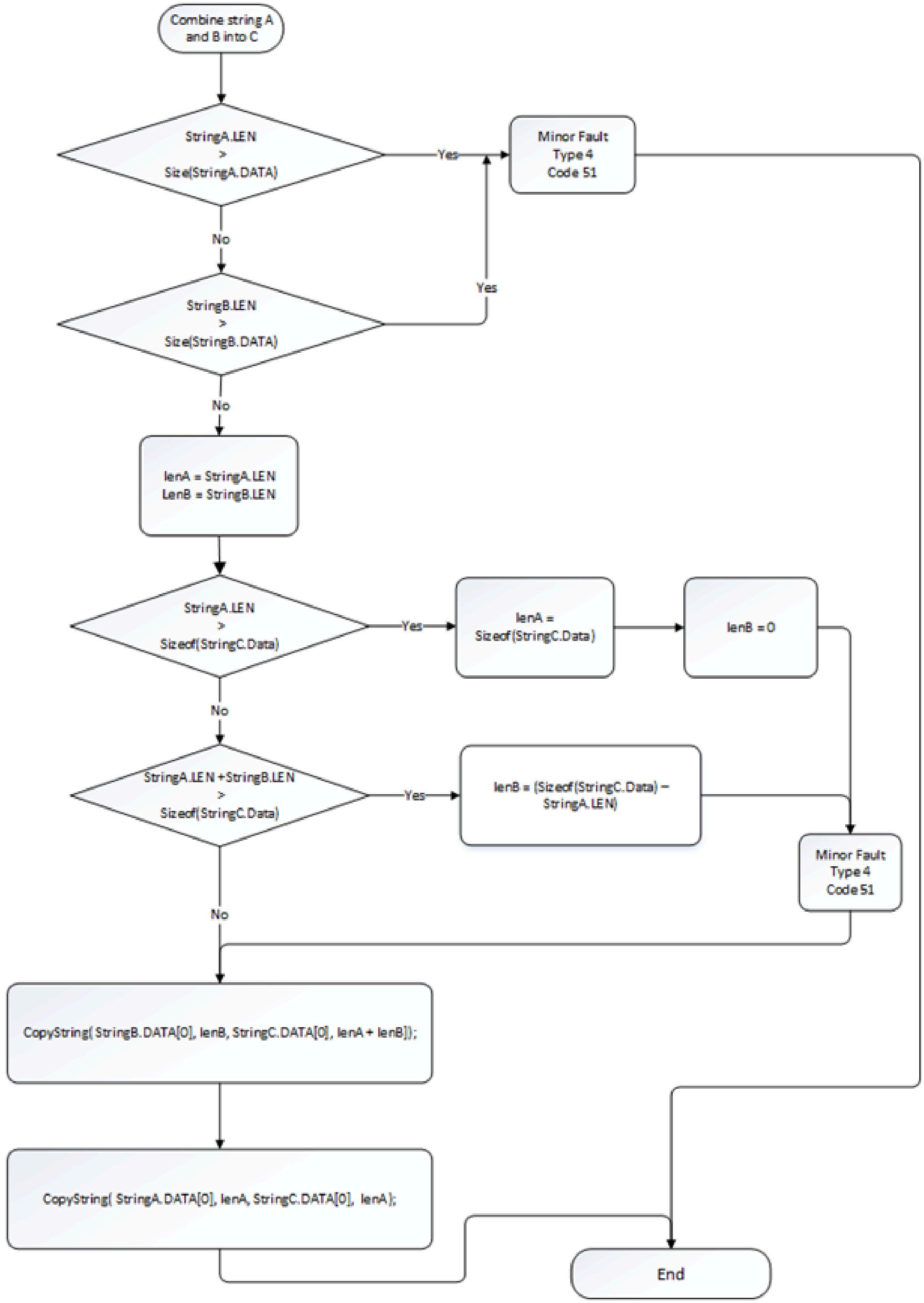
Condition	Action Taken
Prescan	N/A

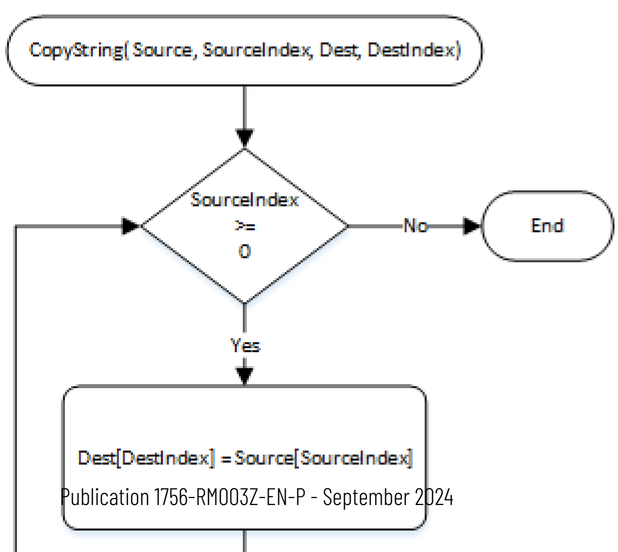
Condition	Action Taken
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

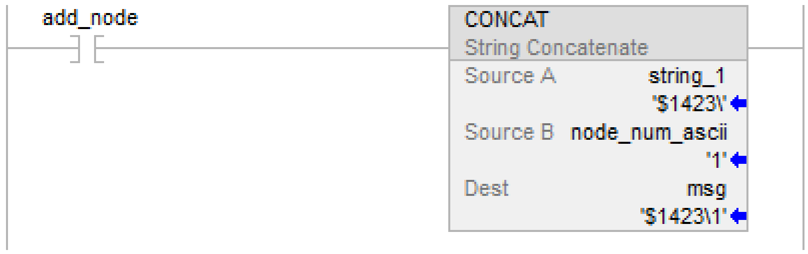
Concat String flow chart





Example

Ladder Diagram



Structured Text

CONCAT(string_1,string_2,msg);

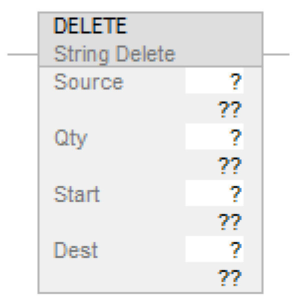
String Delete (DELETE)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The DELETE instruction removes ASCII characters from a string.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

DELETE(Source,Qty,Start,Dest);

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	ANY_STRING	tag	The tag that contains the string from which you want to delete characters	String types are: default STRING data type with max 82 length of characters for the string. any new string type you created with configurable length of characters for the string.
Quantity	SINT INT DINT	immediate tag	The number of characters to delete	The Start plus the Quantity must be less than or equal to the length of the Source plus 1.
Start	SINT INT DINT	immediate tag	The position of the first character to delete	Enter a number between 1 and the DATA size of the Source.
Destination	String type	tag	The tag to store the result	

See Structured Text Syntax on page 879 for more information on the syntax of expressions within structured text.

Description

The DELETE instruction deletes (removes) one or more characters from the Source and places the remaining characters in the Destination.

- The Start position and Quantity define the characters to remove.
- Unless Source A and the Destination are the same tag, Source A remains unchanged.

Affects Math Status Flags

No

Major/Minor Faults

A minor fault will occur if:	Fault Type	Fault Code
The LEN value of the Source string tag is greater than the DATA size of the Source string tag.	4	51
The length of output string is larger than the DATA size of the destination string tag.	4	52
The Start or Quantity value is invalid.	4	56

See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

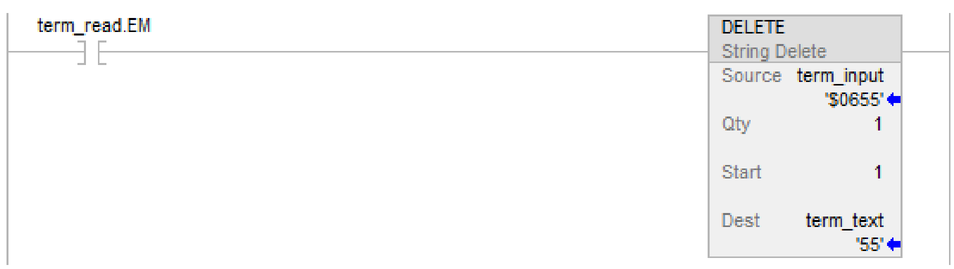
Structured Text

Condition/State	Action
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Examples

ASCII information from a terminal contains a header character. After the controller reads the data (term_read.EM is on), the DELETE instruction removes the header character. The controller can then use the text of the message or pass it on to another device.

Ladder Diagram



Structured Text

IF term_read.EM THEN

DELETE(term_input,1,1,term_text);

term_read.EM := 0;

END_IF;

ASCII Conversion Instructions

Use the ASCII conversion instructions to convert data to or from strings of ASCII characters.

Available Instructions

Ladder Diagram and Structured Text

STOD on page 830	STOR on page 833	RTOS on page 828	DTOS on page 824	LOWER on page 826	UPPER on page 835

Function Block

Not available

If you want to convert:	Use this instruction:
ASCII representations of integer values to SINT, INT, DINT, or REAL values (e.g., converting from a weight scale or other ASCII device to an integer so you can use it in your logic).	STOD
ASCII representations of a floating-point value to a REAL value (e.g., converting a value from a weight scale or other ASCII device to a REAL value so you can use it in your logic).	STOR
SINT, INT, DINT, or REAL values to a string of ASCII characters (e.g., converting a variable to an ASCII string so you can send it to a MessageView™ terminal).	DTOS
REAL values to a string of ASCII characters (e.g., converting a variable to an ASCII string so you can send it to a MessageView terminal).	RTOS
the letters in a string of ASCII characters to upper case (e.g., converting an entry made by an operator to all upper case so you can search for it in an array).	UPPER
the letters in a string of ASCII characters to lower case (e.g., converting an entry made by an operator to all lower case so you can search for it in an array).	LOWER

You can also use the following instructions to compare or manipulate ASCII characters.

If you want to:	Use this instruction:
Add characters to the end of a string	CONCAT
Delete characters from a string	DELETE
Determine the starting character of a sub-string	FIND
Insert characters into a string	INSERT
Extract characters from a string	MID
Rearrange the bytes of an INT, DINT, or REAL tag	SWPB

If you want to:	Use this instruction:
Compare a string to another string	CMP
See if the characters are equal to specific characters	EQ
See if the characters are not equal to specific characters	NE
See if the characters are equal to or greater than specific characters	GE
See if the characters are greater than specific characters	GT
See if the characters are equal to or less than specific characters	LE
See if the characters are less than specific characters	LT
Find a string in an array of strings	FSC

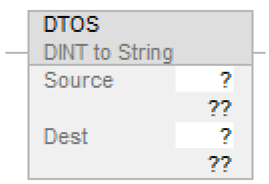
DINT to String-DTOS

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The DTOS instruction produces the ASCII representation of a value.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

DTOS(Source,Dest);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	SINT INT DINT	Tag	The tag that contains the value	If the Source is a REAL, the instruction converts it to a DINT value.

Operand	Type	Format	Description	Notes
	REAL			
Destination	String type	Tag	The tag to store the integer value	String types are: • default STRING data type • any new string type you create

Description

The DTOS instruction converts the Source to a string of ASCII characters and places the result in the Destination.

Affects Math Status Flags

No

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag. 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 type that is large enough for the output string. Use the new string type as the data type for the destination.

See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table

Condition	Action
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Example

When temp_high is set, the DTOS instruction converts the value in msg_num to a string of ASCII characters and places the result in msg_num_ascii. Subsequent rungs insert or concatenate msg_num_ascii with other strings to produce a complete message for a display terminal.

Ladder Diagram

Structured Text

```
IF temp_high THEN
DTOS(msg_num,msg_num_ascii);
temp_high := 0;
END_IF;
```

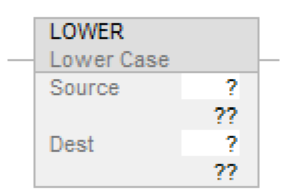
Lower Case-LOWER

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The LOWER instruction converts the alphabetical characters in a string to lower case characters.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

LOWER(Source,Dest);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Source	String	Tag	The tag that contains the characters you want to convert to lower case
Destination	String	Tag	The tag to store the characters in lower case

See *Structured Text* for more information on the syntax of expressions within structured text.

Description

The LOWER instruction converts all the letters in the Source to lower case, and places the result in the Destination.

- ASCII characters are case-sensitive. Upper case A ($41) is not equal to lower case a ($61).
- If operators directly enter ASCII characters, convert the characters to all upper case or lower case before you compare them.

Any characters in the Source string that are not letters remain unchanged.

Affects Math Status Flags

No

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag. 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 type that is large enough for the output string. Use the new string type as the data type for the destination.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

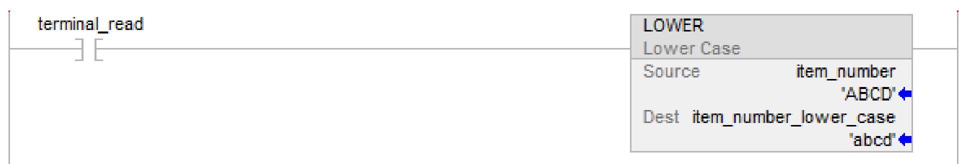
Structured Text

Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Examples

To find information about a specific item, an operator enters the item number into an ASCII terminal. After the controller reads the input from a terminal (terminal_read is set), the LOWER instruction converts the characters in item_number to all upper case characters and stores the result in item_number_lower_case. A subsequent rung then searches an array for characters that match those in item_number_lower_case.

Ladder Diagram



Structured Text

IF terminal_read THEN

LOWER(item_number,item_number_lower_case);

terminal_read := 0;

END_IF;

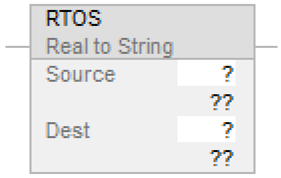
REAL to String (RTOS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The REAL to String (RTOS) instruction produces the ASCII representation of a REAL value.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

RTOS(Source,Dest);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	REAL	Tag	The tag that contains the REAL value	
Destination	String type	Tag	The tag to store the ASCII value	String types are: • Default STRING data type • Any new string type you create

See *Structured Text Syntax* for more information on the syntax of expressions.

Description

The RTOS instruction converts the Source to a string of ASCII characters and places the result in the Destination.

Affects Math Status Flags

No

Major/Minor Faults

Type	Code	Cause	Recovery Method
4	52	The output string is larger than the destination	Create a new string type that is large enough for the output string. Use the new string type as the data type for the destination.

See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Examples

When send_data is set, the RTOS instruction converts the value in data_1 to a string of ASCII characters and places the result in data_1_ascii. Subsequent rungs insert or concatenate data_1_ascii with other strings to produce a complete message for a display terminal.

Ladder Diagram

Structured Text

```
IF send_data THEN

RTOS(data_1,data_1_ascii);

send_data:= 0;

END_IF;
```

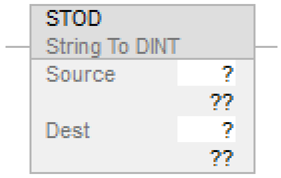
String to DINT (STOD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The String to DINT (STOD) instruction converts the ASCII representation of an integer to an integer or REAL value.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

STOD(Source,Dest);

Operands

There are data conversion rules for mixed data types within an instructions. See *Data Conversion*.

Ladder Diagram and Structured Text

Operand	**Type**	**Format**	**Description**	**Notes**
Source	String type	Tag	The tag that contains the value in ASCII	String types are: • Default STRING data type • Any new string type you create
Destination	SINT INT DINT	Tag	The 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).

See *Structured Text Syntax* for more information on the syntax of expressions.

Description

The STOD instruction converts the Source to an integer and places the result in the Destination.

- The instruction converts positive and negative numbers.
- If the Source string contains non-numeric characters, the STOD converts the first set of contiguous numbers:

The instruction skips any initial control or non-numeric characters, except the minus sign in front of a number.

If the string contains multiple groups of numbers that are separated by delimiters (e.g., /), the instruction converts only the first group of numbers.

Affects Math Status Flags

In Ladder Diagrams only. See *Math Status Flags*.

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag. In the LEN value, enter the number of characters that the string contains.
4	53	The output number is beyond the limits of the destination data type.	• Reduce the size of the ASCII value, or • Use a larger data type for the destination

See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes. Destination is cleared The instruction converts the Source.
Postscan	N/A

Structured Text

Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Example

When MV_read.EM is set, the STOD instruction converts the first set of numeric characters in MV_msg to an integer value. The instruction skips the initial control character ($06) and stops at the delimiter (\).

Ladder Diagram

Structured Text

IF MV_read.EM THEN

STOD(MV_msg,MV_msg_nmbr);

MV_read.EM := 0;

END_IF;

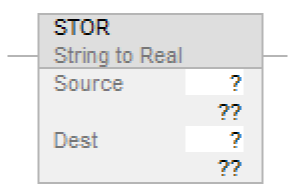
String to REAL (STOR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The String to REAL (STOR) instruction converts the ASCII representation of a floating-point value to a REAL value.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

STOR(Source,Dest);

Operands

There are data conversion rules for mixed data types within an instructions. See *Data Conversion*.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	Notes
Source	String type	tag	The tag that contains the value in ASCII	String types are: • Default STRING data type • Any new string type you create
Destination	REAL	tag	The tag to store the REAL value	

Structured Text for more information on the syntax of expressions within structured text.

Description

The STOR instruction converts the Source to a REAL value and places the result in the Destination.

- The instruction converts positive and negative numbers.
- If the Source string contains non-numeric characters, the STOR converts the first set of contiguous numbers, including the decimal point [.].

The instruction skips any initial control or non-numeric characters (except the minus sign in front of a number).

If the string contains multiple groups of numbers that are separated by delimiters (e.g., /), the instruction converts only the first group of numbers.

Affects Math Status Flags

Conditional, based on programming language. See *Math Status Flags*.

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag. In the LEN value, enter the number of characters that the string contains.
4	53	The output number is beyond the limits of the destination data type.	• Reduce the size of the ASCII value, or • Use a larger data type for the destination

See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

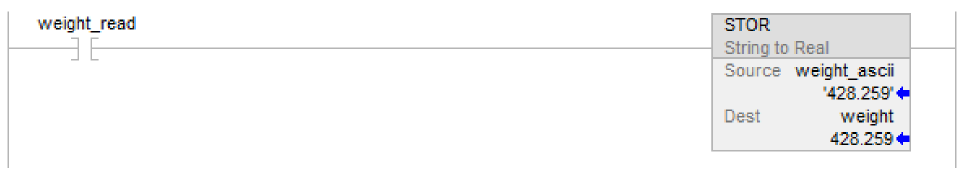
Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Example

After reading the weight from a scale (weight_read is set), the STOR instruction converts the numeric characters in weight_ascii to a REAL value.

You may see a slight difference between the fractional parts of the Source and Destination.

Ladder Diagram



Structured Text

IF weight_read THEN

STOR(weight_ascii,weight);

END_IF;

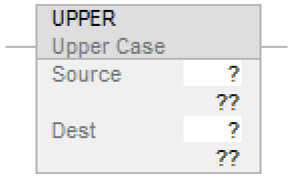
Upper Case (UPPER)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The UPPER instruction converts the alphabetical characters in a string to upper case characters.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

UPPER(Source,Dest);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Source	String	tag	Tag that contains the characters you want to convert to upper case
Destination	String	tag	Tag to store the characters in upper case

See *Structured Text* for more information on the syntax of expressions within structured text.

Description

The UPPER instruction converts all the letters in the Source to upper case, and places the result in the Destination.

- ASCII characters are case-sensitive. Upper case A ($41) is not equal to lower case a ($61).
- If operators directly enter ASCII characters, convert the characters to all upper case or lower case before you compare them.

Any characters in the Source string that are not letters remain unchanged.

Affects Math Status Flags

No

Major/Minor Faults

Type	Code	Cause	Recovery Method
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 type tag.

Type	Code	Cause	Recovery Method
			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 type that is large enough for the output string. Use the new string type as the data type for the destination.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	The instruction executes.
Postscan	N/A

Structured Text

Condition	Action
Prescan	See Prescan in the preceding Ladder Diagram table
Normal execution	See rung-condition-in is true in the preceding Ladder Diagram table.
Postscan	See Postscan in the preceding Ladder Diagram table

Example

To find information about a specific item, an operator enters the catalog number of the item into an ASCII terminal. After the controller reads the input from a terminal (terminal_read is set), the UPPER instruction converts the characters in catalog_number to all upper case characters and stores the result in catalog_number_upper_case. A subsequent rung then searches an array for characters that match those in catalog_number_upper_case.

Ladder Diagram

Structured Text

IF terminal_read THEN

UPPER(catalog_number,catalog_number_upper_case);

```
terminal_read := 0;

END_IF;
```

Debug Instructions

These instructions are compatible only with Studio 5000 Logix Emulate software, which enables emulating a Logix 5000 controller on a personal computer. These instructions are not compatible with emulated 5580 controllers.

Use the debug instructions to monitor the state of the logic when it is in conditions that you determine.

Available Instructions

TPT on page 842 BPT on page 839

Function Block

Not available

Structured Text

Not available

If you want to:	Use this instruction:
Stop program emulation when a rung is true	BPT
Log data you select when a rung is true.	TPT

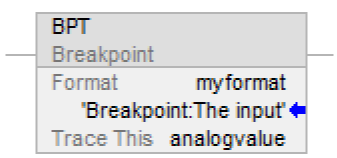
Breakpoints (BPT)

This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

Use the debug instructions to monitor the state of your logic when it is in conditions that you determine.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

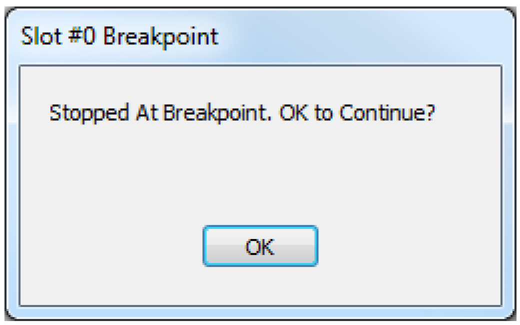
There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Type	Format	Description
Format	String	Tag	A string that sets the formatting for the text that appears in the trace window for the breakpoint.
Trace This	BOOL, SINT, INT, DINT, REAL	Tag	The tag that has a value you want to display in the trace window.

Description

Breakpoints are programmed with the Breakpoint output instruction (BPT). When the inputs on a rung containing a BPT instruction are true, the BPT instruction stops program execution. The software displays a window indicating that the breakpoint triggered and the values that triggered it.



When a breakpoint triggers, the emulator displays a window informing you that a breakpoint occurred. The title bar of the window shows the slot containing the emulator that encountered the breakpoint.

When you click OK, the emulator resumes program execution. If the conditions that triggered the breakpoint persist, the breakpoint will recur.

In addition, the emulator opens a trace window for the breakpoint. The trace window displays information about the breakpoint and the values.

IMPORTANT: When a breakpoint triggers, you will not be able to edit your project until you permit the execution to continue. You can go online with the emulator to observe the state of your project, but you will not be able to edit it. If you try to accept a rung edit while a breakpoint is triggered, you will see a dialog box saying the controller is not in the correct mode.

String Format

With the Format string in the tracepoint and breakpoint instructions, you can control how the traced tags appear in the traces or breakpoint windows. The format of the string is:

- heading:(text)%(type)

where heading is a text string identifying the tracepoint or breakpoint, text is a string describing the tag (or any other text you choose), and %(type) indicates the format of the tag. You need one type indicator for each tag you are tracing with the tracepoint or breakpoint instruction.

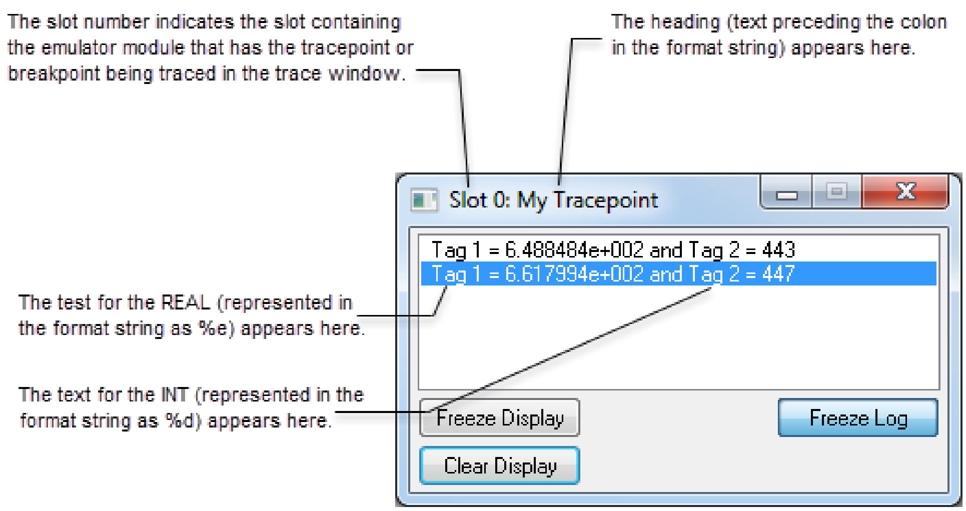
For example, you could format a tracepoint string as shown.

- My tracepoint:Tag 1 = %e and Tag 2 = %d

The %e formats the first traced tag as double-precision float with an exponent, and %d formats the second traced tag as a signed decimal integer.

In this case, you would have a tracepoint instruction that has two Trace This operands (one for a REAL and one for an INT, although the value of any tag can be formatted with any flag).

The resulting tracepoint window that would appear when the tracepoint is triggered would look like the example.



Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

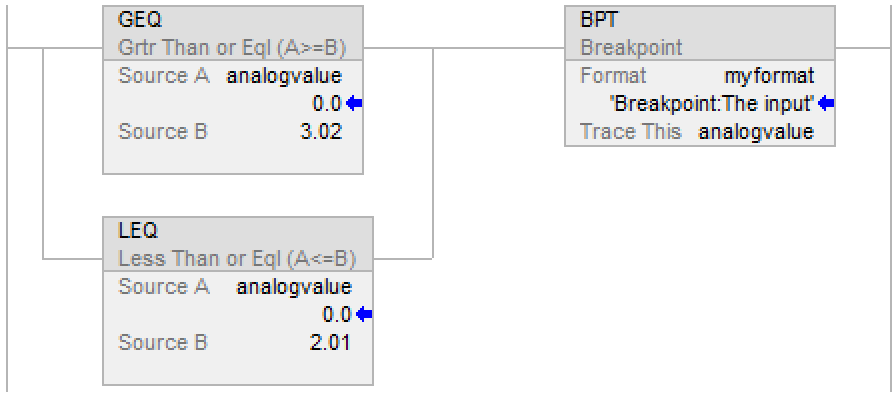
Execution

Condition	Action Taken
Prescan	The rung becomes false.
Rung-condition-in is false	The rung becomes false.
Rung-condition-in is true	The rung becomes true. Execution jumps to the rung that contains the LBL instruction with the referenced label name.
Postscan	The rung becomes false.

Examples

You can display many tag values with the BPT instruction. However, the formatting string can contain only 82 characters. Because the formatting string requires two characters for each tag you want in the breakpoint, you cannot trace more than 41 tags with a single BPT instruction. However, to separate tag data in your traces, you will need to include spaces and other formatting, thus reducing the number of tag values that one BPT instruction can effectively display to far fewer than 41.

This rung shows a breakpoint that stops program execution when an analog value is greater than 3.02 or less than 2.01.

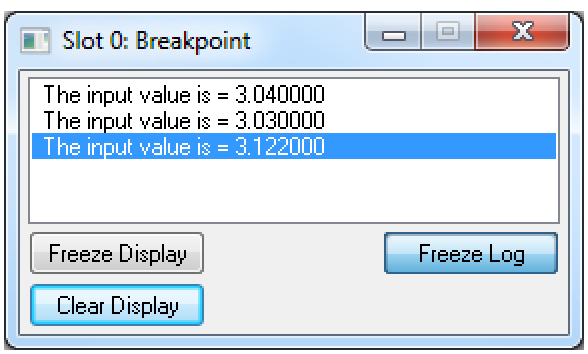


Display the breakpoint information in the Format string (myformat). In this case, the format string contains the following text:

- Breakpoint:The input value is %f

When the breakpoint triggers, the breakpoint trace window shows the characters before the colon ('Breakpoint') in the title bar of the trace window. The other characters make up the traces. In this example, %f represents the first (and in this case, the only) tag to be traced ('analogvalue').

The resulting traces appear as shown here.



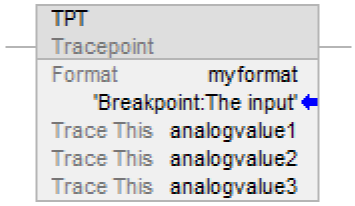
Tracepoints (TPT)

This instruction is compatible with Studio 5000 Logix Emulate controllers only. This instruction is not compatible with emulated 5580 controllers.

Tracepoints log data you select when a rung is true.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

This instruction is not available in structured text.

Operands

There are data conversion rules for mixed data types within an instruction. See Data conversions on page 851.

Ladder Diagram

Operand	Type	Format	Description
Format	String	Tag	A string that sets the formatting for the trace reports (both on-screen and logged to disk).
Trace This	BOOL SINT INT DINT REAL	Tag	The tag you want to trace.

Description

Tracepoints are programmed with the tracepoint output instruction (TPT). When the inputs on a rung containing a TPT instruction are true, the TPT instruction writes a trace entry to a trace display or log file.

You can trace many tags with the TPT instruction. However, the formatting string can contain only 82 characters. Because the formatting string requires two characters for each tag you want to trace, you cannot trace more than 41 tags with a single TPT instruction. However, to separate tag data in your traces, you will need to include spaces and other formatting, thus reducing the number of tags that one TPT instruction can effectively trace to far fewer than 41.

String Format

With the Format string in the tracepoint and breakpoint instructions, you can control how the traced tags appear in the traces or breakpoint windows. The format of the string is as shown here:

- heading:(text)%(type)

where heading is a text string identifying the tracepoint or breakpoint, text is a string describing the tag (or any other text you choose), and %(type) indicates the format of the tag. You need one type indicator for each tag you are tracing with the tracepoint or breakpoint instruction.

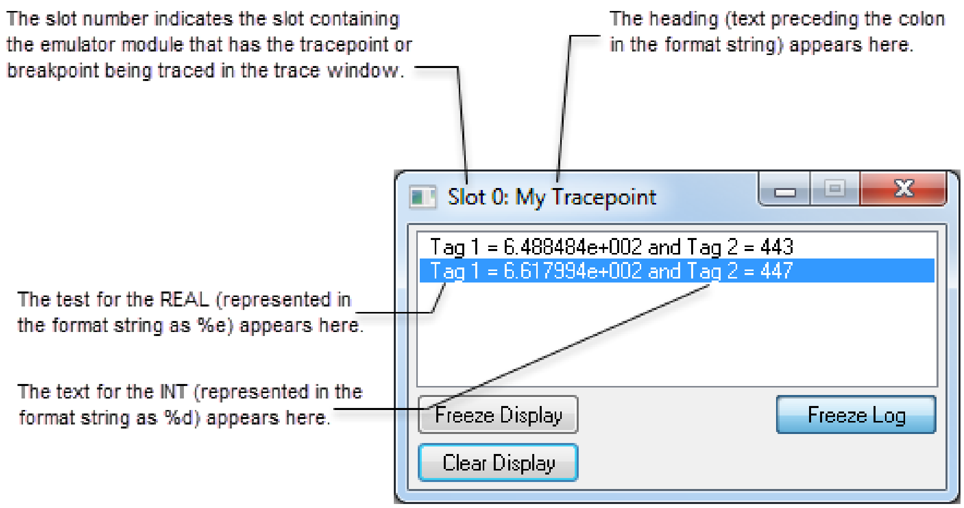
For example, you could format a tracepoint string as shown:

- My tracepoint:Tag 1 = %e and Tag 2 = %d

The %e formats the first traced tag as double-precision float with an exponent, and %d formats the second traced tag as a signed decimal integer.

In this case, you have a tracepoint instruction that has two Trace This operands (one for a REAL and one for an INT, although the value of any tag can be formatted with any flag).

The resulting tracepoint window that would appear when the tracepoint is triggered would look like the example.



Affects Math Status Flags

No

Major/Minor Faults

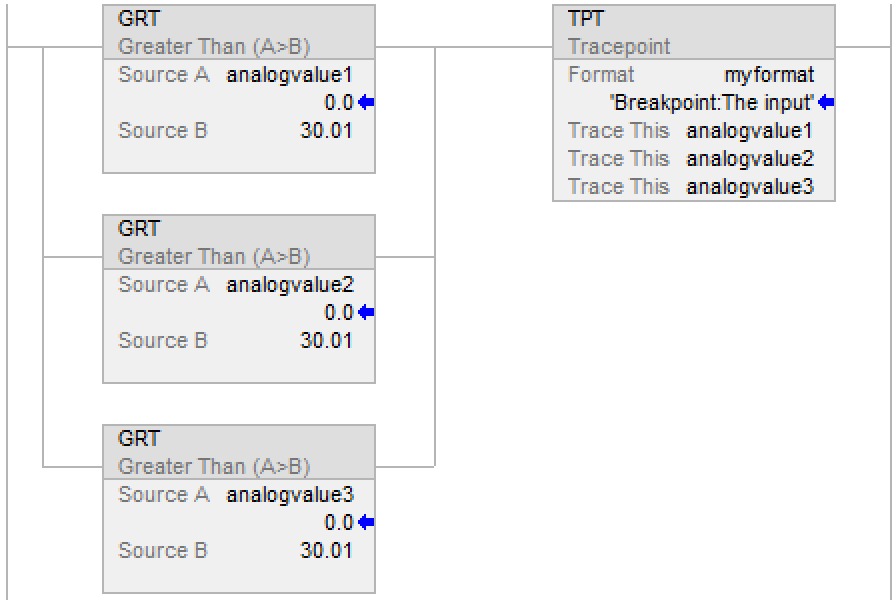
None specific to this instruction. See Common Attributes for General Instructions on page 849 for operand-related faults.

Execution

Condition	Relay Ladder Action
Prescan	The rung becomes false.
Rung-condition-in is false	The rung becomes false.
Rung-condition-in is true	The rung becomes true. Execution jumps to the rung that contains the LBL instruction with the referenced label name.
Postscan	The rung becomes false.

Example

This rung triggers a trace of three analog values when any one of them exceeds a given value (30.01).



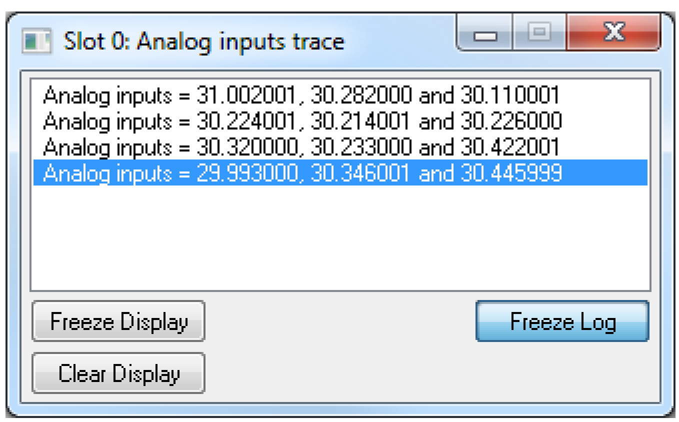
Display the tracepoint information in the Format string (myformat).

In this case, the format string contains this text:

- Analog inputs trace:Analog inputs = %f, %f, and %f

When the tracepoint triggers, the characters before the colon ('Analog inputs trace') appear in the title bar of the trace window. The other characters make up the traces. In this example, %f represents the tags to be traced ('analogvalue1,' 'analogvalue2,' and 'analogvalue3').

The resulting traces appear as shown here.



When this trace is logged to disk, the characters before the colon appear in the traces.

This indicates which tracepoint caused which trace entry. This is an example of a trace entry. 'Analog inputs trace:' is the heading text from the tracepoint's format string.

Analog inputs trace: Analog inputs = 31.00201, 30.282000, and 30.110001.

License Instructions

Use the License instruction to verify licenses in a project.

Available Languages

Ladder Diagram

LV on page 847

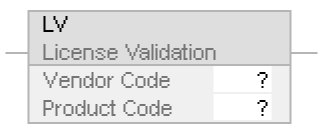
License Validation (LV)

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The License Validation (LV) instruction verifies if a non-expired license associated with a routine or Add-On Instruction is present in the controller.

Available Languages

Ladder Diagram



Function Block

Not available

Structured Text

Not available

Operands

Ladder Diagram

Operand	Type	Format	Description
Vendor Code	DINT	immediate	Unique number identifying the vendor of the license associated with a routine or Add-On Instruction. Accepts an immediate integer value in the range of 0 to 2,147,483,647.

Operand	Type	Format	Description
Product Code	DINT	immediate	Unique number identifying the product code of the license associated with a routine or Add-On Instruction. Accepts an immediate integer value in the range of 0 to 2,147,483,647.

Affects Math Status Flags

No

Major/Minor Faults

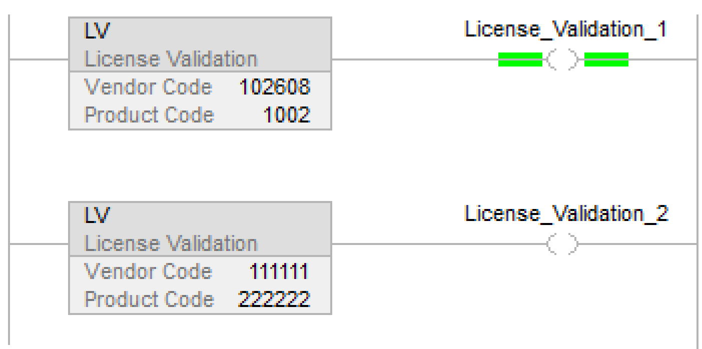
None specific to this instruction.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	N/A
Rung-condition-in is false	N/A
Rung-condition-in is true	Numeric compare" If the license is valid and used in the project Set Rung-condition-out to true else Clear Rung-condition-out to false
Postscan	N/A

Example



Common Attributes for General Instructions

Follow the guidelines in this chapter for the common attributes for the General Instructions.

Math status flags

Follow these guidelines for Math Status Flags.

Description

A set of Math Status Flags for accessing directly with instructions. These flags are only updated in ladder diagram routines, and are not tags, and flag aliases are not applicable.

Status Flags

This table describes that specific status flags.

Status Flag	Description
S:FS First scan flag	The first scan flag is set by the controller: • The first time a program is scanned after the controller goes to Run mode • The first time a program is scanned after the program is uninhibited • When a routine is called from an SFC Action and the step that owns that Action is first scanned. Use the first scan flag to initialize data for use in later scans. It is also referred to as the first pass bit.
S:N Negative flag	The controller sets the negative flag when the result of a math or logical operation is a negative value. Use this flag as a quick test for a negative value.
S:Z Zero flag	The zero flag is set by the controller when the result of a math or logical operation is zero. Use this flag as a quick test for a zero value. The zero flag clears at the start of executing an instruction capable of setting this flag.
S:V Overflow flag	The controller sets the overflow flag when: • The result of a math operation results in an overflow. For example, adding 1 to a SINT generates an overflow when the value goes from 127 through -128. • The destination tag is too small to hold the value. For example, if you try to store the value 123456 to a SINT or INT tag. Use the overflow flag to verify the result of an operation is still in range.

Status Flag	Description
	If the data being stored is a string type, S:V is set if the string is too large to fit into the destination tag. If applicable, set S:V with an OTE or OTL instruction. Select **Controller Properties > Advanced tab > Report Overflow Faults** to enable or disable reporting overflow faults. If an overflow occurs while evaluating an array subscript, a minor fault is generated and a major fault is generated to indicate the index is out of range.
S:C Carry flag	The controller sets the carry flag when the result of a math operation resulted in the generation of a carry out of the most significant bit. Only the ADD and SUB instructions, and not the + and – operators, with integer values affect this flag.
S:MINOR Minor fault flag	The controller sets the minor fault flag when there is at least one minor program fault. Use the minor fault tag to test if a minor fault occurred. This bit only triggers by programming faults, such as overflow. It is not triggered by a battery fault. The bit clears at the beginning of every scan. If applicable, explicitly set S:MINOR with an OTE or OTL instruction.

IMPORTANT: The math status flags are set based on the stored value. Instructions that normally do not affect math status flags might appear to affect math status flags if type conversion occurs from mixed data types for the instruction parameters. The type conversion process sets the math status flags.

Expressions in Array Subscripts

Expressions do not set status flags based on the results of math operations. If expressions overflow:

- A minor fault generates if the controller is configured to generate minor faults.
- A major fault (type 4, code 20) generates because the resulting value is out of range.

Tip: If an array subscript is too large (out of range), a major fault (type 4, code 20) generates.

Immediate values

When you enter an immediate value (constant) in decimal format (for example, -2, 3) the controller stores the value by using 32 bits. If you enter a value in a radix other than decimal, such as binary or hexadecimal, and do not specify all 32 bits, the controller places a zero in the bits that you do not specify (zero-fill).

IMPORTANT: Zero-fill of immediate binary, octal or hexadecimal values less than 32 bits.

If you enter	The controller stores
-1	16#ffff ffff (-1)
16#ffff (-1)	16#0000 ffff (65535)
8#1234 (668)	16#0000 029c (668)
2#1010 (10)	16#0000 000a (10)

Integer Immediate Values

If you enter	The controller stores
Without any suffix	DINT
"U"	UDINT
"L"	LINT
"UL"	ULINT

Floating Point Immediate Values

If you enter	The controller stores
Without any suffix	REAL
"L"	LREAL

Data conversions

Data conversions occur when mixing data types in programming. When programming ladder diagram, mix data types for the parameters within one instruction or expression.

Instructions execute faster and require less memory if all the operands of the instruction use:

- The same data type.
- An intermediate data type:
    - If mixing data types or use tags that are not the optimal data type, the controller converts the data according to these rules:
        - Operands are converted according to the ranking of data types from SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, and LREAL with ranking from 1 (the lowest) to 10 (the highest).

Tip: To reduce the time and memory for converting data, use the same data type for all the operands of an instruction.

Convert SINT or INT to DINT or DINT to LINT

A SINT or INT input source tag gets promoted to a DINT value by a sign-extension for Source Tag. Instructions that convert SINT or INT values to DINT values use one of the following conversion methods.

This conversion method	Converts data by placing
Sign-extension	The value of the leftmost bit (the sign of the value) into each bit position to the left of the existing bits until there are 32 or 64 bits.

This conversion method	Converts data by placing
Zero-fill	Zeros to the left of the existing bits until there are 32 or 64 bits.

Logical instructions use zero fill. All other instructions use sign-extension

The following example shows the results of converting a value using sign- extension and zero-fill.

This value	2#1111_1111_1111_1111	(-1)
Converts to this value by sign-extension	2#1111_1111_1111_1111_1111_1111_1111_1111	(-1)
Converts to this value by zero-fill	2#0000_0000_0000_0000_1111_1111_1111_1111	(65535)

If you use a SINT or INT tag and an immediate value in an instruction that converts data by sign-extension, use one of these methods to handle immediate values.

Specify any immediate value in the decimal radix.

If you enter the value in a radix other than decimal, specify all 32 bits of the immediate value. To do so, enter the value of the leftmost bit into each bit position to its left until there are 32 bits.

Create a tag for each operand and use the same data type throughout the instruction. To assign a constant value, either:

Enter it into one of the tags.

Add a MOV instruction that moves the value into one of the tags.

Use a MEQ instruction to check only the required bits.

The following examples show two ways to mix an immediate value with an INT tag. Both examples check the bits of a 1771 I/O module to determine if all the bits are on. Since the input data word of a 1771 I/O module is an INT tag, it is easiest to use a 16-bit constant value.

IMPORTANT:

- When mixing an INT tag with an immediate value, since remote_rack_1:I.Data[0] is an INT tag, the value to check it against is also entered as an INT tag. When mixing an INT tag with an immediate value, since remote_rack_1:I.Data[0] is an INT tag, the value to check it against is also entered as an INT tag.
- When mixing an INT tag with an immediate value, since remote_rack_1:I.Data[0] is an INT tag, the value to check it against first moves into int_0, also an INT tag. The EQU instruction then compares both tags.

Convert Integer to REAL

The controller stores REAL values in IEEE single-precision, floating-point number format. It uses one bit for the sign of the value, 23 bits for the base value, and eight bits for the exponent (32 bits total). If you mix an integer tag (SINT,

INT, or DINT) and a REAL tag as inputs in the same instruction, the controller converts the integer value to a REAL value before the instruction executes.

- A SINT or INT value always converts to the same REAL value.
- A DINT value may not convert to the same REAL value:
- A REAL value uses up to 24 bits for the base value (23 stored bits plus a 'hidden' bit).
- A DINT value uses up to 32 bits for the value (one for the sign and 31 for the value).

If the DINT value requires more than 24 significant bits, it might not convert to the same REAL value. If it will not, the controller stores the uppermost 24 bits rounded to the nearest even value.

NOTE: The Logix Designer application interprets numbers differently depending on whether the controller model is a 5x80 controller or a 5x70 controller. For example:

- For a 5x70 controller, Logix Designer interprets literal 2 as a REAL.
- For a 5x80 controller, Logix Designer interprets literal 2 as a DINT.

Convert DINT to SINT or INT

To convert a DINT value to a SINT or INT value, the controller truncates the upper portion of the DINT and stores the lower bits that fit in the data type. If the value is too large the conversion generates an overflow.

Convert a DINT to an INT and a SINT

This DINT value	Converts to this smaller value	
16#0001_0081 (65,665)	INT	16#0081 (129)
	SINT	16#81 (-127)

Convert REAL to SINT, INT, or DINT

To convert a REAL value to an integer value, the controller rounds any fractional part and stores the bits that fit in the result data type. If the value is too large the conversion generates an overflow.

Numbers round as in the following examples.

Fractions < 0.5 round down to the nearest whole number.

Fractions > 0.5 round up to the nearest whole number.

Fractions = 0.5 round up or down to the nearest even number.

Conversion of REAL values to DINT values

This REAL value	Converts to this DINT value
-2.5	-2
-3.5	-4
-1.6	.2
-1.5	.2
-1.4	.1

This REAL value	Converts to this DINT value
1.4	1
1.5	2
1.6	2
2.5	2
3.5	4

Elementary data types

The controller supports the elementary data types defined in IEC 1131-3 defined data types. The elementary data types are:

Data type	Description	Range
BOOL	1-bit boolean	0 = cleared 1 = set
SINT	1-byte integer	-128 to 127
INT	2-byte integer	-32,768 to 32,767
DINT	4-byte integer	-2,147,483,648 to 2,147,483,647
REAL	4-byte floating-point number	$-3.402823E^{38}$ to $-1.1754944E^{-38}$ (negative values) and 0 and $1.1754944E^{-38}$ to $3.402823E^{38}$ (positive values)
LINT	8-byte integer	0 to 32,535,129,599,999,999
USINT	1-byte unsigned integer	0 to 255
UINT	2-byte unsigned integer	0 to 65,535
UDINT	4-byte unsigned integer	0 to 4,294,967,295
ULINT	8-byte unsigned integer	0 to 18,446,744,073,709,551,615
REAL	4-byte floating-point number	-3.4028235E38 to -1.1754944E-38 (negative values) and 0.0 and 1.1754944E-38 to 3.4028235E38 (positive values)
LREAL	8-byte floating-point number	-1.7976931348623157E308 to

Data type	Description	Range
		-2.2250738585072014E-308 (negative values) and 0.0 and 2.2250738585072014E-308 to 1.7976931348623157E308 (positive values)

The controller handles all immediate values as DINT data types.

Data type conversions

<table>
<tr><td>Conversion</td><td colspan="3">Result</td></tr>
<tr><td rowspan="5">Larger integer to smaller integer</td><td colspan="3">The controller truncates the upper portion of the larger integer and generates an overflow.
For example:</td></tr>
<tr><td colspan="2">Decimal</td><td>Binary</td></tr>
<tr><td>DINT</td><td>65,665</td><td>0000_0000_0000_0001_0000_
0000_1000_0001</td></tr>
<tr><td>INT</td><td>129</td><td>0000_0000_1000_0001</td></tr>
<tr><td>SINT</td><td>-127</td><td>1000_0001</td></tr>
<tr><td>SINT or INT to REAL</td><td colspan="3">No data precision is lost</td></tr>
<tr><td>DINT to REAL</td><td colspan="3">Data precision could be lost. Both data types store data in 32 bits, but the REAL type uses some of its 32 bits to store the exponent value. If precision is lost, the controller takes it from the least-significant portion of the DINT.</td></tr>
<tr><td>LREAL to LREAL</td><td colspan="3">No data precision is lost.</td></tr>
<tr><td>LREAL TO REAL</td><td colspan="3">Data precision could be lost.</td></tr>
<tr><td>LREAL/REAL to unsigned integer</td><td colspan="3">Data precision could be lost. If the source value is too big to fit into destination the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>Signed Integer/Unsigned Integer to LREAL/REAL</td><td colspan="3">If the integer value has more significant bits than can be stored in the destination, the lower bits will be truncated.</td></tr>
<tr><td>Signed integer to unsigned integer</td><td colspan="3">If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>Unsigned integer to signed integer</td><td colspan="3">If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.</td></tr>
<tr><td>REAL to integer</td><td colspan="3">The controller rounds the fractional part and truncates the upper portion of the non-fractional part. If data is lost, the controller sets the overflow status flag.
Rounding is to the nearest whole number:
less than 0.5, round down; equal to 0.5, round to nearest even integer; greater than 0.5, round up
For example:</td></tr>
</table>

REAL (source)	DINT (result)
1.6	2
-1.6	-2
1.5	2
-1.5	-2
1.4	1
-1.4	-1
2.5	2
-2.5	-2

Do not convert data to or from the BOOL data type.

IMPORTANT: The math status flags are set based on the value being stored. Instructions that normally do not affect math status keywords might appear to do so if type conversion occurs because of mixed data types for the instruction parameters. The type conversion process sets the math status keywords.

Safety Data Types

The Logix Designer application prevents the modification of a User Defined or Add-On Defined type that would cause an invalid data type for User Defined or Add-On Defined types that are referenced directly or indirectly by a Safety tag. (This includes nested structures.)

Safety tags can be composed of the following data types:

- All elementary data types.
- Predefined types that are used for safety application instructions.
- User-defined data types or arrays that are composed of the previous two types.

Online edits of user-defined data type member names in safety tags

Online editing is allowed for member names of user-defined data types on CompactLogix 5380, Compact GuardLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. However, online editing is disabled when a user-defined data type is used on a safety tag and the controller is in the Safety Secured state.

Related information

Math status flags on page 849

Pseudo-operand initialization

Pseudo-operands are placeholders for instruction backing tag structure members.

IMPORTANT: When you use an instruction backing tag for a safety-critical operation, you must initialize the pseudo-operands during first scan. Refer to the GuardLogix 5580 and Compact GuardLogix 5380 Controller Systems Safety Reference, publication 1756-RM012, for methods to initialize pseudo-operands during first scan.

Pseudo-operands are initialized when the application is downloaded and never again, unless modified by the application.

- When you specify a pseudo-operand value it is written directly to the member. If you use the same backing tag in another instruction and specify a different value, the previous value is overwritten by the new value.
- Position (POS) and Accumulator (ACC) are initialized as described but they are also overwritten by the instruction when it executes.

  For example:
  - A false Timer on delay (TON) instruction sets the ACC to 0.
    - A true TON calculates the elapsed time and adds this to the ACC.
    - Preset (PRE) is used by the TON to determine when the DN bit should be set. The instruction does not change the PRE member.
  - When a LIFO Load (LFL) instruction executes (false-to-true transition), the source value is written to the LIFO and the POS is incremented.
  - When a LIFO Unload (LFU) instruction executes, the value at array[POS] is read and the POS is decremented.

This table lists the pseudo-operands.

Tip: ASCII Serial Port instructions (AWT, AWA, ARD, ARL, ABL, ACB, AHL, and ACL) are available only on controllers that have serial ports. Logix Designer versions 37 and later do not support ASCII Serial Port instructions.

Instructions	Pseudo-operands	Allowed in safety routines
ASCII Test for Buffer Line (ABL)	POS	No
ASCII Chars in Buffer (ACB)	POS	No
ASCII Handshake Lines (AHL)	POS	No
ASCII Read (ARD)	LEN, POS	No
ASCII Read Line (ARL)	LEN, POS	No
File Average (AVE)	LEN, POS	No
ASCII Write Append (AWA)	LEN, POS	No
ASCII Write (AWT)	LEN, POS	No
Bit Shift Left (BSL)	LEN	No
Bit Shift Right (BSR)	LEN	No
Count Up (CTU)	PRE, ACC	Yes
Count Down (CTD)	PRE, ACC	Yes
Diagnostic Detect (DDT)	LEN, POS	No
File Arithmetic and Logic (FAL)	LEN, POS	Yes
File Bit Comparison (FBC)	LEN, POS	No
FIFO Load (FFL)	LEN, POS	No
FIFO Unload (FFU)	LEN, POS	No
File Search and Compare (FSC)	LEN, POS	Yes

LIFO Load (LFL)	LEN, POS	No
LIFO Unload (LFU)	LEN, POS	No
Retentive Timer On (RTO)	PRE, ACC	Yes
Sequencer Input (SQI)	LEN, POS	No
Sequencer Load (SQL)	LEN, POS	No
Sequencer Output (SQO)	LEN, POS	No
File Sort (SRT)	LEN, POS	No
File Standard Deviation (STD)	LEN, POS	No
Timer Off Delay (TOF)	PRE, ACC	Yes
Timer On Delay (TON)	PRE, ACC	Yes

Time and date data types

Use time and date data types to standardize time and date values in Logix5000 control systems. Standardized time and date values increase the accuracy and reliability of time-stamped inputs, scheduled outputs, and time-based registration for motion control. They also help increase accuracy for Sequence of Events, Time-stamped Data Logging and Analytics, and Time Synchronization within and across systems.

In the ladder editor, time and date data types are supported in these instructions: ADD, CLR, EQ, GE, GSV, GT, JSR, LE, LT, MOVE, NE, RET, SBR, SUB, and SSV.

In Structured Text (ST), you can use Time and date data types with these single operator expressions and instructions:

- Single operator expressions: +, -, =, >=, >, <=, <, and <>
- Instructions: GSV, JSR, RET, SBR, and SSV

In the Function Block Diagram (FBD) editor, time and date data types are supported in these functions and instructions:

- Functions: ADD__F, SUB__F, MOVE (IREF->OREF), EQ__F, GE__F, GT__F, LE__F, LT__T, and NE__F
- Instructions: JSR, SBR, and RET

Absolute time data types

Use these absolute time data types for a specific point in time.

Data type	Description
DT	Date and time. 64-bit storage; units are in microseconds.
LDT	Long date and time. 64-bit storage; units are in nanoseconds.

Relative time data types

Use these relative time data types for a duration or length of time.

Data type	Description
TIME32	Duration of time. 32-bit storage; units are in microseconds.

Data type	Description
TIME	Duration of time. 64-bit storage; units are in microseconds.
LTIME	Long duration of time. 64-bit storage; units are in nanoseconds.

Considerations

Keep these considerations in mind when using relative time (LTIME, TIME32, TIME) and absolute time (LDT, DT) data types:

- Use the Move (MOVE) instruction as a bridge between systems adopting time and date data types and legacy systems. Using time and date data types and LINT data types with MOVE allows the Logix Designer application to carry out a straight memory copy.
- You cannot mix time and date operands with any other kind of data type except LINT. LINT data types were often used in legacy systems for time stamping, so they are the only data types that are interoperable with the time and date data types. The system allows LINTs to be used broadly but it assumes that every LINT is an LDT data type, and type conversion occurs based on that assumption. Systems using LINT microsecond tags would need to:
  - Manage that discrepancy wherever a LINT microsecond tag is used, or
  - MOVE its value to a DT tag, or
  - Convert the LINT microsecond tag to nanoseconds and then MOVE that value to an LDT tag.
- For Add (ADD), Subtract (SUB), and Compare Instructions:
  - If both Source A and Source B are relative time, the Dest must be relative time.
  - If Source A is relative time and Source B is absolute time or vice versa, the Dest must be absolute time.
  - In ADD instructions, Source A and Source B cannot both be absolute time.

Relative time formats

Literal string and Tag display formats:

```
T32#2d_3h_1m_22s_123ms_678us  T#8h_33s_234ms_679us

LT#10s_522ms
```

You can modify relative time literal strings directly inline.

You can modify relative time tags directly inline or in the **Time Browser**. To use the **Time Browser** in a routine, double-click the tag value. In the **Data Monitor**, select the ellipsis to open the **Time Browser** or replace any portion of the relative time tag string using its literal string format.

Absolute time formats

Literal string and Tag display formats:

```
DT#2020-03-05-08:11:44.345_678  LDT#2020-10-25-11:05:20.123_456_789
```

You can modify absolute time literal strings directly inline.

You can modify absolute time tags directly inline or in the **Time Browser**. To use the **Time Browser** in a routine, double-click the tag value. In the **Data Monitor**, select the ellipsis to open the Time Browser or replace any portion of the absolute time tag string using its literal string format.

GSV and SSV objects that support time and date data types

Use time and date data types to standardize time and date values in Logix control systems.

These tables list the Get System Value (GSV) and Set System Value (SSV) objects that support time and date data types. See GSV/SSV Objects on page 271 for a list of data types supported by each attribute.

GSV object attributes that support time and date

Object	Attribute
CST	CurrentValue
Message	ConnectionRate
	UnconnectedTimeout
Axis	Registration 1 Time
	Registration 2 Time
	Interpolation Time
MotionGroup	CycleStartTime
	MaximumInterval
	MinimumInterval
	StartTime
	TaskAverageIOTime
	TaskAverageScanTime
	TaskLastIOTime
	TaskLastScanTime
	TaskMaximumIOTime
	TaskMaximumScanTime
	TimeOffset
Program	LastScanTime
	MaxScanTime
Task	LastScanTime
	MaximumInterval
	MaxScanTime
	MinimumInterval
	StartTime
TimeSynchronize	CurrentTimeMicroseconds
	CurrentTimeNanoseconds
WallClockTime	CSToffset
	CurrentValue

SSV object attributes that support time and date

Object	Attribute
Message	ConnectionRate
	UnconnectedTimeout
Axis	Interpolation Time
MotionGroup	MaximumInterval
	TaskAverageIOTime
	TaskAverageScanTime
	TaskMaximumIOTime
	TaskMaximumScanTime
Program	LastScanTime
	MaxScanTime
Task	LastScanTime
	MaxScanTime
	MinimumInterval
	StartTime
WallClockTime	CToffset
	CurrentValue

Floating Point Values

Logix controllers handle floating point values according to the IEEE 754 standard for floating-point arithmetic. This standard defines how floating point numbers are stored and calculated. The IEEE 754 standard for floating point math was designed to provide speed and the ability to handle very large numbers in a reasonable amount of storage space.

A REAL tag stores a single-precision, normalized floating-point number.

An LREAL tag stores a double-precision, normalized floating-point number.

The controllers support these elementary data types:

- REAL
- LREAL

Denormalized numbers and -0.0 are treated as 0.0

If a computation results in a NAN value, the sign bit could be positive or negative. In this situation, the software displays 1#.NAN with no sign.

Not all decimal values can be exactly represented in this standard format, which results in a loss of precision. For example, if you subtract 10 from 10.1, you expect the result to be 0.1. In a Logix controller, the result could very well be 0.10000038. In this example, the difference between 0.1 and 0.10000038 is .000038%, or practically zero. For most operations, this small inaccuracy is insignificant. To put things in perspective, if you were sending a floating point value to an analog output module, there would be no difference in the output voltage for a value being sent to the module that differs by .000038%.

Guidelines for Floating-point Math Operations

Follow these guidelines:

When performing certain floating-point math operations, there may be a loss of precision due to rounding error. Floating-point processors have their own internal precision that can impact resultant values.

Do not use floating point math for money values or for totalizer functions. Use INT or DINT values, scale the values up, and keep track of the decimal place (or use one INT or DINT value for dollars, and a second INT or DINT value for cents).

Do not compare floating-point numbers. Instead, check for values within a range. The LIMIT instruction is provided specifically for this purpose.

Totalizer Examples

The precision of the REAL data type affects totalization applications such that errors occur when adding very small numbers to very large numbers.

For example, add 1 to a number over a period of time. At some point the add will no longer affect the result because the running sum is much greater than 1, and there are not enough bits to store the entire result. The add stores as many upper bits as possible and discards the remaining lower bits.

To work around this, do math on small numbers until the results get large. Then, transfer them to another location for additional large-number math. For example:

- x is the small incremented variable.
- y is the large incremented variable.
- z is the total current count that can be used anywhere.
- x = x+1;
- if x = 100,000;
- {
- y = y + 100,000;
- x = 0;
- }
- z = y + x;

Or another example:

- x = x + some_tiny_number;
- if (x >= 100)
- {
- z = z + 100;
- x = x - 100; // there might be a tiny remainder
- }

FBD Functions

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

.

FBD Functions are implemented in accordance with IEC 61131-3 Edition 3. Arithmetic and Numeric functions are provided in the Function Block Diagram language. Ladder Diagram and Structured Text languages include Arithmetic and Numeric as operators and functions.

FBD Functions have one or more inputs and one output. FBD Functions are implemented for efficiency, have smaller footprints and use less system resources to operate than FBD Function Blocks.

FBD Functions

- Require all inputs and outputs. All inputs must be of a supported data type.
- Do not have backing tags or predefined data types. Connected input values do not convert to predefined data types.
- Do not have EnableIn bits and are always executed.

Example: Add Function

Index through arrays

To dynamically change the array element that your logic references, use tag or expression as the subscript to point to the element. This is similar to indirect addressing in PLC-5 logic. You can use these operators in an expression to specify an array subscript:

Operator	Description
+	add
-	subtract/negate
*	multiply
/	divide
AND	AND
FRD	BCD to integer
NOT	complement
OR	OR
TOD	integer to BCD
SQR	square root
XOR	exclusive OR

For example:

Definitions	Example	Description
my_list defined as DINT[10]	my_list[5]	This example references element 5 in the array. The reference is static because the subscript value remains constant.
my_list defined as DINT[10] position defined as DINT	MOV the value 5 into position my_list[position]	This example references element 5 in the array. The reference is dynamic because

Definitions	Example	Description
		the logic can change the subscript by changing the value of position.
my_list defined as DINT[10] position defined as DINT offset defined as DINT	MOV the value 2 into position MOV the value 5 into offset my_list[position+offset]	This example references element 7 (2+5) in the array. The reference is dynamic because the logic can change the subscript by changing the value of position or offset.

Make sure any array subscript you enter is within the boundaries of the specified array. Instructions that view arrays as a collection of elements generate a major fault (type 4, code 20) if a subscript exceeds its corresponding dimension.

Bit Addressing

Bit addressing is used access a particular bit within a larger container. Larger containers include any integer, structure or BOOL array. For example:

Definition	Example	Description
Variable0 defined as LINT has 64 bits	variable0.42	This example references the bit 42 of variable0.
variable1 defined as DINT has 32 bits	variable1.2	This example references the bit 2 of variable1.
variable2 defined as INT has 16 bits	variable2.15	This example references the bit 15 of variable2.
variable3 defined as SINT holds 8 bits	variable3.[4]	This example references bit 4 of variable3.
variable4 defined as COUNTER structure has 5 status bits	variable4.DN	This example references the DN bit of variable4.
MyVariable defined as BOOL[100] MyIndex defined as SINT	MyVariable[(MyIndex AND NOT 7) / 8].[MyIndex AND 7]	This example references a bit within a BOOL array.
MyArray defined as BOOL[20]	MyArray[3]	This example references the bit 3 of MyArray.

Definition	Example	Description
variable5 defined as ULINT holds 64 bits	variable5.53	This example references the bit 53 of variable5.

Use Bit Addressing anywhere a BOOL typed tag is allowed.

Related information

Index through arrays on page 863

Major fault types and codes

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

Minor fault types and codes

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

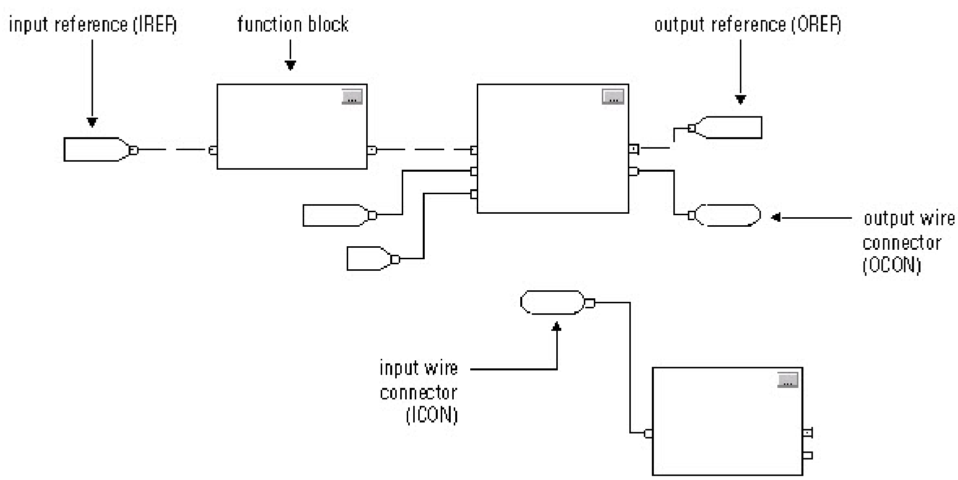
You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

Function Block Attributes

Click a topic below for more information on issues that are unique to function block programming. Review this information to make sure you understand how your function block routines will operate.

Choose the Function Block Elements

To control a device, use these elements:



Use the following table to help you choose your function block elements:

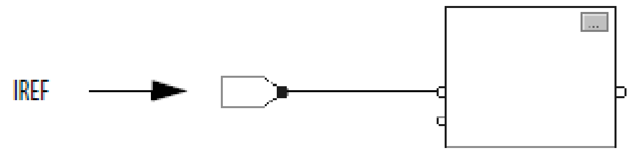
If you want to supply a value from an input device or tag	Then use an input reference (IREF)
Send a value to an output device or tag	Output reference (OREF)
Perform an operation on an input value or values and produce an output value or values.	Function block
Transfer data between function blocks when they are: • Far apart on the same sheet • On different sheets within the same routine	Output wire connector (OCON) and an input wire connector (ICON)
Disperse data to several points in the routine	Single output wire connector (OCON) and multiple input wire connectors (ICON)

The function block moves the input references into the block structure. If necessary, the function block converts those input references to REAL values. The function block executes and moves the results into the output references.

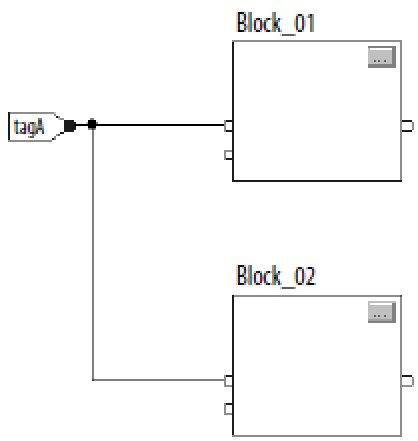
Again, if necessary, the function block converts those result values from REAL to the data types for the output references.

Latching Data

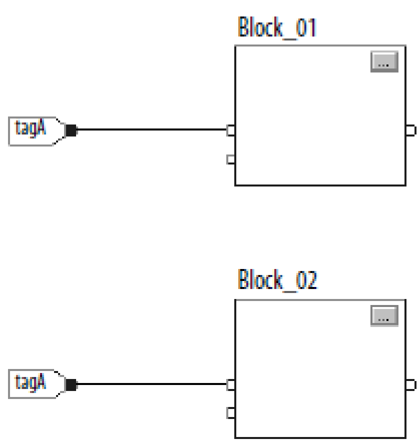
If you use an IREF to specify input data for a function block instruction, the data in that IREF is latched for the scan of the function block routine. The IREF latches data from program-scoped and controller-scoped tags. The controller updates all IREF data at the beginning of each scan.



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.

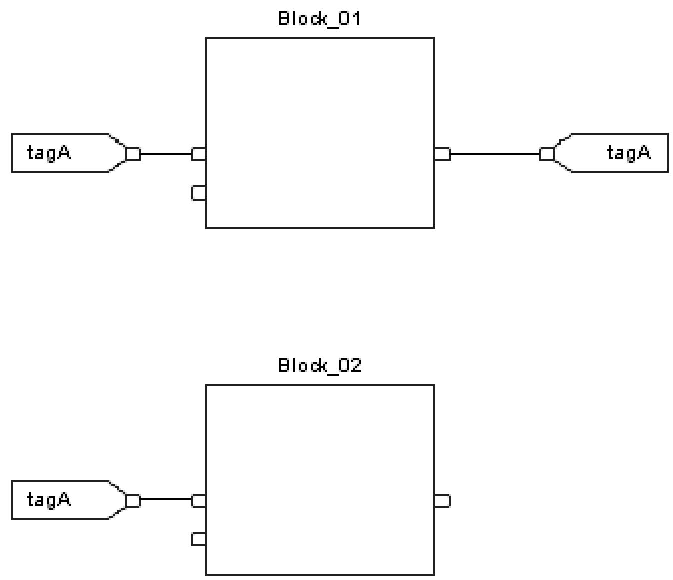


This example is the same as the one above. The value of tagA is stored only once at the beginning of the routine's execution. The routine uses this stored value throughout the routine.



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.



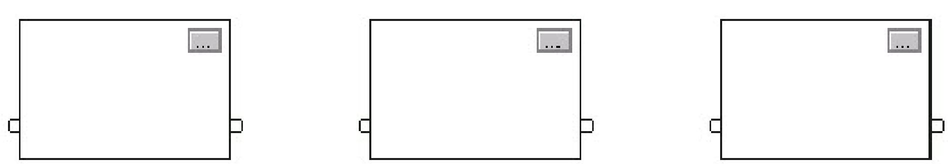
Order of Execution

The Logix Designer programming application automatically determines the order of execution for the function blocks in a routine when you:

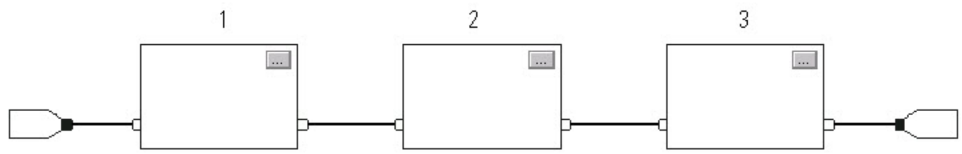
- verify a function block routine
- verify a project that contains a function block routine
- download a project that contains a function block routine

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

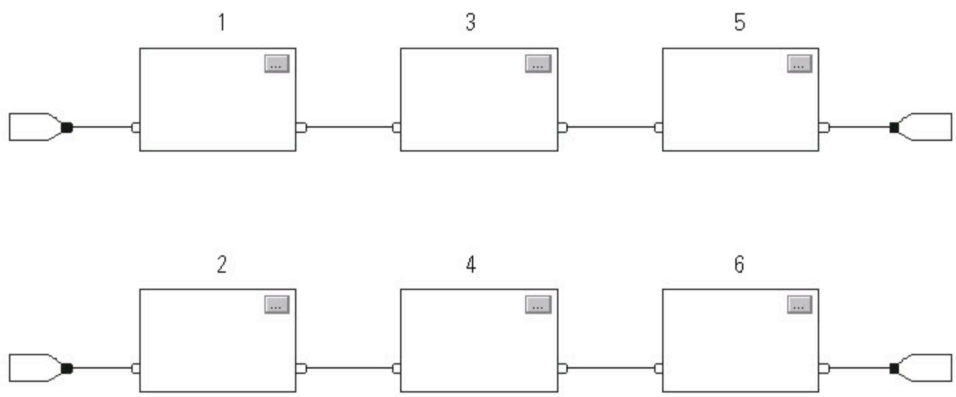
If function blocks are not wired together, it does not matter which block executes first. There is no data flow between the blocks



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.

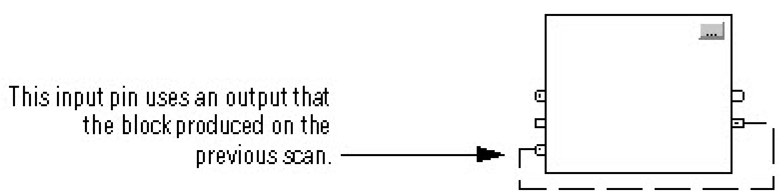


Execution order is only relative to the blocks that are wired together. The following example is fine because the two groups of blocks are not wired together. The blocks within a specific group execute in the appropriate order in relation to the blocks in that group.

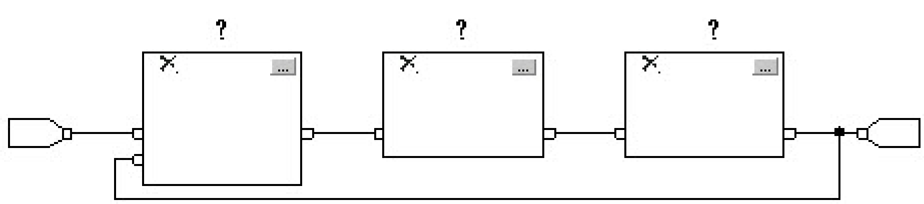


Resolve a Loop

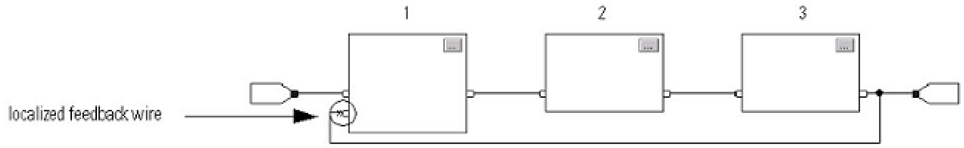
To create a feedback loop around a block, wire an output pin of the block to an input pin of the same block. The following example is OK. The loop contains only a single block, so execution order does not matter.



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.

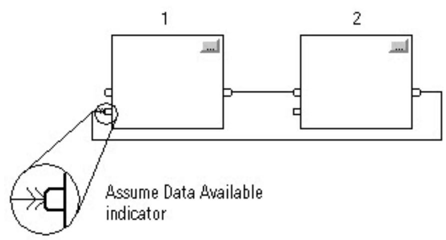


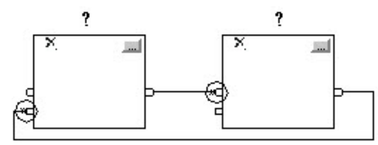
To identify which block to execute first, mark the input wire that creates the loop (the feedback wire) with the *Assume Data Available* indicator. In the following example, block 1 uses the output from block 3 that was produced in the previous execution of the routine.



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.

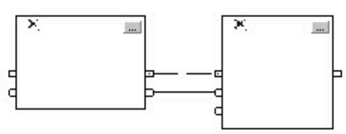
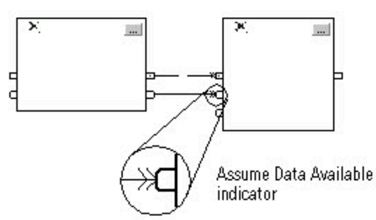
<table>
<tr><th>This is OK</th><th>This is NOT OK</th></tr>
<tr><td>

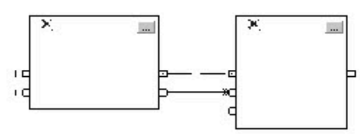
The *Assume Data Available* indicator defines the data flow within the loop.</td><td>

The controller cannot resolve the loop because all the wires use the *Assume Data Available* indicator.</td></tr>
</table>

Resolve Data Flow Between Two Blocks

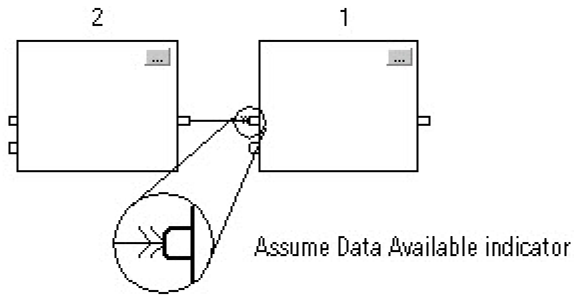
If you use two or more wires to connect two blocks, use the same data flow indicators for all of the wires between the two blocks.

<table>
<tr><th>This is OK</th><th>This is NOT OK</th></tr>
<tr><td>
Neither wire uses the *Assume Data Available* indicator.


Both wires use the *Assume Data Available* indicator.</td><td>
One wire uses the *Assume Data Available* indicator while the other wire does not.</td></tr>
</table>

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.

Function Block Responses to Overflow Conditions

In general, the function block instructions that maintain history do not update history with ± NAN, or ±INF values when an overflow occurs. Each instruction has one of these responses to an overflow condition.

Response	Instruction
Response 1 Blocks execute their algorithm and check the result for ±NAN or ±INF. If ±NAN or ±INF, the block outputs ±NAN or ±INF.	ALM NTCH DEDT PMUL DERV POSP ESEL RLIM FGEN RMPS HPF SCRV LDL2 SEL LDLG SNEG LPF SRTP MAVE SSUM MAXC TOT MINC UPDN MSTD MUX
Response 2 Blocks with output limiting execute their algorithm and check the result for ±NAN or ±INF. The output limits are defined by the HighLimit and LowLimit input parameters. If ±INF, the block outputs a limited result. If ±NAN, the output limits are not used and the block outputs ±NAN.	HLL, INTG, PI, PIDE, SCL, SOC

Response 3 The overflow condition does not apply. These instructions typically have a boolean output.	BAND, BNOT, BOR, BXOR, CUTD, D2SD, D3SD, DFF, JKFF, OSFI, OSRI, RESD, RTOR, SETD, TOFR, TONR

Timing Modes

These process control and drives instructions support different timing modes.

• DEDT	• LDLG	• RLIM
• DERV	• LPF	• SCRV
• HPF	• NTCH	• SOC
• INTG	• PI	• TOT
• LDL2	• PIDE	

There are three different timing modes.

<table>
<tr><th>Timing Mode</th><th colspan="2">Description</th></tr>
<tr><td rowspan="5">Periodic</td><td colspan="2">Periodic mode is the default mode and is suitable for most control applications.
We recommend that you place the instructions that use this mode in a routine that executes in a periodic task. The delta time (DeltaT) for the instruction is determined as follows:</td></tr>
<tr><td>If the instruction executes in a</td><td>Then DeltaT equals</td></tr>
<tr><td>Periodic task</td><td>Period of the task</td></tr>
<tr><td>Event or continuous task</td><td>Elapsed time since the previous execution
The controller truncates the elapsed time to whole milliseconds (ms). For example, if the elapsed time = 10.5 ms, the controller sets DeltaT = 10 ms.</td></tr>
<tr><td colspan="2">The update of the process input needs to be synchronized with the execution of the task or sampled 5-10 times faster than the task executes in order to minimize the sampling error between the input and the instruction.</td></tr>
<tr><td>Oversample</td><td colspan="2">In oversample mode, the delta time (DeltaT) used by the instruction is the value written into the OversampleDT parameter of the instruction. If the process input has a time stamp value, use the real time sampling mode instead.
Add logic to your program to control when the instruction executes. For example, you can use a timer set to the OversampleDeltaT value to control the execution by using the EnableIn input of the instruction.
The process input needs to be sampled 5-10 times faster than the instruction is executed in order to minimize the sampling error between the input and the instruction.</td></tr>
<tr><td>Real time sampling</td><td colspan="2">In the real time sampling mode, the delta time (DeltaT) used by the instruction is the difference between two time stamp values that correspond to the updates of the</td></tr>
</table>

	process input. Use this mode when the process input has a time stamp associated with its updates and you need precise coordination. The time stamp value is read from the tag name entered for the RTSTimeStamp parameter of the instruction. Normally this tag name is a parameter on the input module associated with the process input. The instruction compares the configured RTSTime value (expected update period) against the calculated DeltaT to determine if every update of the process input is being read by the instruction. If DeltaT is not within 1 millisecond of the configuration time, the instruction sets the RTSMissed status bit to indicate that a problem exists reading updates for the input on the module.

Time-based instructions require a constant value for DeltaT in order for the control algorithm to properly calculate the process output. If DeltaT varies, a discontinuity occurs in the process output. The severity of the discontinuity depends on the instruction and range over which DeltaT varies.

A discontinuity occurs if the following happens:

- Instruction is not executed during a scan.
- Instruction is executed multiple times during a task.
- Task is running and the task scan rate or the sample time of the process input changes.
- User changes the time-base mode while the task is running.
- Order parameter is changed on a filter block while the task is running.
- Changing the Order parameter selects a different control algorithm within the instruction.

Common Instruction Parameters for Timing Modes

The instructions that support time-base modes have these input and output parameters.

Input Parameters

Input Parameter	Data Type	Description
TimingMode	DINT	Selects timing execution mode. Value: Description: 0 Periodic mode 1 Oversample mode 2 Real time sampling mode Valid = 0 to 2 Default = 0 When TimingMode = 0 and task is periodic, periodic timing is enabled and DeltaT is set to the task scan rate. When TimingMode = 0 and task is event or continuous, periodic timing is enabled and DeltaT is set equal to the elapsed time span since the last time the instruction was executed. When TimingMode = 1, oversample timing is enabled and DeltaT is set to the value

		of the OversampleDT parameter. When TimingMode = 2, real time sampling timing is enabled and DeltaT is the difference between the current and previous time stamp values read from the module associated with the input. If TimingMode invalid, the instruction sets the appropriate bit in Status.
OversampleDT	REAL	Execution time for oversample timing. The value used for DeltaT is in seconds. If TimingMode = 1, then OversampleDT = 0.0 disables the execution of the control algorithm. If invalid, the instruction sets DeltaT = 0.0 and sets the appropriate bit in Status. Valid = 0 to 4194.303 seconds Default = 0.0
RTSTime	DINT	Module update period for real time sampling timing. The expected DeltaT update period is in milliseconds. The update period is normally the value that was used to configure the module's update time. If invalid, the instruction sets the appropriate bit in Status and disables RTSMissed checking. Valid = 1…32,767ms Default = 1
RTSTimeStamp	DINT	Module time stamp value for real time sampling timing. The time stamp value that corresponds to the last update of the input signal. This value is used to calculate DeltaT. If invalid, the instruction sets the appropriate bit in Status, disables execution of the control algorithm, and disables RTSMissed checking. Valid =0…32,767ms (wraps from 32767 to 0) 1 count = 1 millisecond Default = 0

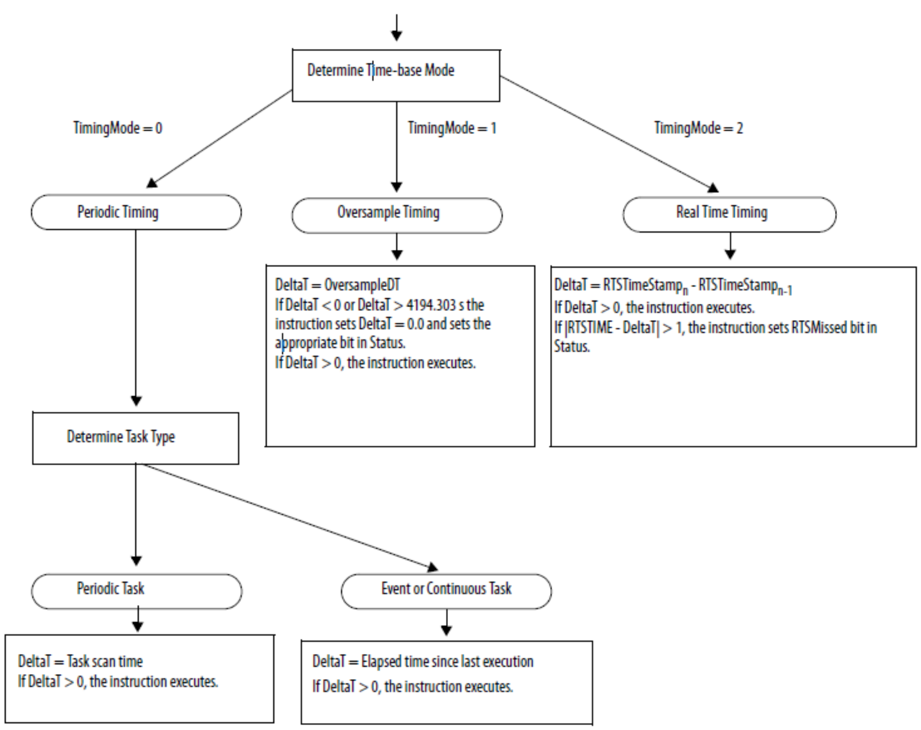
Output Parameters

Output Parameter	Data Type	Description

DeltaT	REAL	Elapsed time between updates. This is the elapsed time in seconds used by the control algorithm to calculate the process output. Periodic: DeltaT = task scan rate if task is Periodic task, DeltaT = elapsed time since previous instruction execution if task is Event or Continuous task Oversample: DeltaT = OversampleDT Real Time Sampling: DeltaT = (RTSTimeStampn - RTSTimeStampn-1)
Status	DINT	Status of the function block.
TimingModeInv (Status.27)	BOOL	Invalid TimingMode value.
RTSMissed (Status.28)	BOOL	Only used in real time sampling mode. Set when ABS | DeltaT - RTSTime | > 1 (.001 second).
RTSTimeInv (Status.29)	BOOL	Invalid RTSTime value.
RTSTimeStampInv (Status.30)	BOOL	Invalid RTSTimeStamp value.
DeltaTInv (Status.31)	BOOL	Invalid DeltaT value.

Overview of Timing Modes

The following diagram shows how an instruction determines the appropriate timing mode.



Program/Operator Control

The following instructions support the concept of Program/Operator control.

- 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.

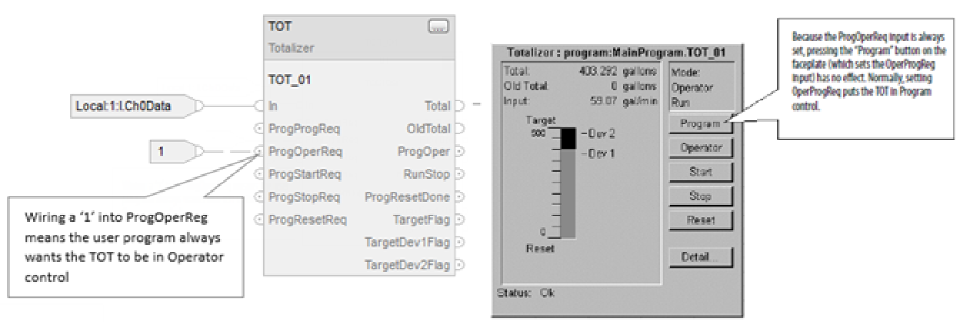
Input	Description
.ProgProgReq	A program request to go to Program control.
.ProgOperReq	A program request to go to Operator control.
.OperProgReq	An operator request to go to Program control.
.OperOperReq	An operator request to go to Operator control.

To determine whether an instruction is in Program or Operator 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.

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.

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.

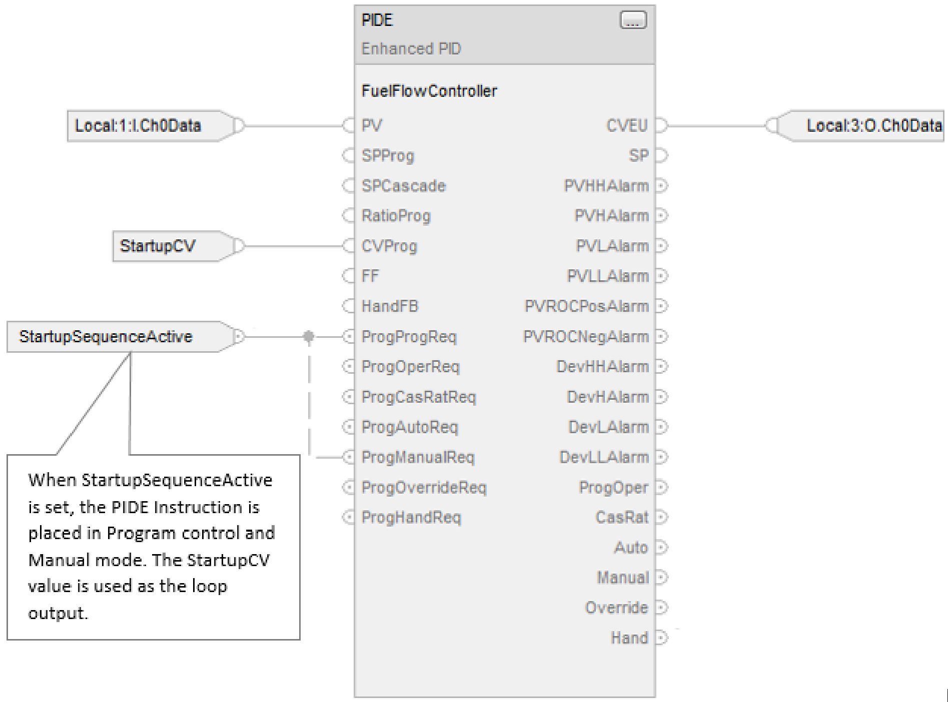
For example, let's assume that a Totalizer instruction will always be used in Operator control, and your user program will never control the running or stopping of the Totalizer. In this case, you could wire a literal value of 1 into the ProgOperReq. This would prevent the operator from ever putting the Totalizer into Program control by setting the OperProgReq from an operator interface device.



Likewise, 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.

The following example shows how to lock an instruction into Program control.

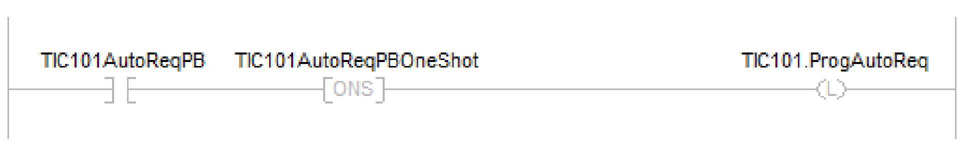


Operator request inputs to an instruction are always cleared by the instruction when it executes. This allows operator interfaces to work with these instructions by merely setting the desired mode request bit. You don't have to program the operator interface to reset the request bits. For example, if an operator interface sets the OperAutoReq input to a PIDE instruction, when the PIDE instruction executes, it determines what the appropriate response should be and clears the OperAutoReq.

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.

In this example, a rung of ladder logic in another routine is used to one-shot latch a ProgAutoReq to a PIDE instruction when a push button is pushed.

When the TIC101AutoReq push button is pressed, one-shot latch ProgAutoReq for the PIDE instruction TIC101. TIC101 has been configured with the ProgValueReset input set. ProgAutoReq get reset because ProgValuieReset is always set.



Structured Text Programming

These are the issues that are unique with structured text programming. Review the following topics to make sure you understand how your structured text programming executes.

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.

Structured text is not case sensitive. Structured text can contain these components.

Term	Definition	Examples
Assignment	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	An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression), a String (string expression), or to a true or false state (BOOL expression)	

Tag Expression	A named area of the memory where data is stored (BOOL, SINT, INT, DINT, REAL, String).	value1
Immediate Expression	A constant value	4
Operators Expression	A symbol or mnemonic that specifies an operation within an expression.	tag1 + tag2 tag1 >= value1
Function Expression	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 be used only in expressions. Instructions cannot be used in expressions.	function(tag1)

Instruction	An instruction is a standalone statement.
	An instruction uses parentheses 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(;).
	Even though their syntax is similar, instructions differ from functions in that instructions cannot be used in expressions. Functions can be used only in expressions.
		instruction();
		instruction(operand);
		instruction(operand1, operand2,operand3);

Construct	A conditional statement used to trigger structured text code (that is, other statements). Terminate the construct with a semi colon (;).	IF...THEN CASE FOR...DO WHILE...DO REPEAT...UNTIL EXIT
Comment	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*/

Structured Text Components: Comments

To make your structured text easier to interpret, add comments to it.

- Comments let you use plain language to describe how your structured text works.
- Comments do not affect the execution of the structured text.

To add comments to your structured text:

To add a comment	Use one of these formats
on a single line at the end of a line of structured text	//comment (*comment*) /*comment*/
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*/

For example:

Format	Example

//comment	**At the beginning of a line**
	//Check conveyor belt direction
	IF conveyor_direction THEN...
	At the end of a line
	ELSE //If conveyor isn't moving, set alarm light
	light := 1;
	END_IF;

(*comment*)	Sugar.Inlet[:=]1;(*open the inlet*) IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN... (*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*) IF tank.temp > 200 THEN...
/*comment*/	Sugar.Inlet:=0;/*close the inlet*/ IF bar_code=65 /*A*/ THEN... /*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/ SIZE(Inventory,0,Inventory_Items);

Structured Text Components: Assignments

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

tag := expression;

where:

<table>
<tr><td>Component</td><td colspan="2">Description</td></tr>
<tr><td>Tag</td><td colspan="2">Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL.
Tip: The STRING tag is applicable to

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

only.</td></tr>
<tr><td>:=</td><td colspan="2">Is the assignment symbol</td></tr>
<tr><td rowspan="5">Expression</td><td colspan="2">Represents the new value to assign to the tag</td></tr>
<tr><td>If tag is this data type</td><td>Use this type of expression</td></tr>
<tr><td>BOOL</td><td>BOOL</td></tr>
<tr><td>SINT

INT

DINT

REAL</td><td>Numeric</td></tr>
<tr><td>STRING

(

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

only).</td><td>String type, including string tag and string literal

(

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

only).</td></tr>
<tr><td>;</td><td colspan="2">Ends the assignment</td></tr>
</table>

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

The expression can be simple, such as an immediate value or another tag name, or the expression can be complex and include several operators and functions, or both. Refer to Expressions for more information.

Tip: I/O module data updates asynchronously to the execution of logic. If you reference an input multiple times in your logic, the input could change state between separate references. If you need the input to have the same state for each reference, buffer the input value and reference that buffer tag. For more information, see Logix 5000 Controllers Common Procedures, publication 1756-PM001.

You can also use Input and Output program parameters which automatically buffer the data during the Logix Designer application execution. See LOGIX 5000 Controllers Program Parameters Programming Manual, publication 1756-PM021.

Specify a non-retentive assignment

The non-retentive assignment is different from the regular assignment described above in that the tag in 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. This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine by using a JSR instruction.

A non-retentive assignment has this syntax:

tag [:=] expression ;

where:

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>tag</td><td colspan="2">Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL.
Tip: The STRING tag is applicable to

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

only.</td></tr>
<tr><td>[:=]</td><td colspan="2">Is the non-retentive assignment symbol.</td></tr>
<tr><td rowspan="8">expression</td><td colspan="2">Represents the new value to assign to the tag.</td></tr>
<tr><td>If tag is this data type</td><td>Use this type of expression</td></tr>
<tr><td>BOOL</td><td>BOOL</td></tr>
<tr><td>SINT</td><td rowspan="4">Numeric</td></tr>
<tr><td>INT</td></tr>
<tr><td>DINT</td></tr>
<tr><td>REAL</td></tr>
<tr><td>STRING

(

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

only).</td><td>String type, including string tag and string literal

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers

(only)</td></tr>
</table>

Assign an ASCII character to a string data member

Use the assignment operator to assign an ASCII character to an element of the DATA member of a string tag. To assign a character, specify the value of the character or specify the tag name, DATA member, and element of the character. For example:

This is OK	This is not OK

string1.DATA[0] := 65;	string1.DATA[0] := A;
string1.DATA[0]:= string2.DATA[0];	string1 := string2; **Tip:** This assigns all content of string2 to string1 instead of just one character.

To add or insert a string of characters to a string tag, use either of these ASCII string instructions:

To	Use this instruction
Add characters to the end of a string	CONCAT
Insert characters into a string	INSERT

Structured Text Components: 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)
- String literal that you enter directly into the expression (
  CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers
  only)
- Functions, such as: ABS, TRUNC
- Operators, such as: +, -, <, >, And, Or

Follow these guidelines for writing expressions:

- Use any combination of upper-case and lower-case letter. For example, these variations of "AND" are acceptable: AND, And, and.
- For more complex requirements, use parentheses to group expressions within expressions. This makes the whole expression easier to read, and ensures that the expression executes in the desired sequence.

Use these expressions for structured text:

BOOL expression: An expression that produces 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, 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.
- Nest a numeric expression within a BOOL expression. For example, (tag1+5)>65.

String expression: An expression that represents a string

- A simple expression can be a string literal or a string tag

Use this table to select the operators for expressions.

If	Use
Calculating an arithmetic value	Arithmetic operators and functions
Comparing two values or strings	Relational operators
Verifying if conditions are true or false	Logical operators
Comparing the bits within values	Bitwise operators

Use arithmetic operators and functions

Combine multiple operators and functions in arithmetic expressions.

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

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

The table provides examples for using arithmetic operators and functions.

Use this format	Example	
	For this situation	Write
value1 operator value2	If gain_4 and gain_4_adj are DINT tags and your specification says:	gain_4_adj := gain_4+15;

	'Add 15 to gain_4 and store the result in gain_4_adj'"	
operator value1	If alarm and high_alarm are DINT tags and your specification says: 'Negate high_alarm and store the result in alarm.'	alarm:= -high_alarm;
function(numeric_expression)	If overtravel and overtravel_POS are DINT tags and your specification says: 'Calculate the absolute value of overtravel and store the result in overtravel_POS.'	overtravel_POS := ABS(overtravel);
value1 operator (function((value2+value3)/2)	If adjustment and position are DINT tags and sensor1 and sensor2 are REAL tags and your specification says: 'Find the absolute value of the average of sensor1 and sensor2, add the adjustment, and store the result in position.'	position := adjustment + ABS((sensor1 + sensor2)/2);

Use bitwise operators

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

The following provides an overview of the bitwise operators.

For	Use this operator	Optimal data type
bitwise AND	&, AND	DINT
bitwise OR	OR	DINT
bitwise exclusive OR	XOR	DINT
bitwise complement	NOT	DINT

This is an example.

Use this format	Example	
	For this situation	Use
value1 operator value2	If input1, input2, and result1 are DINT tags and your specification says: "Calculate the bitwise result of input1 and input2. Store the result in result1."	result1 := input1 AND input2;

Use logical operators

Use logical operators to verify 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

Use these logical operators.

For this comparison	Use this operator	Optimal data type
logical AND	&, AND	BOOL
logical OR	OR	BOOL
logical exclusive OR	XOR	BOOL
logical complement	NOT	BOOL

The table provides examples of using logical operators.

Use this format	Example	
	For this situation	Use
BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye_1 is on then..."	IF photoeye THEN...
NOT BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye is off then..."	IF NOT photoeye THEN...
expression1 & expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on and temp is less than 100 then..."	IF photoeye & (temp<100) THEN...
expression1 OR expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on or temp is less than 100 then...".	IF photoeye OR (temp<100) THEN...
expression1 XOR expression2	If photoeye1 and photoeye2 are BOOL tags and your specification says: "If: photoeye1 is on while photoeye2 is off or photoeye1 is off while photoeye2 is on then..."	IF photoeye1 XOR photoeye2 THEN...
BOOLtag := expression1 & expression2	If photoeye1 and photoeye2 are BOOL tags, open is a BOOL tag, and your specification says: "If photoeye1 and photoeye2 are both on, set open to true"	open := photoeye1 & photoeye2;

Use 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

Use these relational operators.

For this comparison	Use this operator	Optimal data type
Equal	=	DINT, REAL, String type
Less than	<	DINT, REAL, String type
Less than or equal	<=	DINT, REAL, String type
Greater than	>	DINT, REAL, String type
Greater than or equal	>=	DINT, REAL, String type
Not equal	<>	DINT, REAL, String type

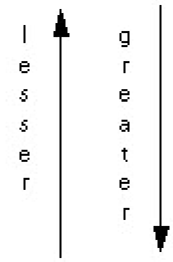
The table provides examples of using relational operators

Use this format	Example	
	For this situation	Write
value1 operator value2	If temp is a DINT tag and your specification says: 'If temp is less than 100· then…'	IF temp<100 THEN…
stringtag1 operator stringtag2	If bar_code and dest are string tags and your specification says: 'If bar_code equals dest then…'	IF bar_code=dest THEN…
stringtag1 operator 'character string literal'	If bar_code is a string tag and your specification says: 'If bar_code equals 'Test PASSED' then…'	IF bar_code='Test PASSED' THEN…
char1 operator char2 To enter an ASCII character directly into the expression, enter the decimal value of the character.	If bar_code is a string tag and your specification says: 'If bar_code.DATA[0] equals 'A' then…'	IF bar_code.DATA[0]=65 THEN…
bool_tag := bool_expressions	If count and length are DINT tags, done is a BOOL tag, and your specification says: 'If count is greater than or equal to length, you are done counting.'	Done := (count >= length);

How strings are evaluated

The hexadecimal values of the ASCII characters determine if one string is less than or greater than another string.

- When the two strings are sorted as in a telephone directory, the order of the strings determines which one is greater.



ASCII Characters	Hex Codes	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
ab	$61$62	

- Strings are equal if their characters match.
- Characters are case sensitive. Upper case "A" ($41) is not equal to lower case "a" ($61).

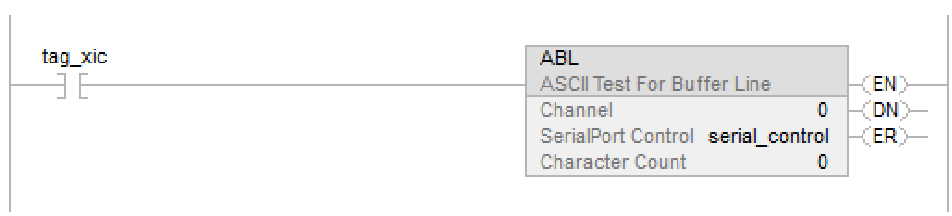
Structured Text Components: Instructions

Structured text statements can also be instructions. 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 ladder diagram instructions that use rung-condition-in to trigger execution. Some ladder diagram instructions only execute when rung- condition-in toggles from false to true. These are transitional ladder diagram instructions. In structured text, instructions execute when they are scanned unless pre-conditioning the execution of the structured text instruction.

For example, the ABL instruction is a transitional instruction in ladder diagram. 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 clears.



In structured text, if writting 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;
```

Structured Text Components: Constructs

Program constructs alone or nest within other constructs.

If	Use this construct
Doing something if or when specific conditions occur	IF. . . THEN
Selecting what to do based on a numerical value	CASE. . . OF
Doing something a specific number of times before doing anything else	FOR. . . DO
Continuing doing something when certain conditions are true	WHILE. . . DO
Continuing doing something until a condition is true	REPEAT. . . UNTIL

Some Key Words are Reserved

These constructs are not available:

- GOTO
- REPEAT

Logix Designer application will not let you use them as tag names or constructs.

Character string literals

Character string literals include single byte or double byte encoded characters. A single-byte string literal is a sequence of zero or more characters that are prefixed and terminated by the single quote character ('). In single byte character strings, the three-character combination of the dollar sign ($) followed by two hexadecimal digits is interpreted as the hexadecimal representation of the eight-bit character code as shown in the following table.

Tip: Character string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers. Studio 5000 only supports single byte characters.

Character string literals

No.	Description	Example
1a	Empty string (length zero)	''

1b	String of length one or character CHAR containing a single character	'A'
1c	String of length one or character CHAR containing the “space” character	' '
1d	String of length one or character CHAR containing the “single quote” character	'$''
1e	String of length one or character CHAR containing the “double quote” character	'"'
1f	Support of two character combinations	'RL'
1g	Support of a character representation with ‘$’ and two hexadecimal characters	'$0A'

Two-character combinations in character strings

No.	Description	Example
1	Dollar sign	$$
2	Single quote	$'
3	Line feed	$L or $l
4	Newline	$N or $n
5	Form feed (page)	$P or $p
6	Carriage return	$R or $r
7	Tabulator	$T or $t

Tip: The newline character provides an implementation-independent means of defining the end of a line of data for both physical and file I/O; for printing, the effect is that of ending a line of data and resuming printing at the beginning of the next line.

The $' combination is only valid inside single quoted string literals.

Integer literal suffixes

This table lists suffixes you can add to integer literals in Structured Text, and the corresponding range for each suffix.

Suffix	**Literal data type**	**Range**
None	DINT	-2,147,483,648 to 2,147,483,648
L	LINT	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,808
U	UDINT	0 to 4,294,967,295
UL	ULINT	0 to 18,446,744,073,709,551,615

String Types

Store ASCII characters in tags that use a string type data type to:

- Use the default STRING data type, which stores up to 82 characters
- Create a new string type that stores less or more characters

To create a new string type, refer to the Logix 5000 Controllers ASCII Strings Programming Manual publication 1756-PM013.

Each string type contains the following 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 using: • The String Browser to enter characters • 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. For example, to access the characters of the string_1 tag, enter string_1. Each element of the DATA array contains one character. Create new string types that store less or more characters.

CASE_OF

Use CASE_OF to select what to do based on a numerical value.

Operands

CASE numeric_expression OF

selector1: statement;

selectorN: statement; ELSE

Structured Text

Operand	Type	Format	Enter
Numeric_ expression	SINT INT DINT REAL	Tag expression	Tag or expression that evaluates to a number (numeric expression)
Selector	SINT INT DINT REAL	Immediate	Same type as numeric_expression

IMPORTANT: If using REAL values, use a range of values for a selector because a REAL value is more likely to be within a range of values than an exact match of one, specific value.

Description

The syntax is described in the table.

Specify as many alternative selector values (paths) as you need.

```
CASE numeric_expression OF
    selector1: <statement>;   (1)
               .
               .
               .
    selector2: <statement>;   (2)
               .
               .
               .
    selector3: <statement>;   (3)
               .
               .
               .
ELSE
               <statement>;   (4)
               .
               .
               .
END_CASE;
```

optional

These are the syntax for entering the selector values.

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>

The CASE construct is similar to a switch statement in the C or C++ programming languages. With the CASE construct, the controller executes only the statements that associated with the first matching selector value. Execution always breaks after the statements of that selector and goes to the END_CASE statement.

Affects Math Status Flags

No

Major/Minor Faults

None

Example

If you want this	Enter this structured text
If recipe number = 1 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	CASE recipe_number OF
	1: Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
If recipe number = 2 or 3 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	2,3: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 4, 5, 6, or 7 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	4..7: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 8, 11, 12, or 13 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	8,11..13 Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
Otherwise all outlets = closed (0)	ELSE
	Ingredient_A.Outlet_1 [:=]0; Ingredient_A.Outlet_4 [:=]0; Ingredient_B.Outlet_2 [:=]0; Ingredient_B.Outlet_4 [:=]0;
	END_CASE;

The [:=] tells the controller to also clear the outlet tags whenever the controller does the following:

Enters the RUN mode.

Leaves the step of an SFC if configuring the SFC for Automatic reset. This applies only embedding the assignment in the action of the step or using the action to call a structured text routine via a JSR instruction.

FOR_DO

Use the FOR_DO loop to perform an action a number of times before doing anything else.

When enabled, the FOR instruction repeatedly executes the Routine until the Index value exceeds the Terminal value. The step value can be positive or negative. If it is negative, the loop ends when the index is less than the terminal value.. If it is positive, the loop ends when the index is greater than the terminal value.

Each time the FOR instruction executes the routine, it adds the Step size to the Index.

Do not loop too many times in a single scan. An excessive number of repetitions causes the controller watchdog to timeout and causes a major fault.

Operands

FOR count:= initial_value TO

final_value BY increment DO

<statement>;

END_FOR;

Operand	Type	Format	Description
count	SINT INT DINT	Tag	Tag to store count position as the FOR_DO executes
initial_ value	SINT INT DINT	Tag expression Immediate	Must evaluate to a number Specifies initial value for count
final_ value	SINT INT DINT	Tag expression Immediate	Specifies final value for count, which determines when to exit the loop
increment	SINT INT DINT	Tag expression Immediate	(Optional) amount to increment count each time through the loop If you don't specify an increment, the count increments by 1.

IMPORTANT: Do not iterate within the loop too many times in a single scan.

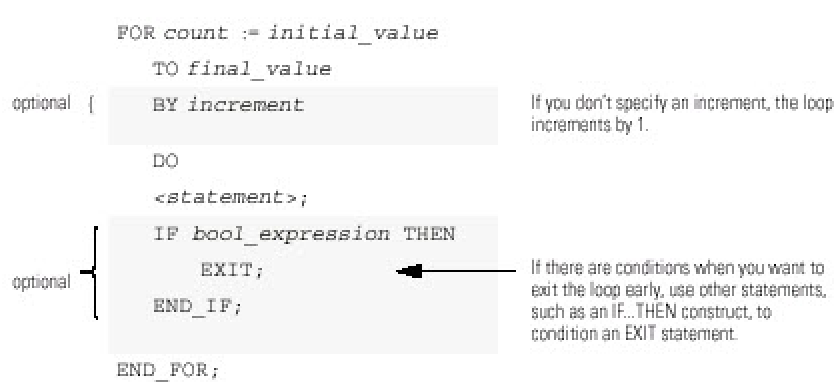
The controller does not execute other statements in the routine until it completes the loop.

A major fault occurs when completing the loop takes longer than the watchdog timer for the task.

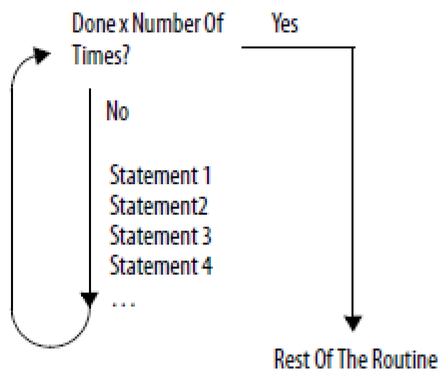
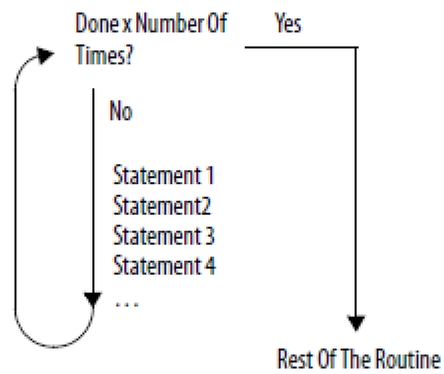
Consider using a different construct, such as IF_THEN.

Description

The syntax is described in the table.



This diagrams illustrates 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.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if	Fault type	Fault code
The construct loops too long.	6	1

Example 1

If performing the following,	Enter this structured text

Clear bits 0...31 in an array of BOOLs:
Initialize the subscript tag to 0.
Clear i . For example, when subscript = 5, clear array[5].
Add 1 to subscript.
If subscript is ≤ to 31, repeat 2 and 3.
Otherwise, stop.
	For subscript:=0 to 31 by 1 do
	array[subscript] := 0;
	End_for;

Example 2

If performing the following,	Enter this structured text

A user-defined data type (structure) stores the following information about an item in your inventory:
• Barcode ID of the item (String data type)
• Quantity in stock of the item (DINT data type)
An array of the above structure contains an element for each different item in your inventory. You want to search the array for a specific product (use its bar code) and determine the quantity that is in stock.
1. Get the size (number of items) of the Inventory array and store the result in
2. Inventory_Items (DINT tag).
	SIZE(Inventory,0,Inventory_Items);
	For position:=0 to Inventory_Items - 1 do
	If Barcode = Inventory[position].ID then
	Quantity := Inventory[position].Qty;
	Exit;
	End_if;
	End_for;

Initialize the position tag to 0. 1. If Barcode matches the ID of an item in the array, then: Set the Quantity tag = Inventory[position].Qty. This produces the quantity in stock of the item. Stop. Barcode is a string tag that stores the bar code of the item for which you are searching. For example, when position = 5, compare Barcode to Inventory[5].ID. 1. Add 1 to position. 2. If position is ≤ to (Inventory_Items -1), repeat 3 and 4. Since element numbers start at 0, the last element is 1 less than the number of elements in the array. Otherwise, stop.	

IF_THEN

Use IF_THEN to complete an action when specific conditions occur.

Operands

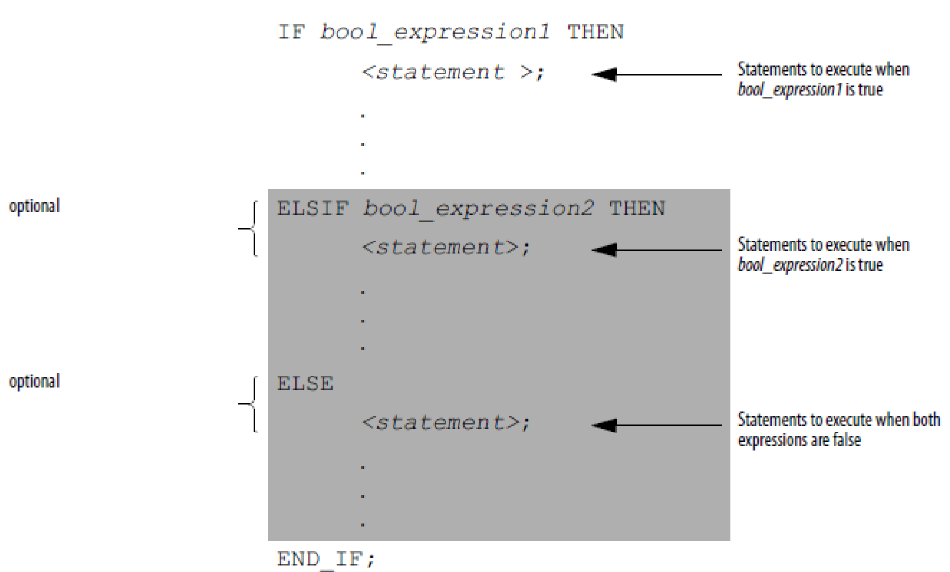
IF bool_expression THEN

<statement>;

Operand	Type	Format	Enter
Bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

Description

The syntax is described in the table.



To use ELSIF or ELSE, follow these guidelines.

To select from several possible groups of statements, add one or more ELSIF statements.

Each ELSIF represents an alternative path.

Specify as many ELSIF paths as you need.

The controller executes the first true IF or ELSIF and skips the rest of the ELSIFs and the ELSE.

To do something when all of the IF or ELSIF conditions are false, add an ELSE statement.

The table summarizes different combinations of IF, THEN, ELSIF, and ELSE.

If	And	Use this construct
Doing something if or when conditions are true	Do nothing if conditions are false	IF_THEN
	Do something else if conditions are false	IF_THEN_ELSE
Selecting alternative statements or groups of statements based on input conditions	Do nothing if conditions are false	IF_THEN_ELSIF
	Assign default statements if all conditions are false	IF_THEN_ELSIF_ELSE

Affects Math Status Flags

No

Major/Minor Faults

None.

Examples

Example 1

IF...THEN

If performing this	Enter this structured text
IF rejects > 3 then	IF rejects > 3 THEN
conveyor = off (0)	conveyor := 0;
alarm = on (1)	alarm := 1;
	END_IF;

Example 2

IF_THEN_ELSE

If performiing this	Enter this structured text
If conveyor direction contact = forward (1) then	IF conveyor_direction THEN
light = off	light := 0;
Otherwise light = on	ELSE

	light [:=] 1;
	END_IF;

The [:=] tells the controller to clear light whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 3

IF...THEN...ELSIF

If performing this	Enter this structured text
If sugar low limit switch = low (on) and sugar high limit switch = not high (on) then	IF Sugar.Low & Sugar.High THEN
inlet valve = open (on)	Sugar.Inlet [:=] 1;
Until sugar high limit switch = high (off)	ELSIF NOT(Sugar.High) THEN
	Sugar.Inlet := 0;
	END_IF;

The [:=] tells the controller to clear Sugar.Inlet whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 4

IF...THEN...ELSIF...ELSE

If performing this	Enter this structured text
If tank temperature > 100	IF tank.temp > 200 THEN
then pump = slow	pump.fast :=1; pump.slow :=0; pump.off :=0;
If tank temperature > 200	ELSIF tank.temp > 100 THEN
then pump = fast	pump.fast :=0; pump.slow :=1; pump.off :=0;
Otherwise pump = off	ELSE
	pump.fast :=0; pump.slow :=0; pump.off :=1;
	END_IF;

REPEAT_UNTIL

Use the REPEAT_UNTIL loop to continue performing an action until conditions are true.

Operands

REPEAT

<statement>;

Structured Text

Operand	Type	Format	Enter
bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

IMPORTANT: Do not iterate within the loop too many times in a single scan.

The controller does not execute other statements in the routine until it completes the loop.

A major fault occurs when completing the loop takes longer than the watchdog timer for the task.

Consider using a different construct, such as IF_THEN.

Description

The syntax is:

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

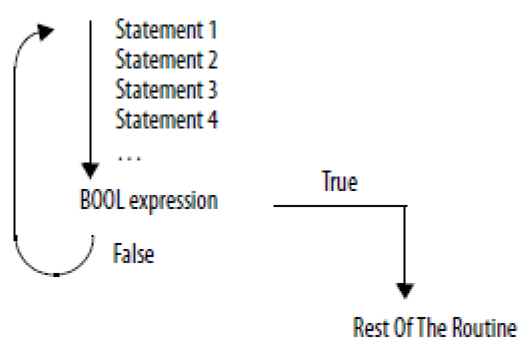
statements to execute while *bool_expression1* is false

optional

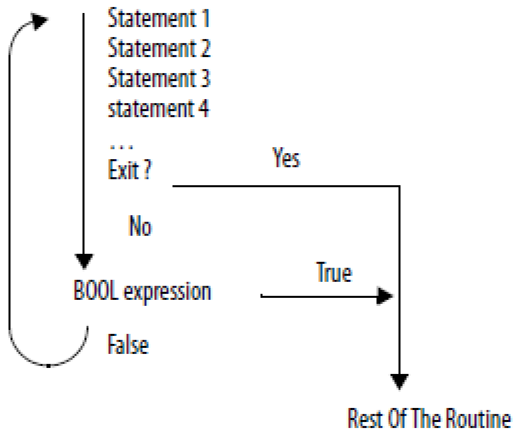
If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.

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.



Affects Math Status Flags

No

Fault Conditions

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

Example 1

If performing the following,	Enter this structured text

The REPEAT_UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. This differs from the WHILE_DO loop because the WHILE_DO The WHILE_DO loop evaluates its conditions first.
If the conditions are true, the controller then executes the statements within the loop. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.
	pos := -1;
	REPEAT
	pos := pos + 2;
	UNTIL ((pos = 101) OR (structarray[pos].value = targetvalue))
	end_repeat;

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
Set String_tag[element_number] = the character at SINT_array[element_number].
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	Repeat
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then

Add 1 to element_number. This lets the controller check the next character in	exit;
	end_if;
SINT_array.	Until SINT_array[element_number] = 13
Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.) If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.) If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.	end_repeat;

WHILE_DO

Use the WHILE_DO loop to continue performing an action while certain conditions are true.

Operands

WHILE bool_expression DO

<statement>;

Structured Text

Operand	Type	Format	Description
bool_expression	BOOL	tag expression	BOOL tag or expression that evaluates to a BOOL value

IMPORTANT: Do not iterate within the loop too many times in a single scan.

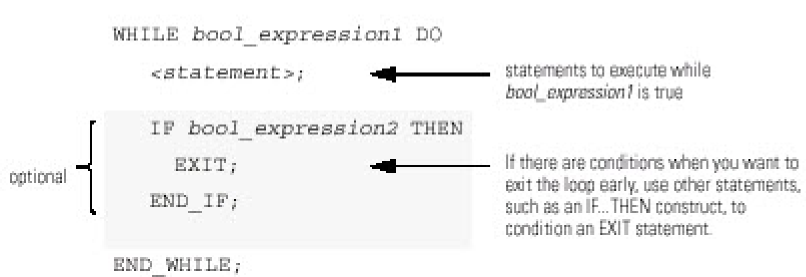
The controller does not execute any other statements in the routine until it completes the loop.

A major fault occurs when completing the loop takes longer than the watchdog timer for the task.

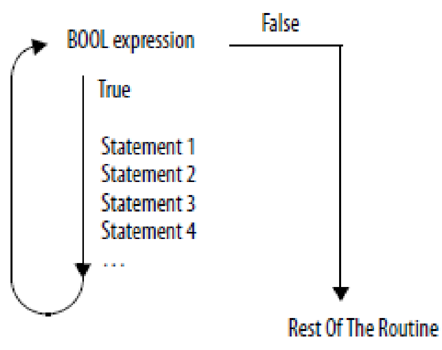
Consider using a different construct, such as IF_THEN.

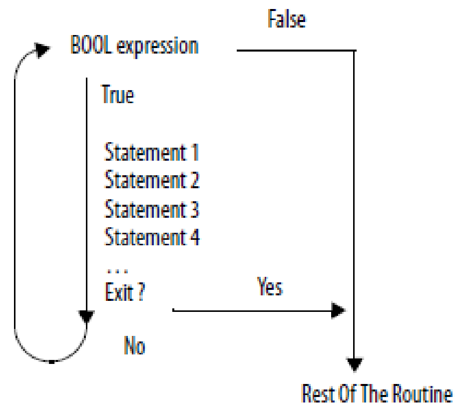
Description

The syntax is:



The following diagrams illustrate 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.

Affects Math Status Flags

No

Fault Conditions

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

Example 1

If performing the following,	Enter this structured text	
The WHILE_DO loop evaluates its conditions first. If the conditions are true, the controller then executes the statements within the loop. This differs from the REPEAT_UNTIL loop because the REPEAT_UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.	pos := 0;	
	While ((pos <= 100) & structarray[pos].value <> targetvalue)) do	
		pos := pos + 2;
		String_tag.DATA[pos] := SINT_array[pos];
	end_while;	

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
Set String_tag[element_number] = the character at SINT_array[element_number].
Add 1 to element_number. This lets the controller check the next character in SINT_array.
Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.)
If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	While SINT_array[element_number] <> 13 do
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;
	end_while;

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.

rockwellautomation.com — expanding **human possibility**™

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-RM003Z-EN-P - September 2024
Supersedes Publication 1756-RM003Y-EN-P - November 2023