

Concept  
Block Library LL984  
Volume 2

840 USE 506 00 eng Version 2.6



© 2002 Schneider Electric All Rights Reserved

---

---

## Table of Contents



---

**About the book . . . . . XI**

**The chapters marked gray are not included in this volume.**

<b>Part I</b>	<b>General Information . . . . .</b>	<b>1</b>
	Introduction . . . . .	1
<b>Chapter 1</b>	<b>Instructions . . . . .</b>	<b>3</b>
	Parameter Assignment of Instructions . . . . .	3
<b>Chapter 2</b>	<b>Instruction Groups . . . . .</b>	<b>5</b>
	At a Glance . . . . .	5
	Instruction Groups . . . . .	6
	ASCII Functions . . . . .	7
	Counters and Timers Instructions . . . . .	7
	Fast I/O Instructions . . . . .	8
	Loadable DX . . . . .	9
	Math Instructions . . . . .	9
	Matrix Instructions . . . . .	11
	Miscellaneous . . . . .	12
	Move Instructions . . . . .	13
	Skips/Specials . . . . .	13
	Special Instructions . . . . .	14
	Coils, Contacts and Interconnects . . . . .	14
<b>Chapter 3</b>	<b>Closed Loop Control / Analog Values . . . . .</b>	<b>15</b>
	At a Glance . . . . .	15
	Closed Loop Control / Analog Values . . . . .	16
	PCFL Subfunctions . . . . .	17
	A PID Example . . . . .	21
	PID2 Level Control Example . . . . .	25
<b>Chapter 4</b>	<b>Formatting Messages for ASCII READ/WRITE Operations . . . . .</b>	<b>29</b>

---

	At a Glance . . . . .	29
	Formatting Messages for ASCII READ/WRITE Operations . . . . .	30
	Format Specifiers . . . . .	31
	Special Set-up Considerations for Control/Monitor Signals Format . . . . .	34
<b>Chapter 5</b>	<b>Interrupt Handling . . . . .</b>	<b>37</b>
	Interrupt Handling . . . . .	37
<b>Chapter 6</b>	<b>Subroutine Handling . . . . .</b>	<b>39</b>
	Subroutine Handling . . . . .	39
<b>Chapter 7</b>	<b>Installation of DX Loadables . . . . .</b>	<b>41</b>
	Installation of DX Loadables . . . . .	41
<b>Chapter 8</b>	<b>Coils, Contacts and Interconnects . . . . .</b>	<b>43</b>
	At a Glance . . . . .	43
	Coils . . . . .	44
	Contacts . . . . .	46
	Interconnects (Shorts) . . . . .	48
<b>Part II</b>	<b>Instruction Descriptions . . . . .</b>	<b>49</b>
	At a Glance . . . . .	49
<b>Chapter 9</b>	<b>AD16: Ad 16 Bit . . . . .</b>	<b>55</b>
	At a Glance . . . . .	55
	Short Description . . . . .	56
	Representation . . . . .	56
<b>Chapter 10</b>	<b>ADD: Addition . . . . .</b>	<b>57</b>
<b>Chapter 11</b>	<b>AND: Logical And . . . . .</b>	<b>59</b>
<b>Chapter 12</b>	<b>BCD: Binary to Binary Code . . . . .</b>	<b>63</b>
<b>Chapter 13</b>	<b>BLKM: Block Move . . . . .</b>	<b>65</b>
<b>Chapter 14</b>	<b>BLKT: Block to Table . . . . .</b>	<b>69</b>
<b>Chapter 15</b>	<b>BMDI: Block Move with Interrupts Disabled . . . . .</b>	<b>73</b>
<b>Chapter 16</b>	<b>BROT: Bit Rotate . . . . .</b>	<b>75</b>
<b>Chapter 17</b>	<b>CHS: Configure Hot Standby . . . . .</b>	<b>79</b>
<b>Chapter 18</b>	<b>CKSM: Check Sum . . . . .</b>	<b>85</b>
<b>Chapter 19</b>	<b>CMPR: Compare Register . . . . .</b>	<b>89</b>
<b>Chapter 20</b>	<b>COMP: Complement a Matrix . . . . .</b>	<b>93</b>

---

---

Chapter 21	DCTR: Down Counter . . . . .	97
Chapter 22	DIOH: Distributed I/O Health . . . . .	99
Chapter 23	DIV: Divide . . . . .	103
Chapter 24	DLOG: Data Logging for PCMCIA Read/Write Support . . . . .	107
Chapter 25	DRUM: DRUM Sequencer . . . . .	113
Chapter 26	DV16: Divide 16 Bit . . . . .	117
Chapter 27	EMTH: Extended Math . . . . .	121
Chapter 28	EMTH-ADDDP: Double Precision Addition . . . . .	127
Chapter 29	EMTH-ADDFP: Floating Point Addition . . . . .	131
Chapter 30	EMTH-ADDIF: Integer + Floating Point Addition . . . . .	135
Chapter 31	EMTH-ANLOG: Base 10 Antilogarithm . . . . .	139
Chapter 32	EMTH-ARCOS: Floating Point Arc Cosine of an Angle (in Radians) . . . . .	143
Chapter 33	EMTH-ARSIN: Floating Point Arcsine of an Angle (in Radians) . . . . .	147
Chapter 34	EMTH-ARTAN: Floating Point Arc Tangent of an Angle (in Radians) . . . . .	151
Chapter 35	EMTH-CHSIN: Changing the Sign of a Floating Point Number . . . . .	155
Chapter 36	EMTH-CMPFP: Floating Point Comparison . . . . .	159
Chapter 37	EMTH-CMPIF: Integer-Floating Point Comparison . . . . .	163
Chapter 38	EMTH-CNVDR: Floating Point Conversion of Degrees to Radians . . . . .	167
Chapter 39	EMTH-CNVFI: Floating Point to Integer Conversion . . . . .	171
Chapter 40	EMTH-CNVIF: Integer-to-Floating Point Conversion . . . . .	175
Chapter 41	EMTH-CNVRD: Floating Point Conversion of Radians to Degrees . . . . .	179
Chapter 42	EMTH-COS: Floating Point Cosine of an Angle (in Radians) . . . . .	183

---

---

Chapter 43	EMTH-DIVDP: Double Precision Division . . . . .	187
Chapter 44	EMTH-DIVFI: Floating Point Divided by Integer . . . . .	191
Chapter 45	EMTH-DIVFP: Floating Point Division . . . . .	195
Chapter 46	EMTH-DIVIF: Integer Divided by Floating Point . . . . .	199
Chapter 47	EMTH-ERLOG: Floating Point Error Report Log. . . . .	203
Chapter 48	EMTH-EXP: Floating Point Exponential Function. . . . .	207
Chapter 49	EMTH-LNFP: Floating Point Natural Logarithm . . . . .	211
Chapter 50	EMTH-LOG: Base 10 Logarithm . . . . .	215
Chapter 51	EMTH-LOGFP: Floating Point Common Logarithm . . . . .	219
Chapter 52	EMTH-MULDP: Double Precision Multiplication. . . . .	223
Chapter 53	EMTH-MULFP: Floating Point Multiplication. . . . .	227
Chapter 54	EMTH-MULIF: Integer x Floating Point Multiplication . . . . .	231
Chapter 55	EMTH-PI: Load the Floating Point Value of "Pi" . . . . .	235
Chapter 56	EMTH-POW: Raising a Floating Point Number to an Integer Power . . . . .	239
Chapter 57	EMTH-SINE: Floating Point Sine of an Angle (in Radians) .	243
Chapter 58	EMTH-SQRFP: Floating Point Square Root. . . . .	247
Chapter 59	EMTH-SQRT: Floating Point Square Root. . . . .	251
Chapter 60	EMTH-SQRTP: Process Square Root. . . . .	255
Chapter 61	EMTH-SUBDP: Double Precision Subtraction . . . . .	259
Chapter 62	EMTH-SUBFI: Floating Point - Integer Subtraction . . . . .	263
Chapter 63	EMTH-SUBFP: Floating Point Subtraction . . . . .	267
Chapter 64	EMTH-SUBIF: Integer - Floating Point Subtraction . . . . .	271
Chapter 65	EMTH-TAN: Floating Point Tangent of an Angle (in Radians) . . . . .	275
<b>Chapter 66</b>	<b>ESI: Support of the ESI Module . . . . .</b>	<b>279</b>
<b>Chapter 67</b>	<b>EUCA: Engineering Unit Conversion and Alarms . . . . .</b>	<b>297</b>

---

---

<b>Chapter 68</b>	<b>FIN: First In</b> .....	<b>309</b>
<b>Chapter 69</b>	<b>FOUT: First Out</b> .....	<b>313</b>
<b>Chapter 70</b>	<b>FTOI: Floating Point to Integer</b> .....	<b>317</b>
<b>Chapter 71</b>	<b>HLTH: History and Status Matrices</b> .....	<b>319</b>
<b>Chapter 72</b>	<b>IBKR: Indirect Block Read</b> .....	<b>333</b>
<b>Chapter 73</b>	<b>IBKW: Indirect Block Write</b> .....	<b>335</b>
<b>Chapter 74</b>	<b>ICMP: Input Compare</b> .....	<b>337</b>
<b>Chapter 75</b>	<b>ID: Interrupt Disable</b> .....	<b>343</b>
<b>Chapter 76</b>	<b>IE: Interrupt Enable</b> .....	<b>347</b>
<b>Chapter 77</b>	<b>IMIO: Immediate I/O</b> .....	<b>351</b>
<b>Chapter 78</b>	<b>IMOD: Interrupt Module Instruction</b> .....	<b>357</b>
<b>Chapter 79</b>	<b>ITMR: Interrupt Timer</b> .....	<b>365</b>
<b>Chapter 80</b>	<b>ITOF: Integer to Floating Point</b> .....	<b>371</b>
<b>Chapter 81</b>	<b>JSR: Jump to Subroutine</b> .....	<b>373</b>
<b>Chapter 82</b>	<b>LAB: Label for a Subroutine</b> .....	<b>375</b>
<b>Chapter 83</b>	<b>LOAD: Load Flash</b> .....	<b>379</b>
<b>Chapter 84</b>	<b>MAP 3: MAP Transaction</b> .....	<b>383</b>
<b>Chapter 85</b>	<b>MBIT: Modify Bit</b> .....	<b>391</b>
<b>Chapter 86</b>	<b>MBUS: MBUS Transaction</b> .....	<b>395</b>
<b>Chapter 87</b>	<b>MRTM: Multi-Register Transfer Module</b> .....	<b>405</b>
<b>Chapter 88</b>	<b>MSTR: Master</b> .....	<b>411</b>
<b>Chapter 89</b>	<b>MU16: Multiply 16 Bit</b> .....	<b>453</b>
<b>Chapter 90</b>	<b>MUL: Multiply</b> .....	<b>455</b>
<b>Chapter 91</b>	<b>NBIT: Bit Control</b> .....	<b>457</b>
<b>Chapter 92</b>	<b>NCBT: Normally Closed Bit</b> .....	<b>459</b>
<b>Chapter 93</b>	<b>NOBT: Normally Open Bit</b> .....	<b>461</b>

---

<b>Chapter 94</b>	<b>NOL: Network Option Module for Lonworks</b> . . . . .	<b>463</b>
<b>Chapter 95</b>	<b>OR: Logical OR</b> . . . . .	<b>467</b>

<b>Index</b>	. . . . .	<b>i</b>
--------------	-----------	----------

**The chapters marked gray are not included in this volume.**

Chapter 96	PCFL: Process Control Function Library . . . . .	471
Chapter 97	PCFL-AIN: Analog Input . . . . .	479
Chapter 98	PCFL-ALARM: Central Alarm Handler . . . . .	485
Chapter 99	PCFL-AOUT: Analog Output . . . . .	489
Chapter 100	PCFL-AVER: Average Weighted Inputs Calculate . . . . .	493
Chapter 101	PCFL-CALC: Calculated preset formula . . . . .	499
Chapter 102	PCFL-DELAY: Time Delay Queue . . . . .	503
Chapter 103	PCFL-EQN: Formatted Equation Calculator . . . . .	509
Chapter 104	PCFL-INTEG: Integrate Input at Specified Interval . . . . .	515
Chapter 105	PCFL-KPID: Comprehensive ISA Non Interacting PID . . . . .	519
Chapter 106	PCFL-LIMIT: Limiter for the Pv . . . . .	525
Chapter 107	PCFL-LIMV: Velocity Limiter for Changes in the Pv . . . . .	529
Chapter 108	PCFL-LKUP: Look-up Table . . . . .	533
Chapter 109	PCFL-LLAG: First-order Lead/Lag Filter . . . . .	537
Chapter 110	PCFL-MODE: Put Input in Auto or Manual Mode . . . . .	541
Chapter 111	PCFL-ONOFF: ON/OFF Values for Deadband . . . . .	545
Chapter 112	PCFL-PI: ISA Non Interacting PI . . . . .	551
Chapter 113	PCFL-PID: PID Algorithms . . . . .	555
Chapter 114	PCFL-RAMP: Ramp to Set Point at a Constant Rate . . . . .	561
Chapter 115	PCFL-RATE: Derivative Rate Calculation over a Specified Timeme . . . . .	567
Chapter 116	PCFL-RATIO: Four Station Ratio Controller . . . . .	571

---



---

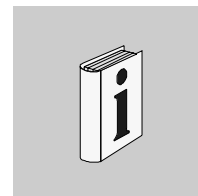
Chapter 117	PCFL-RMPLN: Logarithmic Ramp to Set Point . . . . .	577
Chapter 118	PCFL-SEL: Input Selection . . . . .	581
Chapter 119	PCFL-TOTAL: Totalizer for Metering Flow . . . . .	585
Chapter 120	PEER: PEER Transaction. . . . .	591
Chapter 121	PID2: Proportional Integral Derivative . . . . .	595
Chapter 122	R → T: Register to Table . . . . .	609
Chapter 123	RBIT: Reset Bit . . . . .	613
Chapter 124	READ: Read. . . . .	615
Chapter 125	RET: Return from a Subroutine. . . . .	621
Chapter 126	SAVE: Save Flash . . . . .	623
Chapter 127	SBIT: Set Bit . . . . .	627
Chapter 128	SCIF: Sequential Control Interfaces . . . . .	629
Chapter 129	SENS: Sense . . . . .	635
Chapter 130	SKPC: Skip (Constants). . . . .	639
Chapter 131	SKPR: Skip (Registers) . . . . .	643
Chapter 132	SRCH: Search . . . . .	647
Chapter 133	STAT: Status . . . . .	651
Chapter 134	SU16: Subtract 16 Bit . . . . .	675
Chapter 135	SUB: Subtraction . . . . .	677
Chapter 136	T → R: Table to Register . . . . .	679
Chapter 137	T → T: Table to Table. . . . .	683
Chapter 138	T.01 Timer: One Hundredth Second Timer. . . . .	687
Chapter 139	T0.1 Timer: One Tenth Second Timer. . . . .	689
Chapter 140	T1.0 Timer: One Second Timer . . . . .	691
Chapter 141	T1MS Timer: One Millisecond Timer. . . . .	693
Chapter 142	TBLK: Table to Block . . . . .	699

---

<b>Chapter 143</b>	<b>TEST: Test of 2 Values</b> .....	<b>703</b>
<b>Chapter 144</b>	<b>UCTR: Up Counter</b> .....	<b>705</b>
<b>Chapter 145</b>	<b>WRIT: Write</b> .....	<b>707</b>
<b>Chapter 146</b>	<b>XMIT: XMIT Communication Block</b> .....	<b>713</b>
<b>Chapter 147</b>	<b>XMRD: Extended Memory Read</b> .....	<b>723</b>
<b>Chapter 148</b>	<b>XMWT: Extended Memory Write</b> .....	<b>727</b>
<b>Chapter 149</b>	<b>XOR: Exclusive OR</b> .....	<b>731</b>
<b>Glossary</b>	.....	<b>735</b>

---

## About the book



---

### At a Glance

**Document Scope** This documentation will help you configure the LL984-instructions from Concept.

**Validity Note** This documentation is valid for Concept 2.6 under Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows NT 4.x.

**Note:** For additional up-to-date notes, please refer to the file README of Concept.

---

### Related Documents

Title of Documentation	Reference Number
Concept Installation Instruction	840 USE 502 00
Concept User Manual	840 USE 503 00
Concept IEC Library	840 USE 504 00
Concept-EFB User Manual	840 USE 505 00
XMIT Function Block User Guide	840 USE 113 00
Network Option Module for LonWorks	840 USE 109 00
Quantum Hot Standby Planning and Installation Guide	840 USE 106 00
Modbus Plus Network Planning and Installation Guide	890 USE 100 00
Quantum 140 ESI 062 10 ASCII Interface Module User Guide	840 USE 1116 00
Modicon S980 MAP 3.0 Network Interface Controller User Guide	GM-MAP3-001

---

**User Comments** We welcome your comments about this document. You can reach us by e-mail at [TECHCOMM@modicon.com](mailto:TECHCOMM@modicon.com)

---

About the book

---

---

## ESI: Support of the ESI Module

66

---

### At a Glance

#### Introduction

This chapter describes the instruction ESI.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	280
Representation	281
Parameter Description	282
READ ASCII Message (Subfunction 1)	285
WRITE ASCII Message (Subfunction 2)	289
GET DATA (Subfunction 3)	290
PUT DATA (Subfunction 4)	291
ABORT (Middle Input ON)	295
Run Time Errors	296

---

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

The instruction for the ESI module 140 ESI 062 10 are optional loadable instructions that can be used in a Quantum controller system to support operations using a ESI module. The controller can use the ESI instruction to invoke the module. The power of the loadable is its ability to cause a sequence of commands over one or more logic scans.

With the ESI instruction, the controller can invoke the ESI module to:

- Read an ASCII message from a serial port on the ESI module, then perform a sequence of GET DATA transfers from the module to the controller.
- Write an ASCII message to a serial port on the ESI module after having performed a sequence of PUT DATA transfers to the variable data registers in the module.
- Perform a sequence of GET DATA transfers (up to 16 384 registers of data from the ESI module to the controller); one Get Data transfer will move up to 10 data registers each time the instruction is solved.
- Perform a sequence of PUT DATA (up to 16 384 registers of data to the ESI module from the controller). One PUT DATA transfer moves up to 10 registers of data each time the instruction is solved.
- Abort the ESI loadable command sequence running.

Further Information you will find in the *Quantum 140 ESI 062 10 ASCII Interface Module User Guide*

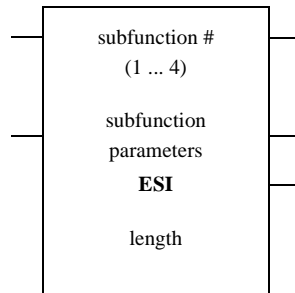
**Note:** After placing the ESI instruction in your ladder diagram you must enter the top, middle and bottom parameters. Proceed by double clicking on the instruction. This action produces a form for the entry of the 3 paramteers. This parametric must be completed to enable the DX zoom function in the `EDIT` menu pulldown.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables the subfunction
Middle input	0x, 1x	None	Abort current message
subfunction (top node)	4x	INT, UINT, WORD	Number of possible subfunction, range 1 ... 4
subfunction parameters (middle node)	4x	INT, UINT, WORD	First of eighteen contiguous 4x holding registers which contain the subfunction parameters
length bottom node		INT, UINT	Number of subfunction parameter registers, i.e. the length of the table in the middle node
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = operation done
Bottom output	0x	None	ON = error detected

## Parameter Description

---

<b>Top Input</b>	When the input to the top node is powered ON, it enables the ESI instruction and starts executing the command indicated by the subfunction code in the top node.
<b>Middle Input</b>	When the input to the middle node is powered ON, an Abort command is issued. If a message is running when the ABORT command is received, the instruction will complete; if a data transfer is in process when the ABORT command is received, the transfer will stop and the instruction will complete.
<b>Subfunction # (Top Node)</b>	The top node may contain either a 4x register or an integer. The integer or the value in the register must be in the range 1 ... 4. It represents one of four possible subfunction command sequences to be executed by the instruction:

Subfunction	Command Sequence
1	One command <i>READ ASCII Message, p. 285</i> followed by multiple GET DATA commands
2	Multiple PUT DATA commands followed by one command <i>WRITE ASCII Message, p. 289</i>
3	Zero or more commands <i>GET DATA, p. 290</i>
4	Zero or more commands <i>PUT DATA, p. 291</i>

<b>Note:</b> A fifth command, ABORT ASCII Message (See <i>ABORT, p. 295</i> ), can be initiated by enabling the middle input to the ESI instruction.
--

---



**Subfunction Parameters (Middle Node)**

The first of eighteen contiguous 4x registers is entered in the middle node. The remaining seventeen registers are implied. The following subfunction parameters are available:

Register	Parameter	Contents
Displayed	ESI status register	Returned error codes
First implied	Address of the first 4x register in the command structure	Register address minus the leading 4 and any leading zeros, as specified in the I/O Map (e.g., 1 represents register 400001)
Second implied	Address of the first 3x register in the command structure	Register address minus the leading 3 and any leading zeros, as specified in the I/O Map (e.g., 7 represents register 300007)
Third implied	Address of the first 4x register in the controller's data register area	Register address minus the leading 4 and any leading zeros (e.g., 100 representing register 400100)
Fourth implied	Address of the first 3x register in the controller's data register area	Register address minus the leading 3 and any leading zeros (e.g., 1000 representing register 301000)
Fifth implied	Starting register for data register area in module	Number in the range 0 ... 3FFF hex
Sixth implied	Data transfer count	Number in the range 0 ... 4000 hex
Seventh implied	ESI timeout value, in 100 ms increments	Number in the range 0 ... FFFF hex, where 0 means no timeout
Eighth implied	ASCII message number	Number in the range 1 ... 255 dec
Ninth implied	ASCII port number	1 or 2
<b>Note:</b> The registers below are internally used by the ESI loadable. Do not write registers while the ESI loadable is running. For best use, initialize these registers to 0 (zero) when the loadable is inserted into logic.		
10th implied	ESI loadable previous scan power in state	
11th implied	Data left to transfer	
12th implied	Current ASCII module command running	
13th implied	ESI loadable sequence number	
14th implied	ESI loadable flags	
15th implied	ESI loadable timeout value (MSW)	
16th implied	ESI loadable timeout value (LSW)	
17th implied	Parameter Table Checksum generated by ESI loadable	

**Note:** Once power has been applied to the top input, the ESI loadable starts running. Until the ESI loadable compiles (successfully or in error), the subfunction parameters should not be modified. If the ESI loadable detects a change, the loadable will compile in error (Parameter Table Checksum Error (See *Run Time Errors*, p. 296)).

---

**Length (Bottom Node)**

The bottom node contains the length of the table in the middle node, i.e., the number of subfunction parameter registers. For READ/ WRITE operations, the length must be 10 registers. For PUT/GET operations, the required length is eight registers; 10 may be specified and the last two registers will be unused.

---

**Oupputs**

**Note:** NSUP must be loaded before ESI in order for the loadable to work properly. If ESI is loaded before NSUP or ESI is loaded alone, all three outputs will be turned ON.

---

**Middle Output**

The middle output goes ON for one scan when the subfunction operation specified in the top node is completed, timed out, or aborted

---

**Bottom Output**

The bottom output goes ON for one scan if an error has been detected. Error checking is the first thing that is performed on the instruction when it is enabled, it is completed before the subfunction is executed. For more details see error checking (See *Run Time Errors*, p. 296).

---

## READ ASCII Message (Subfunction 1)

### READ ASCII Message

A READ ASCII command causes the ESI module to read incoming data from one of its serial ports and store the data in internal variable data registers. The serial port number is specified in the tenth (ninth implied) register of the subfunction parameters table. The ASCII message number to be read is specified in the ninth (eighth implied) register of the subfunction parameters table (See *Subfunction Parameters (Middle Node)*, p. 283). The received data is stored in the 16K variable data space in user-programmed formats.

When the top node of the ESI instruction is 1, the controller invokes the module and causes it to execute one READ ASCII command followed by a sequence of GET DATA commands (transferring up to 16,384 registers of data) from the module to the controller.

### Command Structure

Command Structure

Word	Content (hex)	Meaning
0	01PD	P = port number (1 or 2); D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	00xx	Message number, where xx is in the range 1 ... FF (1 ... 255 dec)
3 ... 11	Not used	

### Response Structure

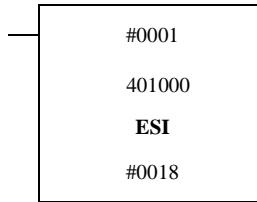
Command Structure

Word	Content (hex)	Meaning
0	01PD	Echoes command word 0
1	xxxx	Echoes starting register number from Command Word 1
2	00xx	Echoes message number from Command Word 2
3	xxxx	Data word 1
4	xxxx	Data word 2
...	...	...
11	xxxx	Module status or data word 9

**A Comparative  
READ ASCII  
Message/Put  
Data Example**

Below is an example of how an ESI loadable instruction can simplify your logic programming task in an ASCII read application. Assume that the 12-point bidirectional ESI module has been I/O mapped to 400001 ... 400012 output registers and 300001 ... 300012 input registers. We want to read ASCII message #10 from port 1, then transfer four words of data to registers 400501 ... 400504 in the controller.

Parameterizing of the ESI instruction:



The subfunction parameter table begins at register 401000 . Enter the following parameters in the table:

Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	Module start register
401006	4	Number of registers to transfer
401007	600	timeout = 60 s
401008	10	ASCII message number
401009	1	ASCII port number
401010-17	N/A	Internal loadable variables

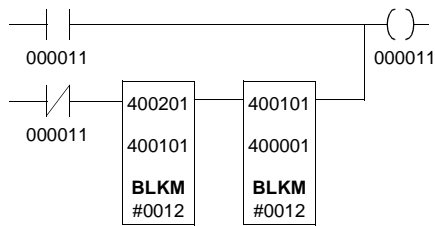
With these parameters entered to the table, the ESI instruction will handle the read and data transfers automatically in one scan.

---

**Read and Data Transfers without ESI Instruction**

The same task could be accomplished in ladder logic **without** the ESI loadable, but it would require the following three networks to set up the command and transfer parameters, then copy the data. Registers 400101 ... 400112 are used as workspace for the output values. Registers 400201 ... 400212 are initial READ ASCII Message command values. Registers 400501 ... 400504 are the data space for the received data from the module.

**First Network**

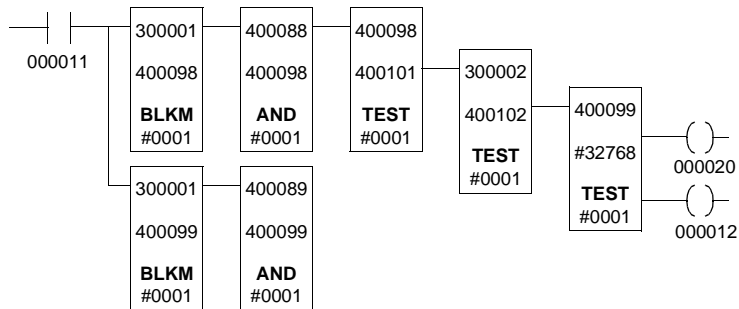


Contents of registers

Register	Value (hex)	Description
400201	0114	READ ASCII Message command, Port 1, Four registers
400202	0064	Module's starting register
400203	nxxx	Not valid: data word 1
...	...	...
400212	nxxx	Not valid: data word 10

The first network starts up the READ ASCII Message command by turning ON coil 000011 forever. It moves the READ ASCII Message command into the workspace, then moves the workspace to the output registers for the module.

**Second Network**



Contents of registers

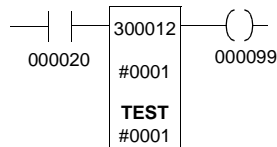
Register	Value (hex)	Description
400098	nnnn	Workspace for response word
400099	nnnn	Workspace for response word
400088	7FFF	Response word mask
400089	8000	Status word valid bit mask

As long as coil 000011 is ON, READ ASCII Message response Word 0 in the input register is tested to make sure it is the same as command Word 0 in the workspace. This is done by ANDing response Word 0 in the input register with 7FFF hex to get rid of the Status Word Valid bit (bit 15) in Response Word 0.

The module start register in the input register is also tested against the module start register in the workspace to make sure that are the same.

If both these tests show matches, test the Status Word Valid bit in response Word 0. To do this, AND response Word 0 in the input register with 8000 hex to get rid of the echoed command word 0 information. If the ANDed result equals the Status Word Valid bit, coil 000020 is turned ON indicating an error and/or status in the Module Status Word. If the ANDed result is not the status word valid bit, coil 000012 is turned ON indicating that the message is done and that you can start another command in the module.

**Third Network**



If coil 000020 is ON, this third network will test the Module Status Word for busy status. If the module is busy, do nothing. If the Module Status Word is greater than 1 (busy), a detected error has been logged in the high byte and coil 000099 will be turned ON. At this point, you need to determine what the error is using some error-handling logic that you have developed.

---

## WRITE ASCII Message (Subfunction 2)

### WRITE ASCII Message

In a WRITE ASCII Message command, the ESI module writes an ASCII message to one of its serial ports. The serial port number is specified in the tenth (ninth implied) register of the subfunction parameters table (See *Subfunction Parameters (Middle Node)*, p. 283). The ASCII message number to be written is specified in the ninth (eighth implied) register of the subfunction parameters table.

When the top node of the ESI instruction is 2, the controller invokes the module and causes it to execute one Write ASCII command. Before starting the WRITE command, subfunction 2 executes a sequence of PUT DATA transfers (transferring up to 16 384 registers of data) from the controller to the module.

### Command Structure

Command Structure

Word	Content (hex)	Meaning
0	02PD	P = port number (1 or 2); D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	00xx	Message number, where xx is in the range 1 ... FF (1 ... 255 dec)
3	xxxx	Data word 1
4	xxxx	Data word 2
...	...	...
11	xxxx	Data word 9

### Response Structure

Response Structure

Word	Content (hex)	Meaning
0	02PD	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	00xx	Echoes message number from command word 2
3	0000	Returns a zero
...	...	...
10	0000	Returns a zero
11	xxxx	Module status

## GET DATA (Subfunction 3)

---

### GET DATA

A GET DATA command transfers up to 10 registers of data from the ESI module to the controller each time the ESI instruction is solved in ladder logic. The total number of words to be read is specified in Word 0 of the GET DATA command structure (the data count). The data is returned in increments of 10 in Words 2 ... 11 in the GET DATA response structure.

If a sequence of GET DATA commands is being executed in conjunction with a READ ASCII Message command (via subfunction 1), up to nine registers are transferred when the instruction is solved the first time. Additional data are returned in groups of ten registers on subsequent solves of the instruction until all the data has been transferred.

If there is an error condition to be reported (other than a command syntax error), it is reported in Word 11 in the GET DATA response structure. If the command has requested 10 registers and the error needs to be reported, only nine registers of data will be returned in Words 2 ... 10, and Word 11 will be used for error status.

**Note:** If the data count and starting register number that you specify are valid but some of the registers to be read are beyond the valid register range, only data from the registers in the valid range will be read. The data count returned in Word 0 of the response structure will reflect the number of valid data registers returned, and an error code (1280 hex) will be returned in the Module Status Word (Word 11 in the response table).

### Command Structure

#### Command Structure

Word	Content (hex)	Meaning
0	030D	D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2 ... 11	Not used	

---



**Response Structure**

Response Structure

Word	Content (hex)	Meaning
0	030D	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	xxxx	Data word 1
3	xxxx	Data word 2
...	...	...
11	xxxx	Module status or data word 10

**PUT DATA (Subfunction 4)**

**PUT DATA**

A PUT DATA command writes up to 10 registers of data to the ESI module from the controller each time the ESI instruction is solved in ladder logic. The total number of words to be written is specified in Word 0 of the PUT DATA command structure (the data count).

The data is returned in increments of 10 in words 2 ... 11 in the PUT DATA command structure. The command is executed sequentially until command word 0 changes to another command other than PUT DATA (040D hex).

**Note:** If the data count and starting register number that you specify are valid but some of the registers to be written are beyond the valid register range, only data from the registers in the valid range will be written. The data count returned in Word 0 of the response structure will reflect the number of valid data registers returned, and an error code (1280 hex) will be returned in the Module Status Word (Word 11 in the response table).

**Command Structure**

Command Structure

Word	Content (hex)	Meaning
0	040D	D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	xxxx	Data word 1
3	xxxx	Data word 2
...	...	...
11	xxxx	Data word 10

**Response Structure**

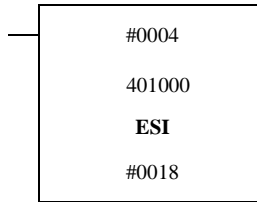
Response Structure

Word	Content (hex)	Meaning
0	040D	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	0000	Returns a zero
...	...	...
10	0000	Returns a zero
11	xxxx	Module status

**A Comparative PUT DATA Example**

Below is an example of how an ESI loadable instruction can simplify your logic programming task in a PUT DATA application. Assume that the 12-point bidirectional ESI 062 module has been I/O mapped to 400001 ... 400012 output registers and 300001 ... 300012 input registers. We want to put 30 controller data registers, starting at register 400501, to the ESI module starting at location 100.

Parameterizing of the ESI instruction:



The subfunction parameter table begins at register 401000 . Enter the following parameters in the table:

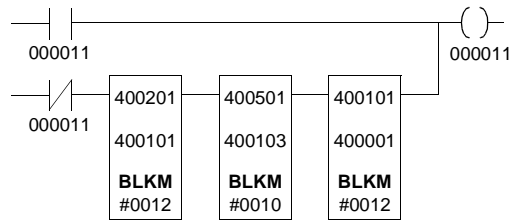
Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	Module start register
401006	30	Number of registers to transfer
401007	0	timeout = never
401008	N/A	ASCII message number
401009	N/A	ASCII port number
401009	N/A	Internal loadable variables

With these parameters entered to the table, the ESI instruction will handle the data transfers automatically over three ESI logic solves.

**Handling of Data Transfer without ESI Instruction**

The same task could be accomplished in ladder logic **without** the ESI loadable, but it would require the following four networks to set up the command and transfer parameters, then copy data multiple times until the operation is complete. Registers 400101 ... 400112 are used as workspace for the output values. Registers 400201 ... 400212 are initial PUT DATA command values. Registers 400501 ... 400530 are the data registers to be sent to the module.

**First Network - Command Register Network**

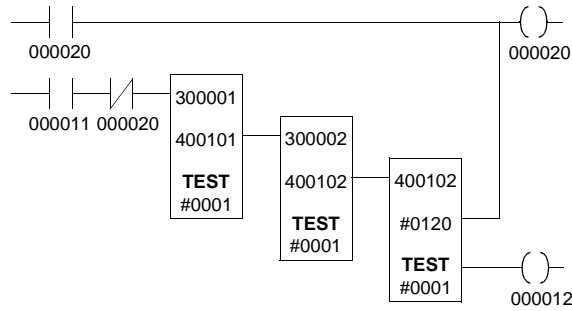


Contents of registers

Register	Value (hex)	Description
400201	040A	PUT DATA command, 10 registers
400202	0064	Module's starting register
400203	nxxx	Not valid: data word 1
...	...	...
400212	nxxx	Not valid: data word 10

The first network starts up the transfer of the first 10 registers by turning ON coil 000011 forever. It moves the initial PUT DATA command into the workspace, moves the first 10 registers (400501 ... 400510) into the workspace, and then moves the workspace to the output registers for the module.

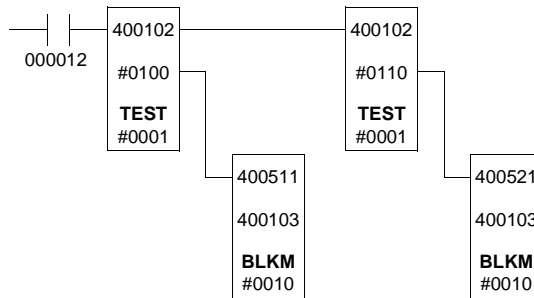
**Second Network - Command Register Network**



As long as coil 000011 is ON and coil 000020 is OFF, PUT DATA response word 0 in the input register is tested to make sure it is the same as the command word in the workspace. The module start register in the input register is also tested to make sure it is the same as the module start register in the workspace.

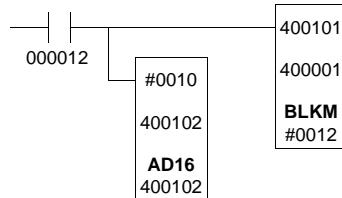
If both these tests show matches, the current module start register is tested against what would be the module start register of the last PUT DATA command for this transfer. If the test shows that the current module start register is greater than or equal to the last PUT DATA command, coil 000020 goes ON indicating that the transfer is done. If the test shows that the current module start register is less than the last PUT DATA command, coil 000012 indicating that the next 10 registers should be transferred.

**Third Network - Command Register Network**



As long as coil 000012 is ON, there is more data to be transferred. The module start register needs to be tested from the last command solve to determine which set of 10 registers to transfer next. For example, if the last command started with module register 400110, then the module start register for this command is 400120.

**Fourth Network - Command Register Network**



As long as coil 000012 is ON, add 10 to the module start register value in the workspace and move the workspace to the output registers for the module to start the next transfer of 10 registers.

**ABORT (Middle Input ON)**

**ABORT**

When the middle input to the ESI instruction is powered ON, the instruction aborts a running ASCII READ or WRITE message. The serial port buffers of the module are not affected by the ABORT, only the message that is currently running.

**Command Structure**

Command Structure

Word	Content (hex)
0	0900
1 ... 11	not used

**Response Structure**

Response Structure

Word	Content (hex)	Meaning
0	0900	Echoes command word 0
1	0000	Returns a zero
...	...	...
10	0000	Returns a zero
11	xxxx	Module status

## Run Time Errors

---

**Run Time Errors** The command sequence executed by the ESI module (specified by the subfunction value (See *Subfunction # (Top Node)*, p. 282) in the top node of the ESI instruction) needs to go through a series of error checking routines before the actual command execution begins. If an error is detected, a message is posted in the register displayed in the middle node.

The following table lists possible error message codes and their meanings:

Error Code (dec)	Meaning
0001	Unknown subfunction specified in the top node
0010	ESI instruction has timed out (exceeded the time specified in the eighth register of the subfunction parameter table (See <i>Subfunction Parameters (Middle Node)</i> , p. 283)
0101	Error in the READ ASCII Message sequence
0102	Error in the WRITE ASCII Message sequence
0103	Error in the GET DATA sequence
0104	Error in the PUT DATA sequence
1000	<i>Length (Bottom Node)</i> , p. 284 is too small
1001	Nonzero value in both the 4x and 3x data offset parameters
1002	Zero value in both the 4x and 3x data offset parameters
1003	4x or 3x data offset parameter out of range
1004	4x or 3x data offset plus transfer count out of range
1005	3x data offset parameter set for GET DATA
1006	Parameter Table Checksum error
1101	Output registers from the offset parameter out of range
1102	Input registers from the offset parameter out of range
2001	Error reported from the ESI module

Once the parameter error checking has completed without finding an error, the ESI module begins to execute the command sequence.

---

---

## EUCA: Engineering Unit Conversion and Alarms

67

---

### At a Glance

#### Introduction

This chapter describes the instruction EUCA.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	298
Representation	299
Parameter Description	300
Examples	301

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information you will find in "*Installation of DX Loadables, p. 41*".

The use of ladder logic to convert binary-expressed analog data into decimal units can be memory-intensive and scan-time intensive operation. The Engineering Unit Conversion and Alarms (EUCA) loadable is designed to eliminate the need for extra user logic normally required for these conversions. EUCA scales 12 bits of binary data (representing analog signals or other variables) into engineering units that are readily usable for display, data logging, or alarm generation.

Using  $Y = mX + b$  linear conversion, binary values between 0 ... 4095 are converted to a scaled process variable (SPV). The SPV is expressed in engineering units in the range 0 ... 9 999.

One EUCA instruction can perform up to four separate engineering unit conversions.

It also provides four levels of alarm checking on each of the four conversions:

Level	Meaning
HA	High absolute
HW	High warning
LW	Low warning
LA	Low absolute

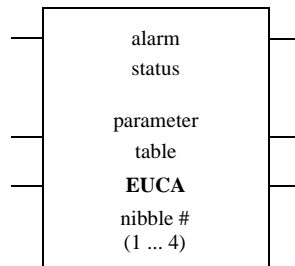
---



## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON initiates the conversion
Middle input	0x, 1x	None	Alarm input
Bottom input	0x, 1x	None	Error input
alarm status (See <i>Alarm Status (Top Node)</i> , p. 300) (top node)	4x	INT, UINT	Alarm status for as many as four EUCA conversions
parameter table (middle node)	4x	INT, UINT,	First of nine contiguous holding registers in the EUCA parameter table
nibble # (1...4) (bottom node)		INT, UINT	Integer value, indicates which one of the four nibbles in the alarm status register to use
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON if the middle input is ON or if the result of the EUCA conversion crosses a warning level
Bottom output	0x	None	ON if the bottom input is ON or if a parameter is out of range

## Parameter Description

---

### Alarm Status (Top Node)

The 4x register entered in the top node displays the alarm status for as many as four EUCA conversions, which can be performed by the instruction. The register is segmented into four four-bit nibbles. Each four-bit nibble represents the four possible alarm conditions for an individual EUCA conversion. The most significant nibble represents the first conversion, and the least significant nibble represents the fourth conversion:

HA1	HW1	LW1	LA1	HA2	HW2	LW2	LA2	HA3	HW3	LW3	LA3	HA4	HW4	LW4	LA4
Nibble 1 (first conversion)				Nibble 2 (second conversion)				Nibble 3 (third conversion)				Nibble 4 (fourth conversion)			

### Alarm Setting

Condition of alarm setting

Alarm type	Condition
HA	An HA alarm is set when the SPV exceeds the user-defined high alarm value expressed in engineering units
HW	An HW alarm is set when SPV exceeds a user-defined high warning value expressed in engineering units
LW	An LW alarm is set when SPV is less than a user-defined low warning value expressed in engineering units
LA	An LA alarm is set when SPV is less than a user-defined low alarm value expressed in engineering units

Only one alarm condition can exist in any EUCA conversion at any given time. If the SPV exceeds the high warning level the HW bit will be set. If the HA is exceeded, the HW bit is cleared and the HA bit is set. The alarm bit will not change after returning to a less severe condition until the deadband (DB) area has also been exited.

---

**Parameter Table  
(Middle Node)**

The 4x register entered in the middle node is the first of nine contiguous holding registers in the EUCA parameter table:

Register	Content	Range
Displayed	Binary value input by the user	0 ... 4 095
First implied	SPV calculated by the EUCA block	
Second implied	High engineering unit (HEU), maximum SPV required and set by the user (top of the scale)	$LEU < HEU \leq 99\,999$
Third implied	Low engineering unit (LEU), minimum SPV required and set by the user (bottom end of the scale)	$0 \leq LEU < HEU$
Fourth implied	DB area in SPV units, below HA levels and above LA levels that must be crossed before the alarm status bit will reset	$0 \leq DB < (HEU - LEU)$
Fifth implied	HA alarm value in SPV units	$HW < HA \leq HEU$
Sixth implied	HW alarm value in SPV units	$LW < HW < HA$
Seventh implied	LW alarm value in SPV units	$LA < LW < HW$
Eighth implied	LA alarm value in SPV units	$LEU \leq LA < LW$

**Note:** An error is generated if any value is out of the range defined above

**Examples**

**Overview**

The following examples are shown:

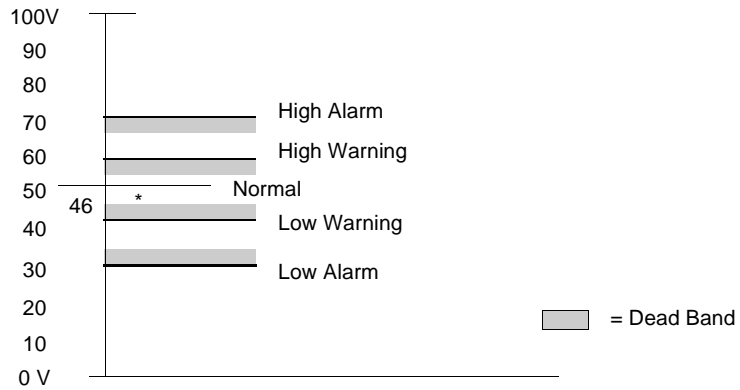
- *Example 1, p. 302*  
Principles of EUCA Operation
- *Example 2, p. 304*  
Use in a Drive System
- *Example 3, p. 306*  
Four EUCA conversions together



Register	Meaning	Content
400452	HIGH_unit	100 DEC
400453	LOW_unit	0 DEC
400454	Dead_band	5 DEC
400455	HIGH_ALARM	70 DEC
400456	HIGH_WARN	60 DEC
400457	LOW_ALARM	40 DEC
400458	LOW_WARN	30 DEC

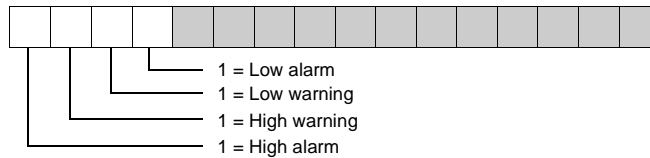
The nine middle-node registers are set using the reference data editor. DB is 5 V followed by 10 V increments of high and low warning. The actual high and low alarm is set at 20 V above and below nominal.

On a graph, the example looks like this:



**Note:** The example value shows a decimal 46, which is in the normal range. No alarm is set, i.e., register 400440 = 0.

You can now verify the instruction in a running PLC by entering values in register 400450 that fall into the defined ranges. The verification is done by observing the bit change in register 400440 where:



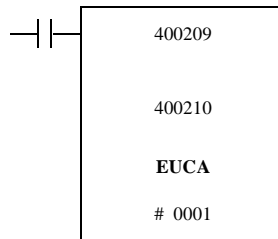
**Example 2**

If the input of 0 ... 4095 indicates the speed of a drive system of 0 ... 5000 rpm, you could set up a EUCA instruction as follows.

The binary value in 400210 results in an SPV of 4835 decimal, which exceeds the high absolute alarm level, sets the HA bit in 400209, and powers the EUCA alarm node.

Parameter	Speed
Maximum Speed	5 000 rpm
Minimum Speed	0 rpm
DB	100 rpm
HA Alarm	4 800 rpm
HW Alarm	4 450 rpm
LW Alarm	2 000 rpm
LA Alarm	1 200 rpm

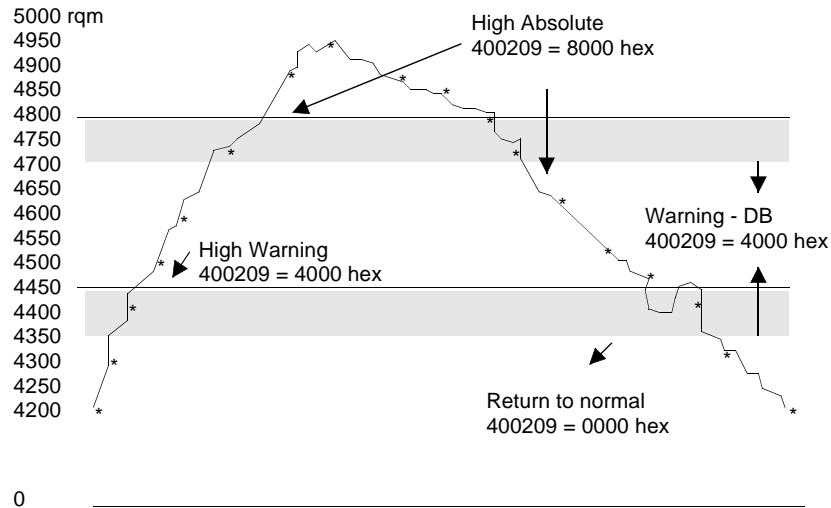
Instruction



Reference Data

Register	Meaning	Content
400209	STATUS	1000000000000000
400210	INPUT	3960 DEC
400211	SPV	4835 DEC
400212	MAX_SPEED	5000 DEC
400213	MIN_SPEED	0 DEC
400214	Dead_band	100 DEC
400215	HIGH_ALARM	4800 DEC
400216	HIGH_WARN	4450 DEC
400217	LOW_ALARM	2000 DEC
400218	LOW_WARN	1200 DEC

The N.O. contact is used to suppress alarm checks when the drive system is shutdown, or during initial start up allowing the system to get above the Low alarm RPM level.

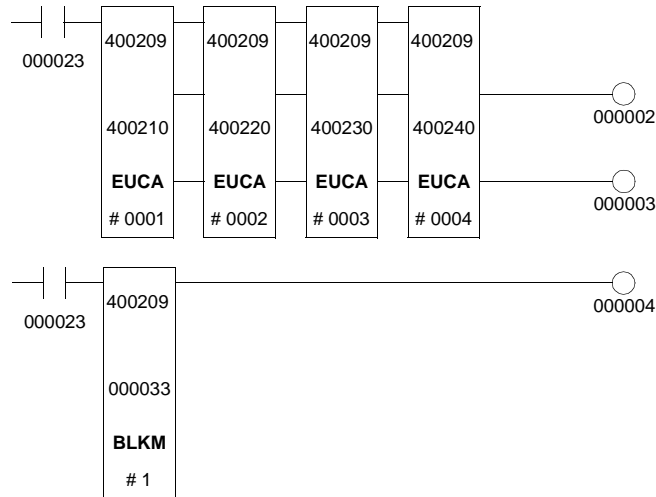


Varying the binary value in register 400210 would cause the bits in nibble 1 of register 400209 to correspond with the changes illustrated above. The DB becomes effective when the alarm or warning has been set, then the signal falls into the DB zone.

The alarm is maintained, thus taking what would be a switch chatter condition out of a marginal signal level. This point is exemplified in the chart above, where after setting the HA alarm and returning to the warning level at 4700 the signal crosses in and out of DB at the warning level (4450) but the warning bit in 400209 stays ON. The same action would be seen if the signal were generated through the low settings.

**Example 3**

You can chain up to four EUCA conversions together to make one alarm status register. Each conversion writes to the nibble defined in the block bottom node. In the program example below, each EUCA block writes its status (based on the table values for that block) into a four bit (nibble) of the status register 400209.



Reference Data

Register	Meaning	Content
400209	STATUS	0000001001001000

The status register can then be transferred using a BLKM instruction to a group of discrettes wired to illuminate lamps in an alarm enunciator panel.

As you observe the status content of register 400209 you see: no alarm in block 1, an LW alarm in block 2, an HW alarm in Block 3, and an HA alarm in block 4.

The alarm conditions for the four blocks can be represented with the following table settings:

	Conversion 1	Conversion 2	Conversion 3	Conversion 4
Input	400210 = 2048	400220 = 1220	400230 = 3022	400240 = 3920
Scaled #	400211 = 2501	400221 = 1124	400231 = 7379	400241 = 0770
HEU	400212 = 5000	400222 = 3300	400232 = 9999	400242 = 0800
LEU	400213 = 0000	400223 = 0200	400233 = 0000	400243 = 0100
DB	400214 = 0015	400224 = 0022	400234 = 0100	400244 = 0006
Hi Alarm	400215 = 40000	400225 = 2900	400235 = 8090	400245 = 0768



EUCA: Engineering Unit Conversion and Alarms

---

	<b>Conversion 1</b>	<b>Conversion 2</b>	<b>Conversion 3</b>	<b>Conversion 4</b>
Hi Warn	400216 = 3500	400226 = 2300	400236 = 7100	400246 = 0680
Lo Warn	400217 = 2000	400227 = 1200	400237 = 3200	400247 = 0280
Lo Alarm	400218 = 1200	400228 = 0430	400238 = 0992	400248 = 0230

---



---

## FIN: First In

68

---

### At a Glance

#### Introduction

This chapter describes the instruction FIN.

#### What's in this chapter?

This chapter contains the following topics:

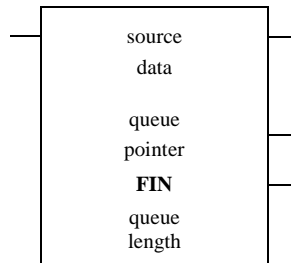
Topic	Page
Short Description	310
Representation	310
Parameter Description	311

## Short Description

**Function Description** The FIN instruction is used to produce a first-in queue. An FOUT instruction needs to be used to clear the register at the bottom of the queue. An FIN instruction has one control input and can produce three possible outputs.

## Representation

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = copies source bit pattern into queue
source data (top node)	0x, 1x, 3x, 4x	ANY_BIT	Source data, will be copied to the top of the destination queue in the current logic scan
queue pointer (See <i>Queue Pointer (Middle Node)</i> , p. 311) (middle node)	4x	WORD	First of a queue of 4x registers, contains queue pointer; the next contiguous register is the first register in the queue
queue length (bottom node)		INT, UINT	Number of 4x registers in the destination queue. Range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = queue full, no more source data can be copied to the queue
Bottom output	0x	None	ON = queue empty (value in queue pointer register = 0)

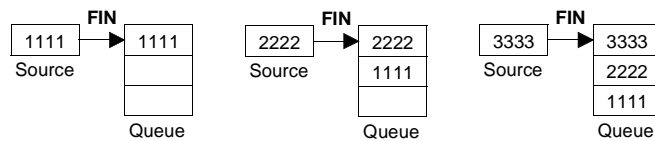
---

## Parameter Description

---

### Mode of Functioning

The FIN instruction is used to produce a first-in queue. It copies the source data from the top node to the first register in a queue of holding registers. The source data is always copied to the register at the top of the queue. When a queue has been filled, no further source data can be copied to it.



### Source Data (Top Node)

When using register types 0x or 1x:

- First 0x reference in a string of 16 contiguous coils or discrete outputs
- First 1x reference in a string of 16 discrete inputs

### Queue Pointer (Middle Node)

The 4x register entered in the middle node is a queue pointer. The first register in the queue is the next contiguous 4x register following the pointer. For example, if the middle node displays a pointer reference of 400100, then the first register in the queue is 400101.

The value posted in the queue pointer equals the number of registers in the queue that are currently filled with source data. The value of the pointer cannot exceed the integer maximum queue length value specified in the bottom node.

If the value in the queue pointer equals the integer specified in the bottom node, the middle output passes power and no further source data can be written to the queue until an FOUT instruction clears the register at the bottom of the queue.

---

FIN: First In

---

---

## FOUT: First Out

69

---

### At a Glance

#### Introduction

This chapter describes the instruction FOUT.

#### What's in this chapter?

This chapter contains the following topics:


Topic	Page
Short Description	314
Representation	315
Parameter Description	316

## Short Description

---

**Function  
Description**

The FOUT instruction works together with the FIN instruction to produce a first in-first out (FIFO) queue. It moves the bit pattern of the holding register at the bottom of a full queue to a destination register or to word that stores 16 discrete outputs. An FOUT instruction has one control input and can produce three possible outputs.

	<b>DANGER</b>
	<p><b>Overriding any disabled coils</b></p> <p>FOUT will override any disabled coils within a destination register without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the FOUT operation.</p> <p><b>Failure to observe this precaution will result in death or serious injury.</b></p>

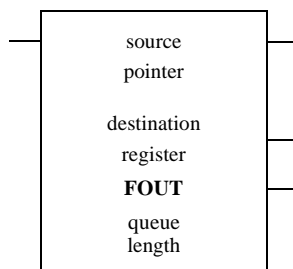
---



## Representation

### Symbol

Representation of the instruction



### Parameter Description

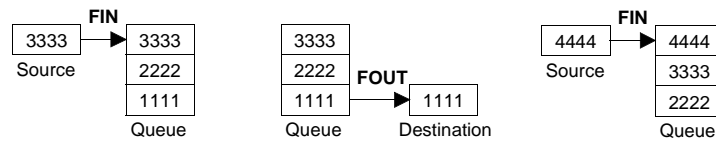
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = clears source bit pattern from the queue
source pointer (top node)	4x	WORD	First of a queue of 4x registers, contains source pointer; the next contiguous register is the first register in the queue
destination register (middle node)	0x, 4x	ANY_BIT	Destination register
queue length (bottom node)		INT, UINT	Number of 4x registers in the queue. Range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = queue full, no more source data can be copied to the queue
Bottom output	0x	None	ON = queue empty (value in queue pointer re

## Parameter Description

### Mode of Functioning

The FOUT instruction works together with the FIN (See *FIN: First In*, p. 309) instruction to produce a first in-first out (FIFO) queue. It moves the bit pattern of the holding register at the bottom of a full queue to a destination register or to word that stores 16 discrete outputs.



**Note:** The FOUT instruction should be placed before the FIN instruction in the ladder logic FIFO to ensure removal of the oldest data from a full queue before the newest data is entered. If the FIN block were to appear first, any attempts to enter the new data into a full queue would be ignored.

### Source Pointer (Top Node)

In the FOUT instruction, the source data comes from the 4x register at the bottom of a full queue. The next contiguous 4x register following the source pointer register in the top node is the first register in the queue. For example, if the top node displays pointer register 400100, then the first register in the queue is 400101. The value posted in the source pointer equals the number of registers in the queue that are currently filled. The value of the pointer cannot exceed the integer maximum queue length value specified in the bottom node. If the value in the source pointer equals the integer specified in the bottom node, the middle output passes power and no further FIN data can be written to the queue until the FOUT instruction clears the register at the bottom of the queue to the destination register.

### Destination Register (Middle Node)

The destination specified in the middle node can be a 0x reference or 4x register. When the queue has data and the top input to the FOUT passes power, the source data is cleared from the bottom register in the queue and is written to the destination register.

---

## FTOI: Floating Point to Integer

70

---

### At a Glance

#### Introduction

This chapter describes the instruction FTOI.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	318
Representation	318

## Short Description

---

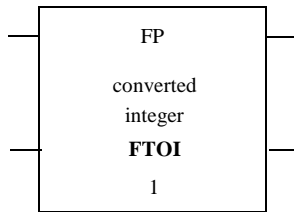
**Function Description** The FTOI instruction performs the conversion of a floating value to a signed or unsigned integer (stored in two contiguous registers in the top node), then stores the converted integer value in a 4x register in the middle node.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables conversion
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
FP (top node)	4x	REAL	First of two contiguous holding registers where the floating point value is stored
converted integer (middle node)	4x	INT, UINT	Converted integer value is posted here
1 (bottom node)		INT, UINT	A constant value of 1 (can not be changed)
Top output	0x	None	ON = integer conversion completed successfully
Bottom output	0x	None	ON = converted integer value is out of range: unsigned integer > 65 535 -32 768 > signed integer > 32 767

---

---

## HLTH: History and Status Matrices

71

---

### At a Glance

#### Introduction

This chapter describes the instruction HLTH.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	320
Representation	321
Parameter Description	322
Parameter Description Top Node (History Matrix)	323
Parameter Description Middle Node (Status Matrix)	328
Parameter Description Bottom Node (Length)	332

---

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information you will find in "*Installation of DX Loadables, p. 41*".

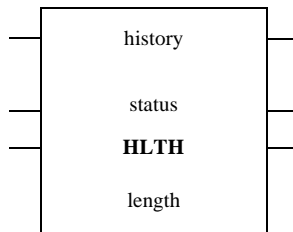
The HLTH instruction creates history and status matrices from internal memory registers that may be used in ladder logic to detect changes in PLC status and communication capabilities with the I/O. It can also be used to alert the user to changes in a PLC System. HLTH has two modes of operation, learn and monitor.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON initiates the designated operation
Middle input	0x, 1x	None	Learn / monitor mode
Bottom input	0x, 1x	None	Learn / monitor mode
history (top node)	4x	INT, UINT, WORD	History matrix (first in a block of contiguous registers, range: 6 ... 135)
status (middle node)	4x	INT, UINT, WORD	Status matrix (first in a block of contiguous registers, range: 3 ... 132)
length (bottom node)		INT, UINT	Number of I/O drops to manage
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Echoes state of the middle input
Bottom output	0x	None	ON = Error

## Parameter Description

### Modes of operation

The HLTH instruction has two modes of operation:

Type of Mode	Meaning
Learn Mode	<p>HLTH can be initialized to learn the configuration in which it is implemented and save the information as a point-in-time reference called <i>History Matrix (Top Node)</i>, p. 323</p> <p>This matrix contains:</p> <ul style="list-style-type: none"> <li>● A user-designated drop number for communications status monitoring</li> <li>● User logic checksum</li> <li>● Disabled I/O indicator</li> <li>● S911 Health</li> <li>● Choice of single or dual cable system</li> <li>● I/O Map display</li> </ul>
Monitor Mode	<p>Monitor mode enables an operation that checks PLC system conditions. Detected changes are recorded in a <i>Status Matrix (Middle Node)</i>, p. 328. The status matrix monitors the most recent system conditions and sets bit patterns to indicate detected changes.</p> <p>The status matrix contains:</p> <ul style="list-style-type: none"> <li>● Communication status of the drop designated in the history matrix</li> <li>● A flag to indicate when there is any disabled I/O</li> <li>● Flags to indicate the "on/off" status of constant sweep and the Memory protect key switch</li> <li>● Flags to indicate a battery-low condition and if Hot Standby is functional</li> <li>● Failed module position data</li> <li>● Changed user logic checksum flag</li> <li>● RIO lost-communication flag</li> </ul>

### Learn / Monitor Mode (Middle and Bottom Input)

The HLTH instruction block has three control inputs and can produce three possible outputs.

The combined states of the middle and bottom inputs control the operating mode:

Middle Input	Bottom Input	Operation
ON	OFF	Learn Mode as Dual Cable System
ON	ON	Learn Mode as Single Cable System
OFF	ON	Monitor Mode
OFF	OFF	Monitor Mode Update Logic Checksum



**Parameter Description Top Node (History Matrix)**

**History Matrix (Top Node)**

The 4x register entered in the top node is the first in a block of contiguous registers that comprise the history matrix. The data for the history matrix is gathered by the instruction during a learn mode operation and is set in the matrix when the mode changes to monitor.

The history matrix can range from 6 ... 135 registers in length. Below is a description of the words in the history matrix. The information from word 1 is contained in the displayed register in the top node and the information from words 2 ... 135 is stored in the implied registers.

**Word 1** Enter drop number (range 0 ... 32) to be monitored for retries

**Word 2** High word of learned checksum

**Word 3** Low word of learned checksum

**Word 4** The status and a counter for multiplexing the inputs. HLTH processes 16 words of input (256 inputs) per scan. This word holds the last word location of the last scan. The register is overwritten on every scan. The value in the counter portion of the word increases to the maximum number of inputs, then restarts at 0.

Usage of word 4:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = at least one disabled input has been found
2 - 16	Count of the number of word checked for disabled inputs prior to this scan.

**Word 5**

Status and a counter for multiplexing outputs to detect if one is disabled. HLTH looks at 16 words (256 outputs) per scan to find one that is disabled. It holds the last word location of the last scan. The block is overwritten on every scan. The value in the counter portion increases to maximum outputs then restarts at 0.

Usage of word 5:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = at least one disabled output has been found.
2 - 16	Count of the number of word checked for disabled outputs prior to this scan.

---

**Word 6**

Hot Standby cable learned data

Usage of word 6:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = S911 present during learn.
2 - 8	Not used
9	1 = cable A is monitored.
10	1 = cable B is monitored.
11 - 16	Not used

---

**Word 7 ... 134**

These words define the learned condition of drop 1 to drop 32 as follows:

Word	Drop No.
7 ... 10	1
11 ... 14	2
15 ... 18	3
:	:
:	:
131 ... 134	32

The structure of the four words allocated to each drop are as follows:

**First Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Drop delay bit 1 <b>Note:</b> Drop delay bits are used by the software to delay the monitoring of the drop for four scans after reestablishing communications with a drop. The delay value is for internal use only and needs no user intervention.
2	Drop delay bit 2
3	Drop delay bit 3
4	Drop delay bit 4
5	Drop delay bit 5
6	Rack 1, slot 1, module found
7	Rack 1, slot 2, module found
8	Rack 1, slot 3, module found
9	Rack 1, slot 4, module found
10	Rack 1, slot 5, module found
11	Rack 1, slot 6, module found
12	Rack 1, slot 7, module found
13	Rack 1, slot 8, module found
14	Rack 1, slot 9, module found
15	Rack 1, slot 10, module found
16	Rack 1, slot 11, module found

**Second Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 2, slot 1, module found
2	Rack 2, slot 2, module found
3	Rack 2, slot 3, module found
4	Rack 2, slot 4, module found
5	Rack 2, slot 5, module found
6	Rack 2, slot 6, module found
7	Rack 2, slot 7, module found
8	Rack 2, slot 8, module found
9	Rack 2, slot 9, module found
10	Rack 2, slot 10, module found
11	Rack 2, slot 11, module found
12	Rack 3, slot 1, module found
13	Rack 3, slot 2, module found
14	Rack 3, slot 3, module found
15	Rack 3, slot 4, module found
16	Rack 3, slot 5, module found

**Third Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 3, slot 6, module found
2	Rack 3, slot 7, module found
3	Rack 3, slot 8, module found
4	Rack 3, slot 9, module found
5	Rack 3, slot 10, module found
6	Rack 3, slot 11, module found
7	Rack 4, slot 1, module found
8	Rack 4, slot 2, module found
9	Rack 4, slot 3, module found
10	Rack 4, slot 4, module found
11	Rack 4, slot 5, module found

Bit	Function
12	Rack 4, slot 6, module found
13	Rack 4, slot 7, module found
14	Rack 4, slot 8, module found
15	Rack 4, slot 9, module found
16	Rack 4, slot 10, module found

**Fourth Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 4, slot 11, module found
2	Rack 5, slot 1, module found
3	Rack 5, slot 2, module found
4	Rack 5, slot 3, module found
5	Rack 5, slot 4, module found
6	Rack 5, slot 5, module found
7	Rack 5, slot 6, module found
8	Rack 5, slot 7, module found
9	Rack 5, slot 8, module found
10	Rack 5, slot 9, module found
11	Rack 5, slot 10, module found
12	Rack 5, slot 11, module found
13 ... 16	not used

## Parameter Description Middle Node (Status Matrix)

---

### Status Matrix (Middle Node)

The 4x register entered in the middle node is the first in a block of contiguous holding registers that will comprise the status matrix. The status matrix is updated by the HLTH instruction during monitor mode (top input is ON and middle input is OFF). The status matrix can range from 3 ... 132 registers in length. Below is a description of the words in the status matrix. The information from word 1 is contained in the displayed register in the middle node and the information from words 2 ... 131 is stored in the implied registers.

---

### Word 1

This word is a counter for lost-communications at the drop being monitored. Usage of word 1:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 8	Indicates the number of the drop being monitored (0 ... 32).
9 - 16	Count of the lost communication incidents (0 ... 15).

---

### Word 2

This word is the cumulative retry counter for the drop being monitored (the drop number is indicated in the high byte of word 1). Usage of word 2:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 4	Not used
5 - 16	Cumulative retry count (0 ... 255).

---

**Word 3**

This word updates PLC status (including Hot Standby health) on every scan.  
Usage of word 3:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	ON = all drops are not communicating.
2	Not used
3	ON = logic checksum has changed since last learn.
4	ON = at least one disabled 1x input detected.
5	ON = at least one disabled 0x output detected.
6	ON = constant sweep enabled.
7 - 10	Not used
11	ON = memory protect is OFF.
12	ON = battery is bad.
13	ON = an S911 is bad.
14	ON = Hot Standby not active.
15 - 16	Not used

**Word 4 ... 131**

These words indicate the status of drop 1 to drop 32 as follows:

Word	Drop No.
4 ... 7	1
8 ... 11	2
12 ... 15	3
:	:
:	:
128 ... 131	32

The structure of the four words allocated to each drop is as follows:

**First Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Drop communication fault detected
2	Rack 1, slot 1, module fault
3	Rack 1, slot 2, module fault
4	Rack 1, slot 3, module fault
5	Rack 1, slot 4, module fault
6	Rack 1, slot 5, module fault
7	Rack 1, slot 6, module fault
8	Rack 1, slot 7, module fault
9	Rack 1, slot 8, module fault
10	Rack 1, slot 9, module fault
11	Rack 1, slot 10, module fault
12	Rack 1, slot 11, module fault
13	Rack 2, slot 1, module fault
14	Rack 2, slot 2, module fault
15	Rack 2, slot 3, module fault
16	Rack 2, slot 4, module fault



**Second Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 2, slot 5, module fault
2	Rack 2, slot 6, module fault
3	Rack 2, slot 7, module fault
4	Rack 2, slot 8, module fault
5	Rack 2, slot 9, module fault
6	Rack 2, slot 10, module fault
7	Rack 2, slot 11, module fault
8	Rack 3, slot 1, module fault
9	Rack 3, slot 2, module fault
10	Rack 3, slot 3, module fault
11	Rack 3, slot 4, module fault
12	Rack 3, slot 5, module fault
13	Rack 3, slot 6, module fault
14	Rack 3, slot 7, module fault
15	Rack 3, slot 8, module fault
16	Rack 3, slot 9, module fault

**Third Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 3, slot 10, module fault
2	Rack 3, slot 11, module fault
3	Rack 4, slot 1, module fault
4	Rack 4, slot 2, module fault
5	Rack 4, slot 3, module fault
6	Rack 4, slot 4, module fault
7	Rack 4, slot 5, module fault
8	Rack 4, slot 6, module fault
9	Rack 4, slot 7, module fault
10	Rack 4, slot 8, module fault
11	Rack 4, slot 9, module fault

Bit	Function
12	Rack 4, slot 10, module fault
13	Rack 4, slot 11, module fault
14	Rack 5, slot 1, module fault
15	Rack 5, slot 2, module fault
16	Rack 5, slot 3, module fault

**Fourth Word**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 5, slot 4, module fault
2	Rack 5, slot 5, module fault
3	Rack 5, slot 6, module fault
4	Rack 5, slot 7, module fault
5	Rack 5, slot 8, module fault
6	Rack 5, slot 9, module fault
7	Rack 5, slot 10, module fault
8	Rack 5, slot 11, module fault
9	Cable A fault
10	Cable B fault
11 ... 16	not used

---

**Parameter Description Bottom Node (Length)**

---

**Length (Bottom Node)**

The decimal value entered in the bottom node is a function of how many I/O drops you want to monitor. Each drop requires four registers/matrix. The length value is calculated using the following formula:

$$\text{length} = (\# \text{ of I/O drops} \times 4) + 3$$

This value gives you the number of registers in the status matrix. You only need to enter this one value as the length because the length of the history matrix is automatically increased by 3 registers -i.e., the size of the history matrix is length + 3.

---

---

## IBKR: Indirect Block Read

72

---

### At a Glance

#### Introduction

This chapter describes the instruction IBKR.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	334
Representation	334

---

## Short Description

---

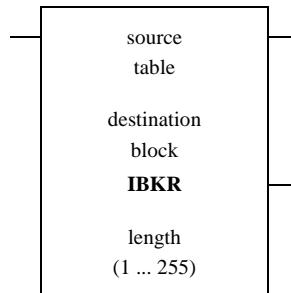
**Function Description** The IBKR (indirect block read) instruction lets you access non-contiguous registers dispersed throughout your application and copy the contents into a destination block of contiguous registers. This instruction can be used with subroutines or for streamlining data access by host computers or other PLCs.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates indirect read operation
source table (top node)	4x	INT, UINT	First holding register in a source table: contain values that are pointers to the non-contiguous registers you want to collect in the operation.
destination block (middle node)	4x	INT, UINT	First in a block of contiguous destination registers, i.e. the block to which the source data will be copied.
length (1 ... 255) (bottom node)		INT, UINT	number of registers in the source table and the destination block, range: 1 ... 255
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = error in source table

---

---

## IBKW: Indirect Block Write

73

---

### At a Glance

#### Introduction

This chapter describes the instruction IBKW.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	336
Representation	336

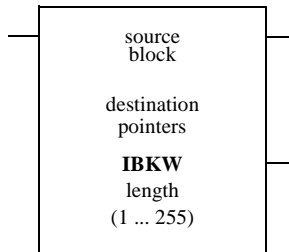
---

## Short Description

**Function Description** The IBKW (indirect block write) instruction lets you copy the data from a table of contiguous registers into several non-contiguous registers dispersed throughout your application.

## Representation

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates indirect write operation
source block (top node)	4x	INT, UINT	First in a block of source registers: contain values that will be copied to non-contiguous registers dispersed throughout the logic program
destination pointers (middle node)	4x	INT, UINT	First in a block of contiguous destination pointer registers. Each of these registers contains a value that points to the address of a register where the source data will be copied.
length (1 ... 255) (bottom node)		INT, UINT	Number of registers in the source block and the destination pointer block, range: 1 ... 255
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = error in destination table

---

## ICMP: Input Compare

74

---

### At a Glance

#### Introduction

This chapter describes the instruction ICMP.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	338
Representation	339
Parameter Description	340
Cascaded DRUM/ICMP Blocks	341

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information you will find in "*Installation of DX Loadables, p. 41*".

The ICMP (input compare) instruction provides logic for verifying the correct operation of each step processed by a DRUM instruction. Errors detected by ICMP may be used to trigger additional error-correction logic or to shut down the system.

ICMP and DRUM are synchronized through the use of a common step pointer register. As the pointer increments, ICMP moves through its data table in lock step with DRUM. As ICMP moves through each new step, it compares-bit for bit-the live input data to the expected status of each point in its data table.

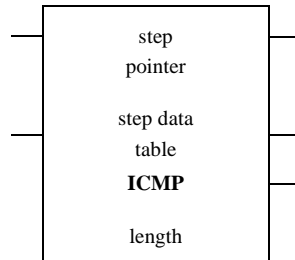
---



## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates the input comparison
Middle input	0x, 1x	None	A cascading input, telling the block that previous ICMP comparison were all good, ON = compare status is passing to the middle output
step pointer (top node)	4x	INT, UINT	Current step number
step data table (middle node)	4x	INT, UINT	First register in a table of step data information
length (bottom node)		INT, UINT	Number of application-specific registers-used in the step data table, range: 1 .. 999
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON =this comparison and all previous cascaded ICMPs are good
Bottom output	0x	None	ON = Error

## Parameter Description

### Step Pointer (Top Node)

The 4x register entered in the top node stores the step pointer, i.e., the number of the current step in the step data table. This value is referenced by ICMP each time the instruction is solved. The value must be controlled externally by a DRUM instruction or by other user logic. The same register must be used in the top node of all ICMP and DRUM instructions that are solved as a single sequencer.

### Step Data Table (Middle Node)

The 4x register entered in the middle node is the first register in a table of step data information. The first eight registers in the table hold constant and variable data required to solve the instruction:

Register	Name	Content
Displayed	raw input data	Loaded by user from a group of sequential inputs to be used by ICMP for current step
First implied	current step data	Loaded by ICMP each time the block is solved; contains a copy of data in the step pointer; causes the block logic to automatically calculate register offsets when accessing step data in the step data table
Second implied	input mask	Loaded by user before using the block; contains a mask to be ANDed with raw input data for each step-masked bits will not be compared; masked data are put in the masked input data register
Third implied	masked input data	Loaded by ICMP each time the block is solved; contains the result of the ANDed input mask and raw input data
Fourth implied	compare status	Loaded by ICMP each time the block is solved; contains the result of an XOR of the masked input data and the current step data; unmasked inputs that are not in the correct logical state cause the associated register bit to go to 1-non-zero bits cause a miscompare, and middle output will not go ON
Fifth implied	machine ID number	Identifies DRUM/ICMP blocks belonging to a specific machine configuration; value range: 0 ... 9999 (0 = block not configured); all blocks belonging to same machine configuration have the same machine ID

Register	Name	Content
Sixth implied	Profile ID Number	Identifies profile data currently loaded to the sequencer; value range: 0... 9999 (0 = block not configured); all blocks with the same machine ID number must have the same profile ID number
Seventh implied	Steps used	Loaded by user before using the block, DRUM will not alter steps used contents during logic solve: contains between 1 ... 999 for 24 bit CPUs, specifying the actual number of steps to be solved; the number must be $\leq$ the table length in the bottom node of the ICMP block

The remaining registers contain data for each step in the sequence.

#### Length (Bottom Node)

The integer value entered in the bottom node is the length-i.e., the number of application-specific registers-used in the step data table. The length can range from 1 .. 999 in a 24-bit CPU.

The total number of registers required in the step data table is the length + 8. The length must be  $>$  the value placed in the steps used register in the middle node.

### Cascaded DRUM/ICMP Blocks

#### Cascaded DRUM/ICMP Blocks

A series of DRUM and/or ICMP blocks may be cascaded to simulate a mechanical drum up to 512 bits wide. Programming the same 4x register reference into the top node of each related block causes them to cascade and step as a grouped unit without the need of any additional application logic.

All DRUM/ICMP blocks with the same register reference in the top node are automatically synchronized. They must also have the same constant value in the bottom node, and must be set to use the same value in the steps used register in the middle node.



---

## ID: Interrupt Disable

75

---

### At a Glance

#### Introduction

This chapter describes the instruction ID.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	344
Representation	344
Parameter Description	345

## Short Description

---

### Function Description

**Note:** This instruction is only available after configuring a CPU without extension.

Three interrupt mask/unmask control instructions are available to help protect data in both the normal (scheduled) ladder logic and the (unscheduled) interrupt handling subroutine logic. These are the Interrupt Disable (ID) instruction, the Interrupt Enable (IE) instruction, and the Block Move with Interrupts Disabled (BMDI) instruction.

The ID instruction masks timer-generated and/or local I/O-generated interrupts. An interrupt that is executed in the timeframe after an ID instruction has been solved and before the next IE instruction has been solved is buffered. The execution of a buffered interrupt takes place at the time the IE instruction is solved. If two or more interrupts of the same type occur between the ID ... IE solve, the mask interrupt overrun error bit is set, and the subroutine initiated by the interrupts is executed only one time

Further Information you will find in the chapter *Interrupt Handling*, p. 37.

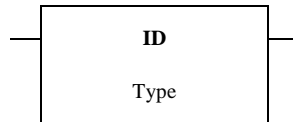
---

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = instruction masks timer-generated and/or local I/O generated interrupts
Type bottom node		INT, UINT	Type of interrupt to be masked (Constant integer)
Top output	0x	None	Echoes state of the top input

---

**Parameter Description**

---

**Type (Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the node. The value represents the type of interrupt to be masked by the ID instruction, where:

Integer Value	Interrupt Type
3	Timer interrupt masked
2	Local I/O module interrupt masked
1	Both interrupt types masked

---

ID: Interrupt Disable

---



---

## IE: Interrupt Enable

76

---

### At a Glance

#### Introduction

This chapter describes the instruction IE.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	348
Representation	348
Parameter Description	349

## Short Description

### Function Description

**Note:** This instruction is only available after configuring a CPU without extension.

Three interrupt mask/unmask control instructions are available to help protect data in both the normal (scheduled) ladder logic and the (unscheduled) interrupt handling subroutine logic. These are the Interrupt Disable (ID) instruction, the Interrupt Enable (IE) instruction, and the Block Move with Interrupts Disabled (BMDI) instruction.

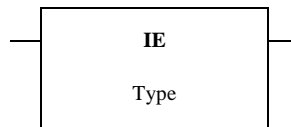
The IE instruction unmask interrupts from the timer or local I/O module and responds to the pending interrupts by executing the designated subroutines. An interrupt that is executed in the timeframe after an ID instruction has been solved and before the next IE instruction has been solved is buffered. The execution of a buffered interrupt takes place at the time the IE instruction is solved. If two or more interrupts of the same type occur between the ID ... IE solve, the mask interrupt overrun error bit is set, and the subroutine initiated by the interrupts is executed only one time.

Further Information you will find in the chapter *Interrupt Handling*, p. 37.

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = instruction unmask interrupts and responds pending interrupts
Type bottom node		INT, UINT	Type of interrupt to be unmasked (Constant integer)
Top output	0x	None	Echoes state of the top input

**Parameter Description**

---

**Top Input**

When the input is energized, the IE instruction unmask interrupts from the timer or local I/O module and responds to the pending interrupts by executing the designated subroutines.

---

**Type (Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the node. The value represents the type of interrupt to be unmasked by the IE instruction, where:

Integer Value	Interrupt Type
3	Timer interrupt unmasked
2	Local I/O module interrupt unmasked
1	Both interrupt types unmasked

---

IE: Interrupt Enable

---

---

## IMIO: Immediate I/O

77

---

### At a Glance

**Introduction** This chapter describes the instruction IMIO.

**Note:** This instruction is only available after configuring a CPU without extension.

**What's in this chapter?**

This chapter contains the following topics:

Topic	Page
Short Description	352
Representation	352
Parameter Description	353
Run Time Error Handling	355

## Short Description

---

### Function Description

**Note:** This instruction is only available after configuring a CPU without extension.

The IMIO instruction permits access of specified I/O modules from within ladder logic. This differs from normal I/O processing, where inputs are accessed at the beginning of the logic solve for the segment in which they are used and outputs are updated at the end of the segment's solution. The I/O modules being accessed must reside in the local backplane with the Quantum PLC.

In order to use IMIO instructions, the local I/O modules to be accessed must be designated in the I/O Map in your panel software.

Further Information you will find in the chapter *Interrupt Handling*, p. 37.

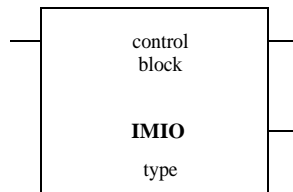
---

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables the immediate I/O access
control block top node	4x	INT, UINT, WORD	Control block (first of two contiguous registers)
type bottom node		INT, UINT	Type of operation (constant integer in the range of 1 ... 3)
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error (indicated by a code in the error status register (See <i>Runtime Errors</i> , p. 355) in the IMIO control block)

## Parameter Description

### Control Block (Top Node)

The first of two contiguous 4x registers is entered in the top node. The second register is implied.

Register	Content
Displayed	This register specifies the <i>Physical Address of the I/O Module, p. 353</i> to be accessed.
First implied	This register logs the error status (See <i>Runtime Errors, p. 355</i> ), which is maintained by the instruction.

### Physical Address of the I/ O Module

The high byte of the displayed register in the control block allows you to specify which rack the I/O module to be accessed resides in, and the low byte allow you to specify slot number within the specified rack where the I/O module resides.  
Usage of word:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 5	Not used
6 - 8	Rack number 1 to 4 (only rack 1 is currently supported)
9 - 11	Not used
12 - 16	Slot number

#### Rack Number

Bit Number			Rack Number
6	7	8	
0	0	1	rack 1
0	1	0	rack 2
0	1	1	rack 3
1	0	0	rack 4

**Slot Number**

Bit Number					Slot Number
12	13	14	15	16	
0	0	0	0	1	slot 1
0	0	0	1	0	slot 2
0	0	0	1	1	slot 3
0	0	1	0	0	slot 4
0	0	1	0	1	slot 5
0	0	1	1	0	slot 6
0	0	1	1	1	slot 7
0	1	0	0	0	slot 8
0	1	0	0	1	slot 9
0	1	0	1	0	slot 10
0	1	0	1	1	slot 11
0	1	1	0	0	slot 12
0	1	1	0	1	slot 13
0	1	1	1	0	slot 14
0	1	1	1	1	slot 15
1	0	0	0	0	slot 16

**Type (Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the bottom node. The value represents the type of operation to be performed by the IMIO instruction, where:

Integer Value	Type of Immediate Access
1	Input operation: transfers data from the specified module to state RAM
2	Output operation: transfers data from state RAM to the specified module
3	I/O operation: does both input and output if the specified module is bidirectional



---

## Run Time Error Handling

---

### Runtime Errors

The implied register in the control block will contain the following error code when the instruction detects an error:

Error Code	Meaning
2001	Invalid type specified in the bottom node
2002	Problem with the specified I/O slot, either an invalid slot number entered in the displayed register of the control block or the I/O Map does not contain the correct module definition for this slot
2003	A type 3 operation is specified in the bottom node, and the module is not bidirectional
F001	Specified I/O module is not healthy

---

IMIO: Immediate I/O

---

---

## IMOD: Interrupt Module Instruction

78

---

### At a Glance

#### Introduction

This chapter describes the instruction IMOD.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	358
Representation	359
Parameter Description	360

---

## Short Description

---

### Function Description

**Note:** This instruction is only available after configuring a CPU without extension.

The IMOD instruction initiates a ladder logic interrupt handler subroutine when the appropriate interrupt is generated by a local interrupt module and received by the PLC. Each IMOD instruction in an application is set up to correspond to a specific slot in the local backplane where the interrupt module resides. The IMOD instruction can designate the same or a separate interrupt handler subroutine for each interrupt point on the associated interrupt module.

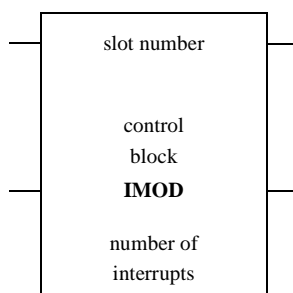
Further Information you will find in the chapter *Interrupt Handling*, p. 37.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates an interrupt
Bottom input	0x, 1x	None	ON = clears a previously detected error
slot number (top node)		INT, UINT	Indicates the slot number where the local interrupt module resides (constant integer in the range of 1 ... 16)
control block (middle node)	4x	INT, UINT, WORD	Control block (first of max. 19 contiguous registers, depending on number of interrupts)
number of interrupts (bottom node)		INT, UINT	Indicates the number of interrupts that can be generated from the associated interrupt module (constant integer in the range of 1 ... 16)
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	ON = error is detected. The source of the error can be from any one of the enabled interrupt points on the interrupt module.

## Parameter Description

---

### General Information to IMOD

Up to 14 IMOD instructions can be programmed in a ladder logic application, one for each possible option slot in a local backplane. Each interrupting point on each interrupt module can initiate a different interrupt handler subroutine. A maximum of 64 interrupt points can be defined in a user logic application. It is not necessary that all possible input points on a local interrupt module be defined in the IMOD instruction as interrupts.

---

### Enabling of the Instruction (Top Input)

When the input to the top node is energized, the IMOD instruction is enabled. The PLC will respond to interrupts generated by the local interrupt module in the designated slot number. When the top input is not energized, interrupts from the module in the designated slot are disabled and all previously detected errors are cleared including any pending masked interrupts.

---

### Clear Error (Bottom Input)

This input clears previous errors.

---

### Slot Number (Top Node)

The top node contains a decimal in the range 1 ... 16, indicating the slot number where the local interrupt module resides. This number is used to index into an array of control structures used to implement the instruction.

**Note:** The slot number in one IMOD instruction must be unique with respect to the slot numbers used in all other IMOD instruction in an application. If not the next IMOD with that particular slot number will have an error.

**Note:** The slot numbers where the PLC and the power supply reside are illegal entries -i.e., a maximum of 14 of the 16 possible slot numbers can be used as interrupt module slots. If the IMOD slot number is the same as the PLC, the IMOD will have an error.

---

**Control Block  
(Middle Node)**

The middle node contains the first 4x register in the IMOD control block. The control block contains parameters required to program an IMOD instruction. The size (number of registers) of the control block will equal the total number of programmed interrupt points + 3.

The first three registers in the control block contain status information, of the remaining registers provide means for you to specify the label (LAB) number of the *Subroutine Handling, p. 39* that is in the last (unscheduled) segment of the ladder logic program.

Control Block for IMOD

Register	Content
Displayed	Function status bits
First implied	State of inputs 1 ... 16 from the interrupt module at the time of the interrupt
Second implied	State of inputs 17 ... 32 from the interrupt module at the time of the interrupt (invalid data for a 16-bit interrupt module)
Third implied	LAB number and status for the first interrupt programmed point on the interrupt module
...	...
Last implied	LAB number and status for the last interrupt programmed point on the interrupt


**Function Status  
Bits**

Function Status Bits

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 2	Not used
3	Error: controller slot
4	Error: interrupt lost due to comm error in backplane
5	Module not healthy or not in I/O map
6	Error: interrupt lost because of on-line editing
7	Error: Maximum number of interrupts exceeded
8	Error: slot number used in previous network (see <b>CAUTION</b> <i>Lost of Interrupts, p. 362</i> )
9 - 15	Not used
16	0 = IMOD disabled 1 = IMOD enabled

**Lost of Interrupts**

	<b>CAUTION</b>
	<p><b>Lost of interrupts from the working IMOD instruction</b></p> <p>An error is indicated in bit 8 when two IMOD instructions are assigned the same slot number. When this happens, it is possible to lose interrupts from the working IMOD instruction without an indication if the number specified in the bottom node of the two instructions is different.</p> <p><b>Failure to observe this precaution can result in injury or equipment damage.</b></p>

**Status Bits and LAB Number for each Interrupt Point**

Bits 1 ... 5 of the third implied through last implied registers are status bits for each interrupt point. Bits 7 ... 16 are used to specify the LAB number for the interrupt handler subroutine. The LAB number is a decimal value in the range 1 ... 1023

Function Status Bits

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
<b>Interrupt Point Status</b>	
1	Execution delayed because of interrupt mask
2	Error: invalid block in the interrupt handler subroutine
3	Error: Mask interrupt overrun
4	Error: execution overrun
5	Error: invalid LAB number
6	not used
<b>LAB number</b>	
7 - 16	LAB number for the associated interrupt handler Value in the range 1 ... 1023

Whenever the input to the bottom node of the IMOD instruction is enabled, the status bits (bits 1 ... 5) are cleared. If a LAB number is specified (in bits 7 ... 16) as 0 or an invalid number, any interrupts generated from that point are ignored by the PLC.



**Number of  
Interrupts  
(Bottom Node)**

The bottom node contains an integer indicating the number of interrupts that can be generated from the associated interrupt module. The size (number of registers) of the control block is this number + 3.

The PLC is able to be configured for a maximum of 64 module interrupts (from all the interrupt modules residing in the local backplane). If the number you enter in the bottom node of an IMOD instruction causes the total number of module interrupts systemwide to exceed 64, an error is logged in bit 7 of the first register in the control block.

For example, if you use four interrupt modules in the local backplane and assign 16 interrupts to each of these modules (by entering 16 in the bottom node of each associated IMOD instruction, the PLC will not be able to handle any more module interrupts. If you attempt to create a fifth IMOD instruction, an error will be logged in that IMOD's control block when you specify a value in the bottom node.

---



---

## ITMR: Interrupt Timer

79

---

### At a Glance

#### Introduction

This chapter describes the instruction ITMR.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	366
Representation	367
Parameter Description	368

## Short Description

---

### Function Description

**Note:** This instruction is only available after configuring a CPU without extension.

The ITMR instruction allows you to define an interval timer that generates interrupts into the normal ladder logic scan and initiates the execution of an interrupt handling subroutine. The user-defined interrupt handler is a ladder logic subroutine created in the last, unscheduled segment of ladder logic with its first network marked by a LAB instruction. Subroutine execution is asynchronous to the normal scan cycle.

Up to 16 ITMR instructions can be programmed in an application. Each interval timer can be programmed to initiate the same or different interrupt handler subroutines, controlled by the *JSR / LAB Method, p. 40* described in the chapter *General*.

Each instance of the interval timer is delayed for a programmed interval while the PLC is running, then generates a processor interrupt when the interval has elapsed.

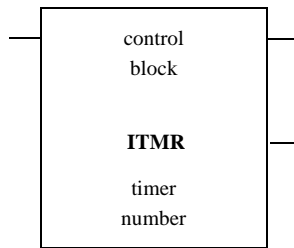
An interval timer can execute at any time during normal logic scan, including system I/O updating or other system housekeeping operations. The resolution of each interval timer is 1 ms. An interval can be programmed in units of 1 ms, 10 ms, 100 ms, or 1 s. An internal counter increments at the specified resolution. Further Information you will find in the chapter *Interrupt Handling, p. 37*.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables instruction
control block (top node)	4x	INT, UINT, WORD	Control block (first of three contiguous registers)
timer number (bottom node)		INT, UINT	Timer number assigned to this ITMR instruction (must be unique with respect to all other ITMR instructions in the application); range: 1 ... 16
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error (source of the error may be in the programmed parameters or a runtime execution error)

## Parameter Description

### Top Input

When the top input is energized, the ITRM instruction is enabled. It begins counting the programmed time interval. When that interval has expired the counter is reset and the designated error handler logic executes.

When the top input is not energized, the following events occur:

- All indicated errors are cleared
- The timer is stopped
- The time count is either reset or held, depending on the state of bit 15 of the first register in the control block (the displayed register in the top node)
- Any pending masked interrupt is cleared for this timer

### Control Block (Top Node)

The top node contains the first of three contiguous 4x registers in the ITMR control block. These registers are used to specify the parameters required to program each ITMR instruction.

Control Block for ITMR

Register	Content
Displayed	Function status and function control bits
First implied	In this register specify a value representing the interval at which the ITRM instruction will generate interrupts and initiate the execution of the interrupt handler. The interval will be incremented in the units specified by bits 12 and 13 of the first control block register, i.e. 1 ms, 10 ms, 100 ms, or 1 s units.
Second implied	In this register specify a value indicating the label (LAB) number that will start the interrupt handler subroutine. The number must be in the range 1 ... 1023.

**Note:** We recommend that the size of the logic subroutine associated with the LAB be minimized so that the application does not become interrupt-driven.

**Function Status and Function Control Bits**

The lower eight bits of the displayed register in the control block allow you to specify function control parameters, and the upper eight bits are used to display function status:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
<b>Function Status</b>	
1	Execution delayed because of interrupt mask.
2	Invalid block in the interrupt handler subroutine.
3	Not used
4	Time = 0
5	Mask interrupt overrun.
6	Execution overrun.
7	No LAB or invalid LAB.
8	Timer number used in previous network.
<b>Function Control</b>	
9 - 11	Not used
12 - 13	0 0 = 1 ms time base 0 1 = 10 ms time base 1 0 = 100 ms time base 1 1 = 1 s time base
14	1 = PLC stop holds counter. 0 = PLC stop resets counter.
15	1 = enable OFF holds counter. 0 = enable OFF resets counter.
16	1 = instruction enabled 0 = instruction disabled

**Timer Number (Bottom Node)**

Up to 16 ITRM instructions can be programmed in an application. The interrupts are distinguished from one another by a unique number between 1 ... 16, which you assign to each instruction in the bottom node. The lowest interrupt number has the highest execution priority.

For example, if ITMR 4 and ITMR 5 occur at the same time, ITMR 4 is executed first. After ITMR 4 has finished, ITMR 5 generally will begin executing.

An exception would be when another ITMR interrupt with a higher priority occurs during ITMR 4's execution. For example, suppose that ITMR 3 occurs while ITMR 5 is waiting for ITMR 4 to finish executing. In this case, ITMR 3 begins executing when ITMR4 finishes, and ITMR 5 continues to wait.





---

## ITOF: Integer to Floating Point

80

---

### At a Glance

#### Introduction

This chapter describes the instruction ITOF.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	372
Representation	372

## Short Description

---

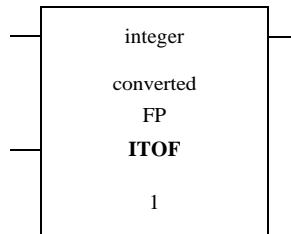
**Function Description** The ITOF instruction performs the conversion of a signed or unsigned integer value (its top node) to a floating point (FP) value, and stores the FP value in two contiguous 4x registers in the middle node.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables conversion
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
integer (top node)	3x, 4x	INT, UINT	Integer value, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
converted FP (middle node)	4x	REAL	Converted FP value (first of two contiguous holding registers)
1 (bottom node)		INT, UINT	Constant value of 1, can not be changed
Top output	0x	None	ON = FP conversion completed successfully

---

---

## JSR: Jump to Subroutine

81

---

### At a Glance

**Introduction** This chapter describes the instruction JSR.

**What's in this chapter?** This chapter contains the following topics:

Topic	Page
Short Description	374
Representation	374

## Short Description

---

### Function Description

When the logic scan encounters an enabled JSR instruction, it stops the normal logic scan and jumps to the specified source subroutine in the last (unscheduled) segment of ladder logic.

You can use a JSR instruction anywhere in user logic, even within the subroutine segment. The process of calling one subroutine from another subroutine is called nesting. The system allows you to nest up to 100 subroutines; however, we recommend that you use no more than three nesting levels. You may also perform a recursive form of nesting called looping, whereby a JSR call within the subroutine recalls the same subroutine.

---

### Example to Subroutine Handling

An example to subroutine handling you will find in the chapter *General*, section *Subroutine Handling*, p. 39.

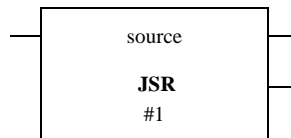
---

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Enables the source subroutine
source (top node)	4x	INT, UINT	Source pointer (indicator of the subroutine to which the logic scan will jump), entered explicitly as an integer or stored in a register; range: 1 ... 1 023
#1 (bottom node)		INT, UINT	Always enter the constant value 1
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error in subroutine jump

---

## LAB: Label for a Subroutine

82

---

### At a Glance

**Introduction** This chapter describes the instruction LAB.

**What's in this chapter?** This chapter contains the following topics:

Topic	Page
Short Description	376
Representation	376
Parameter Description	377

---

## Short Description

---

### Function Description

The LAB instruction is used to label the starting point of a subroutine in the last (unscheduled) segment of user logic. This instruction must be programmed in row 1, column 1 of a network in the last (unscheduled) segment of user logic. LAB is a one-node function block

LAB also serves as a default return from the subroutine in the preceding networks. If you are executing a series of subroutine networks and you find a network that begins with LAB, the system knows that the previous subroutine is finished, and it returns the logic scan to the node immediately following the most recently executed JSR block.

### Example to Subroutine Handling

An example to subroutine handling you will find in the chapter *General*, section *Subroutine Handling*, p. 39.

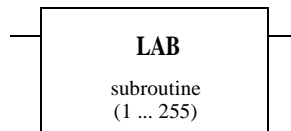
---

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Initiates the subroutine specified by the number in the bottom node
subroutine (bottom node)		INT, UINT	Integer value, identifies the subroutine you are about to execute, range: 1 ... 255
Top output	0x	None	ON = error in the specified subroutine's initiation

---

## Parameter Description

---

**Subroutine  
(Bottom Node)**

The integer value entered in the node identifies the subroutine you are about to execute. The value can range from 1 ... 255. If more than one subroutine network has the same LAB value, the network with the lowest number is used as the starting point for the subroutine.

---

LAB: Label for a Subroutine

---



---

## LOAD: Load Flash

83

---

### At a Glance

#### Introduction

This chapter describes the instruction LOAD.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	380
Representation	380
Parameter Description	381

## Short Description

### Function Description

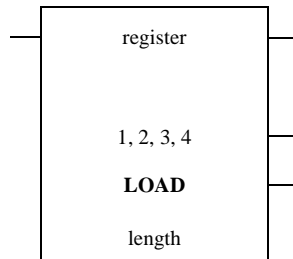
**Note:** This instruction is available with the PLC family TSX Compact, with Quantum CPUs 434 12/ 534 14 and Momentum CPUs CCC 960 x0/ 980 x0.

The LOAD instruction loads a block of 4x registers (previously SAVEd) from state RAM where they are protected from unauthorized modification.

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Start LOAD operation: it should remain ON until the operation has completed successfully or an error has occurred.
register (top node)	4x	INT, UINT, WORD	First of max. 512 contiguous 4x registers to be loaded from state RAM
1, 2, 3, 4 (middle node)		INT	Integer value, which defines the specific buffer where the block of data is to be loaded
length (bottom node)		INT	Number of words to be loaded, range: 1 ... 512
Top output	0x	None	ON = LOAD is active
Middle output	0x	None	ON = a LOAD is requested from a buffer where no data has been SAVEd.
Bottom output	0x	None	ON = Length not equal to SAVEd length

## Parameter Description

---

<b>1, 2, 3, 4 (Middle Node)</b>	The middle node defines the specific buffer where the block of data is to be loaded. Four 512 word buffers are allowed. Each buffer is defined by placing its corresponding value in the middle node, that is, the value 1 represents the first buffer, value 2 represents the second buffer and so on. The legal values are 1, 2, 3, and 4. When the PLC is started all four buffers are zeroed. Therefore, you may not load data from the same buffer without first saving it with the instruction SAVE. When this is attempted the middle output goes ON. In other words, once a buffer is used, it may not be used again until the data has been removed.
<b>Bottom Output</b>	The output from the bottom node goes ON when a LOAD request is not equal to the registers that were SAVEd. This kind of transaction is allowed, however, it is your responsibility to ensure this does not create a problem in your application.

---

LOAD: Load Flash

---

---

## MAP 3: MAP Transaction

84

---

### At a Glance

#### Introduction

This chapter describes the instruction MAP 3.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	384
Representation	384
Parameter Description	385

---

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

Ladder logic applications running in the controller initiate communication with MAP network nodes through the MAP3 instruction.

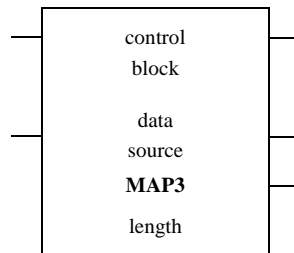
---

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates a transaction
Middle input	0x, 1x	None	ON = new transaction to be initiated in the same scan
control block (top node)	4x	INT, UINT, WORD	Control Block (first register of a block)
data source (middle node)	4x	INT, UINT, WORD	Data source (starting register)
length (bottom node)		INT, UINT	Length of local data area, range: 1 ... 255)
Top output	0x	None	Transaction completes successfully
Middle output	0x	None	Transaction is in progress
Bottom output	0x	None	Error

**Parameter Description**

---

**Top Input**

This input initiates a transaction. To start a transaction the input must be held ON (HIGH) for at least one scan. If the S980 has resources to process the transaction, the middle output passes power. If resources are not available, no outputs pass power.

Once a transaction is started, it will run until a reply is received, a communications error is detected, or a timeout occurs. The values in the Control Block, Data Source and Length must not be altered, or the transaction will not be completed and the bottom output will pass power. A second transaction cannot be started by the same block until the first one is complete.

---

**Middle Input**

If the top input is also HIGH, the middle input going ON allows a new transaction to be initiated in the same scan, following the completion of a previous one. A new transaction begins when the top output passes power from the first transaction.

---

**Control Block  
(Top Node)**

The top node is the starting 4x register of a block of registers that control the block's operation.

The contents of each register is determined by the kind of operation to be performed by the MAP3 block:

- Read or Write
- Information Report
- Unsolicited Status
- Conclude
- Abort

Registers of the Control Block:

Word	Meaning
1	<i>Destination Device, p. 386</i>
2	<i>Qualifier / Function Code, p. 387</i>
3	<i>Network Mode / Network Type, p. 387</i>
4	<i>Function Status, p. 388</i>
5	Register A Reference Type This word is labeled Register A* and contains the reference type for 4 types of Read (0x, 1x, 3x, and 4x registers) and 2 types of Write (0X or 4x).
6	Register B Reference Number This word is labeled Register B* and contains the starting reference number in the range 1 to 99999.
7	Register C Reference Length This word is labeled Register C* and contains the Quantity of references requested.
8	Register D Timeout This word is labeled Register D* and contains the Timeout parameter. This value sets the maximum length of time used to complete a transaction, including retries.

**Destination  
Device**

Word 1 contains the destination device in bit position 9 through 16. The computer works with this byte as the LSB and will accept a range of 1 to 255.

Usage of word 1:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 8	Not used
9 - 16	Destination device



**Qualifier /  
Function Code**

Word 2 contains two bytes of information. The qualifier bits 1 to 8 and the function code in bits 9 to 16.

Usage of word 2:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
<b>Qualifier</b>	
1 - 8	0 = addressed >0 = named
<b>Function Code</b>	
9 - 16	4 = read 5 = write

**Network Mode /  
Network Type**

Word 3 contains two bytes of information. The mode is in bits 5 through 8 and the type is in bits 9 through 16.

Usage of word 3:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 4	Not used
<b>Mode</b>	
5 - 8	1 = association
<b>Type</b>	
9 - 12	7 = 7 layer MAP network
13 - 16	1 = type 1 service

**Function Status** Word 4 is the function status. An error code is returned if an error occurs in a block initiated function.  
The decimal codes are:

Code	Meaning
1	Association request rejected
4	Message timeout application response
5	Invalid destination device
6	Message size exceeded
8	Invalid function code
17	Device not available
19	Unsupported network type
22	No channel available
23	MMS message not sent
24	Control block changed
25	Initiate failed
26	System download in progress
28	Channel not ready
99	Undetermined error
103	Access denied
105	Invalid address
110	Object nonexistent

---

**Function summary** The network controlling device may issue a function code that alters the control block register assignment as given above for Read/Write. Those differences for Information, Status, Conclude and Abort are identified in this summary on the bottom of your screen  
Refer to *Modicon S980 MAP 3.0 Network Interface User Guide* that describes the register contents for each operation.

---

**Data Source (Middle Node)** The middle node is the starting 4x register of the local data source (for a write request) or local data destination (for a read).

---

**Length (Bottom Node)** The bottom node defines the maximum size of the local data area (the quantity of registers) starting at 4x register of data source, in the range of 1 to 255 decimal. The quantity of data to be actually transferred in the operation is determined by a Reference Length parameter in one of the control registers.

---

<b>Top Output</b>	The top output passes power for one scan when a transaction completes successfully.
<b>Middle Output</b>	The middle output passes power when a transaction is in progress. If the top input is ON and the middle input is OFF, then the middle output will go OFF on the same scan that the top output goes ON. If both top input and middle input are ON, then the middle output will remain ON
<b>Bottom Output</b>	The bottom output passes power for one scan when a transaction cannot be completed. An error code is returned to the Function Status Word (register 4x+3) in the function's control block.

---

MAP 3: MAP Transaction

---

---

## MBIT: Modify Bit

85

---

### At a Glance

#### Introduction

This chapter describes the instruction MBIT.

#### What's in this chapter?

This chapter contains the following topics:


Topic	Page
Short Description	392
Representation	393
Parameter Description	394

## Short Description

---

**Function  
Description**

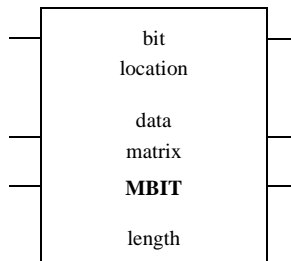
The MBIT instruction modifies bit locations within a data matrix, i.e. it sets the bit(s) to 1 or clears the bit(s) to 0. One bit location may be modified per scan.

	<b>WARNING</b>
	<b>Overriding of disabled coils without enabling them</b> MBIT will override any disabled coils within a destination group without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the MBIT instruction. <b>Failure to observe this precaution can result in severe injury or equipment damage.</b>

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = implements bit modification
Middle input	0x, 1x	None	OFF = clear bit locations to 0 ON = set bit locations to 1
Bottom input	0x, 1x	None	Increment bit location by one after modification
bit location (top node)	3x, 4x	INT, UINT, WORD	Specific bit location to be set or clear in the data matrix; entered explicitly as an integer value or stored in a register (range 1 ... 9 600)
data matrix (middle node)	0x, 4x	INT, UINT, WORD	First word or register in the data matrix
length (bottom node)		INT, UINT	Matrix length; range: 1 ... 600
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Echoes state of the middle input
Bottom output	0x	None	ON = error: bit location > matrix length

## Parameter Description

---

### Bit Location (Top Node)

**Note:** If the bit location is entered as an integer or in a 3x register, the instruction will ignore the state of the bottom input.

### Matrix Length (Bottom Node)

The integer value entered in the bottom node specifies a matrix length, i.e, the number of 16-bit words or registers in the data matrix. The length can range from 1 ... 600 in a 24-bit CPU, e.g, a matrix length of 200 indicates 3200 bit locations.

---



---

## MBUS: MBUS Transaction

86

---

### At a Glance

#### Introduction

This chapter describes the instruction MBUS.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	396
Representation	397
Parameter Description	398
The MBUS Get Statistics Function	400

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

The S975 Modbus II Interface option modules use two loadable function blocks: MBUS and PEER. MBUS is used to initiate a single transaction with another device on the Modbus II network. In an MBUS transaction, you are able to read or write discrete or register data.

PLCs on a Modbus II network can handle up to 16 transactions simultaneously. Transactions include incoming (unsolicited) messages as well as outgoing messages. Thus, the number of message initiations a PLC can manage at any time is 16 - # of incoming messages.

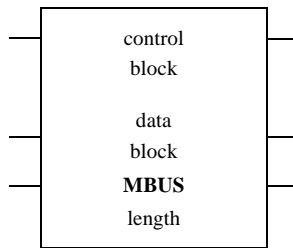
A transaction cannot be initiated unless the S975 has enough resources for the entire transaction to be performed. Once a transaction has been initiated, it runs until a reply is received, an error is detected, or a timeout occurs. A second transaction cannot be started in the same scan that the previous transaction completes unless the middle input is ON. A second transaction cannot be initiated by the same MBUS instruction until the first transaction has completed.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Enable MBUS transaction
Middle input	0x, 1x	None	Repeat transaction in same scan
Bottom input	0x, 1x	None	Clears system statistics
control block (top node)	4x	INT, UINT, WORD	First of seven contiguous registers in the MBUS control block
data block (middle node)	4x	INT, UINT, WORD	First 4x register in a data block to be transmitted or received in the MBUS transaction.
length (bottom node)		INT, UINT	Number of words reserved for the data block is entered as a constant value
Top output	0x	None	Transaction complete
Middle output	0x	None	Transaction in progress or new transaction starting
Bottom output	0x	None	Error detected in transaction

## Parameter Description

---

### Control Block (Top Node)

The 4x register entered in the top node is the first of seven contiguous registers in the MBUS control block:

Register	Content
Displayed	Address of destination device (range: 0 ... 246)
First implied	not used
Second implied	Function code
Third implied	Reference type
Fourth implied	Reference number, e.g., if you placed a 4 in the third implied register and you place a 23 in this register, the reference will be holding register 400023
Fifth implied	Number of words of discrete or register references to be read or written
Sixth implied	Time allowed for a transaction to be completed before an error is declared; expressed as a multiple of 10 ms, e.g., 100 indicates 1 000 ms; the default timeout is 250 ms.

---

### Function Code

This register contains the function code for requested action:

Value	Meaning
01	Read discrettes
02	Read registers
03	Write discrete outputs
04	Write register outputs
255	Get system statistics

---

### Reference Type

This register contains one of 4 possible discrete or register reference types:

Value	Reference type
0	Discrete output (0x)
1	Discrete input (1x)
2	Input register (3x)
3	Holding register (4x)

---

**Number of Words to Read or Write**

Number of words of discrete or register references to be read or written; the length limits are:

Read register	251 registers
Write register	249 registers
Read coils	7.848 discrettes
Write coils	7.800 discrettes

**Length (Bottom Node)**

The number of words reserved for the data block is entered as a constant value in the bottom node. This number does not imply a data transaction length, but it can restrict the maximum allowable number of register or discrete references to be read or written in the transaction.

The maximum number of words that may be used in the specified transaction is:

Max. Number of Words	Transaction
251	Reading registers (one register/word)
249	Writing registers (one register/word)
490	Reading discrettes using 24-bit CPUs (up to 16 discrettes/word)
487	Writing discrettes using 24-bit CPUs (up to 16 discrettes/word)

## The MBUS Get Statistics Function

---

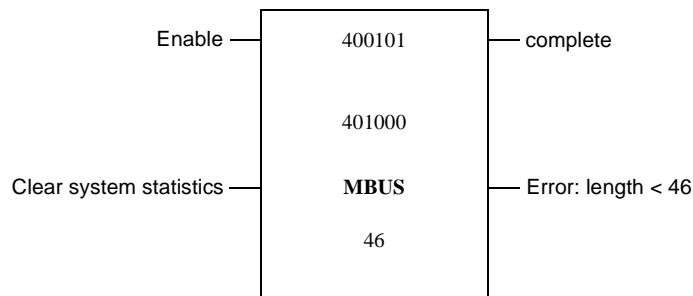
### General

Issuing function code 255 in the second implied register of the MBUS control block obtains a copy of the Modbus II local statistics, a series of 46 contiguous register locations where data describing error and system conditions is stored. To use MBUS for a get statistics operation, set the length in the bottom node to 46, a length < 46 returns an error (the bottom output will go ON), and a length > 46 reserves extra registers that cannot be used.

---

### Example

Parameterizing of the instruction



Register 400101 is the first register in the MBUS control block, making register 400103 the control register that defines the MBUS function code. By entering a value of 255 in register 400103, you implement a get statistics function. Registers 401000 ... 401045 are then filled with the system statistics.

---

### System Statistics Overview

The following system statistics are available:

- *Token Bus Controller (TBC), p. 401*
  - *Software-maintained Receive Statistics, p. 401*
  - *TBC-maintained Error Counters, p. 401*
  - *Software-maintained Transmit Errors, p. 402*
  - *Software-maintained Receive Errors, p. 402*
  - *User Logic Transaction Errors, p. 402*
  - *Manufacturing Message Format Standard, p. 402*
  - *(MMFS) Errors, p. 402*
  - *Background Statistics, p. 403*
  - *Software Revision, p. 403*
-

**Token Bus  
Controller (TBC)**

Registers 401000 ... 401003 are then filled with the following:

Register	Content
401000	Number of tokens passed by this station
401001	Number of tokens sent by this station
401002	Number of time the TBC has failed to pass token and has not found a successor
401003	Number of times the station has had to look for a new successor

**Software-  
maintained  
Receive  
Statistics**

Registers 401004 ... 401010 are then filled with the following:

Register	Content
401004	TBC-detected error frames
401005	Invalid request with response frames
401006	Applications message too long
401007	Media access control (MAC) address out of range
401008	Duplicate application frames
401009	Unsupported logical link control (LLC) message types
401010	Unsupported LLC address

**TBC-maintained  
Error Counters**

Registers 401011 ... 401018 are then filled with the following:

Register	Content
401011	Receive noise bursts (no start delimiter)
401012	Frame check sequence errors
401013	E-bit error in end delimiter
401014	Fragmented frames received (start delimiter not followed by end delimiter)
401015	Receive frames too long
401016	Discarded frames because there is no receive buffer
401017	Receive overruns
401018	Token pass failures

**Software-maintained Transmit Errors**

Registers 401019 ... 401020 are then filled with the following:

Register	Content
401019	Retries on request with response frames
401020	All retries performed and no response received from unit

---

**Software-maintained Receive Errors**

Registers 401021... 401022 are then filled with the following:

Register	Content
401021	Bad transmit request
401022	Negative transmit confirmation

---

**User Logic Transaction Errors**

Registers 401023... 401024 are then filled with the following:

Register	Content
401023	Message sent but no application response
401024	Invalid MBUS/PEER logic

---

**Manufacturing Message Format Standard**

Registers 401025... 401026 are then filled with the following:

Register	Content
401025	Command not executable
401026	Data not available

---

**(MMFS) Errors**

Registers 401027... 401035 are then filled with the following:

Register	Content
401027	Device not available
401028	Function not implemented
401029	Request not recognized
401030	Syntax error
401031	Unspecified error
401032	Data request out of bounds
401033	Request contains invalid controller address
401034	Request contains invalid data type
401035	None of the above

---



**Background  
Statistics**

Registers 401036... 401043 are then filled with the following:

Register	Content
401036	Invalid MBUS/PEER request
401037	Number of unsupported MMFS message types received
401038	Unexpected response or response received after timeout
401039	Duplicate application responses received
401040	Response from unspecified device
401041	Number of responses buffered to be processed (in the least significant byte); number of MBUS/PEER requests to be processed (in the most significant byte)
401042	Number of received requests to be processed (in the least significant byte); number of transactions in process (in the most significant byte)
401043	S975 scan time in 10 ms increments

**Software  
Revision**

Registers 401044... 401045 are then filled with the following:

Register	Content
401044	Version level of fixed software (PROMs): major version number in most significant byte; minor version number in least significant byte
401045	Version of loadable software (EEPROMs): major version number in most significant byte; minor version number in least significant byte



---

## MRTM: Multi-Register Transfer Module

87

---

### At a Glance

#### Introduction

This chapter describes the instruction MRTM.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	406
Representation	407
Parameter Description	408

---

## Short Description

---

### Function Description

**Note:** This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

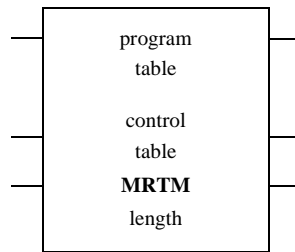
The MRTM instruction is used to transfer blocks of holding registers from the program table to the command block, a group of output registers. To verify each block transfer, an echo of the data contained in the first holding register is returned to an input register.

---

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables the operation
Middle input	0x, 1x	None	ON = one instruction block is transferred, table pointer of control table is incremented by the value of "length"
Bottom input	0x, 1x	None	ON =reset
program table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register of the program table. The digit 4 is assumed as the most significant digit
control table (middle node)	3x, 4x	INT, UINT, WORD	First register of the control table. The digit 4 is assumed as the most significant digit.
length (bottom node)		INT, UINT	Number of registers moved from the program table during each transfer, range: 1 to 127.
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Instruction block is transferred to the command block (stays on only for the remainder of the current scan)
Bottom output	0x	None	ON = pointer value $\geq$ table end

## Parameter Description

### Mode of Functioning

The MRTM transfers contiguous blocks of up to 127 registers from a table of register blocks to a block size holding register area. The MRTM function block controls the operation of the module in the following manner:

If power is applied to the...	Then ...
Top input	The function block is enabled for data transfers. <b>Note:</b> On initial startup, power must be applied to the bottom input.
Middle input	The function block attempts to transfer one instruction block. Before a transfer can occur, the echo register is evaluated. The most significant bit (MSB) of the echo register is not evaluated just bits 0 through 14. Echo mismatch is a condition that prohibits a transfer. If a transfer is permitted, one instruction block is transferred from the table starting at the table pointer. The table pointer in the control table is then advanced. If the pointer's new value is equal to or greater than the table end, the bottom output is turned on. A table pointer value less than the table end turns off the output.
Bottom input	The function block resets. The table pointer in the control table is reloaded with the start of commands value from the header of the program table

### Parameter Description Increment Step (Middle Input)

When power is applied, this input attempts to transfer one instruction block. Before a transfer can occur, the echo register is evaluated. The most significant bit (MSB) of the echo register is not evaluated, just bits 0 through 14. Echo mismatch is a condition that prohibits a transfer. If a transfer is permitted, one instruction block is transferred from the program table starting at the table pointer. The table pointer in the control table is then incremented by the value "Length" (displayed in the bottom node).

**Note:** The MRTM function block is designed to accept fault indications from I/O modules, which echo valid commands to the controller, but set a bit to indicate the occurrence of a fault. This method of fault indication is common for motion products and for most other I/O modules. If using a module that reports a fault condition in any other way, especially if the echo involved is not an echo of a valid command, special care must be taken when writing the error handler for the ladder logic to ensure the fault is detected. Failure to do so may result in a lockup or some other undesirable performance of the MRTM.

**Parameter  
Description  
Reset Pointer  
(Bottom Input)**

When power is applied to this input, the function block is reset. The table pointer in the control table is reloaded with the start of commands value from the header of the program table.

---





---

## MSTR: Master



88

---

### At a Glance

---

<b>Introduction</b>	This chapter describes the instruction MSTR.
---------------------	--

---

**What's in this chapter?**

This chapter contains the following topics:

<b>Topic</b>	<b>Page</b>
Short Description	413
Representation	414
Parameter Description	415
Write MSTR Operation	419
READ MSTR Operation	421
Get Local Statistics MSTR Operation	423
Clear Local Statistics MSTR Operation	425
Write Global Data MSTR Operation	427
Read Global Data MSTR Operation	428
Get Remote Statistics MSTR Operation	429
Clear Remote Statistics MSTR Operation	430
Peer Cop Health MSTR Operation	432
Reset Option Module MSTR Operation	434
Read CTE (Config Extension Table) MSTR Operation	435
Write CTE (Config Extension Table) MSTR Operation	437
Modbus Plus Network Statistics	440
TCP/IP Ethernet Statistics	445
Run Time Errors	446
Modbus Plus and SY/MAX Ethernet Error Codes	446
SY/MAX-specific Error Codes	448
TCP/IP Ethernet Error Codes	450
CTE Error Codes for SY/MAX and TCP/IP Ethernet	452

---

## Short Description

---

**Function  
Description**

PLCs that support networking communications capabilities over Modbus Plus and Ethernet have a special MSTR (master) instruction with which nodes on the network can initiate message transactions.

The MSTR instruction allows you to initiate one of 12 possible network communications operations over the network:

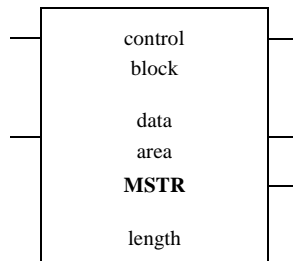
- Read MSTR Operation
  - Write MSTR Operation
  - Get Local Statistics MSTR Operation
  - Clear Local Statistics MSTR Operation
  - Write Global Data MSTR Operation
  - Read Global Data MSTR Operation
  - Get Remote Statistics MSTR Operation
  - Clear Remote Statistics MSTR Operation
  - Peer Cop Health MSTR Operation
  - Reset Option Module MSTR Operation
  - Read CTE (Config Extension) MSTR Operation
  - Write CTE (Config Extension) MSTR Operation
-

## Representation

---

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables selected MSTR operation
Middle input	0x, 1x	None	ON = terminates active MSTR operation
control block (top node)	4x	INT, UINT	Control block (first of several (network-dependant) contiguous holding registers)
data area (middle node)	4x	INT, UINT	Data area (source or destination depending on selected operation)
length (bottom node)		INT	Length of data area (maximum number of registers), range: 1 ... 100
Top output	0x	None	ON while the instruction is active (echoes state of the top input)
Middle output	0x	None	ON if the MSTR operation is terminated prior to completion (echoes state of the middle input)
Bottom output	0x	None	ON = operation successful

---

## Parameter Description

### Mode of Functioning

The MSTR instruction allows you to initiate one of 12 possible network communications operations over the network. Each operation is designated by a code.

Up to four MSTR instructions can be simultaneously active in a ladder logic program. More than four MSTRs may be programmed to be enabled by the logic flow; as one active MSTR block releases the resources it has been using and becomes deactivated, the next MSTR operation encountered in logic can be activated.

### Master Operations

Certain MSTR operations are supported on some networks and not on others:

Code	Type of Operation	Modbus Plus	TCP/IP Ethernet	SY/MAX Ethernet
1	<i>Write MSTR Operation, p. 419</i>	x	x	x
2	<i>READ MSTR Operation, p. 421</i>	x	x	x
3	<i>Get Local Statistics MSTR Operation, p. 423</i>	x	x	-
4	<i>Clear Local Statistics MSTR Operation, p. 425</i>	x	x	-
5	<i>Write Global Data MSTR Operation, p. 427</i>	x	-	-
6	<i>Read Global Data MSTR Operation, p. 428</i>	x	-	-
7	<i>Get Remote Statistics MSTR Operation, p. 429</i>	x	x	-
8	<i>Clear Remote Statistics MSTR Operation, p. 430</i>	x	x	-
9	<i>Peer Cop Health MSTR Operation, p. 432</i>	x	-	-
10	<i>Reset Option Module MSTR Operation, p. 434</i>	-	x	x
11	<i>Read CTE (Config Extension Table) MSTR Operation, p. 435</i>	-	x	x
12	<i>Write CTE (Config Extension Table) MSTR Operation, p. 437</i>	-	x	x

### Legend

x	supported
-	not supported

**Control Block (Top Node)**

The 4x register entered in the top node is the first of several (network-dependant) holding registers that comprise the network control block.

The control block structure differs according to the network in use:

- Modbus Plus (See *Control Block for Modbus Plus*, p. 416)
- TCP/IP Ethernet (See *Control Block for TCP/IP Ethernet*, p. 417)
- SY/MAX Ethernet (See *Control Block for SY/MAX Ethernet*, p. 418)

**Note:** You need to understand the routing procedures used by the network you are using when you program an MSTR instruction. A full discussion of Modbus Plus routing path structures is given in *Modbus Plus Network Planning and Installation Guide*. If TCP/IP or SY/MAX Ethernet routing is being implemented, it must be accomplished via standard third-party Ethernet IP router products.

**Control Block for Modbus Plus**

The first of twelve contiguous 4x registers is entered in the top node. The remaining eleven registers are implied:

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for Modbus Plus (1 ... 9)
First implied	Displays error status (See <i>Run Time Errors</i> , p. 446)
Second implied	Displays length (number of registers transferred)
Third implied	Displays MSTR operation-dependent information
Fourth implied	The Routing 1 register, used to designate the address of the destination node for a network transaction. The register display is implemented physically for the Quantum PLCs
Fifth implied	The Routing 2 register
Sixth implied	The Routing 3 register
Seventh implied	The Routing 4 register
Eighth implied	The Routing 5 register
Ninth implied	not applicable
Tenth implied	not applicable
Eleventh implied	not applicable

### Routing 1 Register for Quantum Automation Series PLCs (Fourth Implied Register)

To target a Modbus Plus Network Option module (NOM) in a Quantum PLC backplane as the destination of an MSTR instruction, the value in the high byte represents the physical slot location of the NOM, e.g. if the NOM resides in slot 7 in the backplane, the high byte of routing register 1 would look like this:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function								
1... 8	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td> </tr> </table> <p>High byte: indicating physical location (range 1 ... 16)</p>	0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1		
9 ... 16	<table border="1"> <tr> <td>0</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td> </tr> </table> <p>Destination address: binary value between 1 ... 64</p>	0	x	x	x	x	x	x	x
0	x	x	x	x	x	x	x		

**Note:** If you have created a logic program using an MSTR instruction for a 984 PLC and want to port it to a Quantum Automation Series PLC without having to edit the routing 1 register value, make sure that NOM #1 is installed in slot 1 of the Quantum backplane (and if a NOM #2 is used, that it is installed in slot 2 of the backplane). If you try to run the ported application with the NOMs in other slots without modifying the register, an F001 status error will appear, indicating the wrong destination node.

### Control Block for TCP/IP Ethernet

The first of nine contiguous 4x registers is entered in the top node. The remaining eight registers are implied:

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for TCP/IP (1 ... 4, 7, 8, 10 ... 12)
First implied	Displays error status (See <i>Run Time Errors</i> , p. 446)
Second implied	Displays length (number of registers transferred)
Third implied	Displays MSTR operation-dependent information
Fourth implied	Low byte: slot address of the NOE module High byte: MBP-to-Ethernet Transporter (MET) Map index
Fifth implied	Byte 4 of the 32-bit destination IP Address
Sixth implied	Byte 3 of the 32-bit destination IP Address
Seventh implied	Byte 2 of the 32-bit destination IP Address
Eighth implied	Byte 1 of the 32-bit destination IP Address

**Control Block for SY/MAX EthernetEthernet**

The first of seven contiguous 4x registers is entered in the top node. The remaining six registers are implied:

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for SY/MAX (1, 2, 10 ... 12)
First implied	Displays error status (See <i>Run Time Errors</i> , p. 446)
Second implied	Displays Read/Write length (number of registers transferred)
Third implied	Displays Read/Write base address
Fourth implied	Low byte: slot address of the NOE module (e.g., slot 10 = 0A00, slot 6 = 0600) High byte: MBP-to-Ethernet Transporter (MET) Map index
Fifth implied	Destination drop number (or set to FF hex)
Sixth implied	Terminator (set to FF hex)

---

**Data Area (Middle Node)**

The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the data area. For operations that provide the communication processor with data, such as a Write operation, the data area is the source of the data. For operations that acquire data from the communication processor, such as a Read operation, the data area is the destination for the data.

In the case of the Ethernet Read (See *Read CTE (Config Extension Table) MSTR Operation*, p. 435) and Write (See *Write CTE (Config Extension Table) MSTR Operation*, p. 437) CTE operations, the middle node stores the contents of the Ethernet configuration extension table in a series of registers.

---



## Write MSTR Operation

### Short Description

An MSTR Write operation transfers data from a master source device to a specified slave destination device on the network. Read and Write use one data master transaction path and may be completed over multiple scans. If you attempt to program the MSTR to Write its own station address, an error will be generated in the first implied register of the MSTR control block. It is possible to attempt a Write operation to a nonexistent register in the slave device. The slave will detect this condition and report it, this may take several scans.

### Network Implementation

The MSTR Write operation can be implemented on the Modbus Plus, TCP/IP Ethernet, and SY/MAX Ethernet networks.

### Control Block Utilization

In a Write operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using:

- Modbus Plus
- TCP/IP Ethernet
- SY/MAX Ethernet

### Control Block for Modbus Plus

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

**Control Block for TCP/IP Ethernet**

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error: <b>Exception code + 3000</b> : Exception response, where response size is correct <b>4001</b> : Exception response, where response size is incorrect <b>4001</b> : Read/Write
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... eighth implied	Destination	Each register contains one byte of the 32-bit IP address

---

**Control Block for SY/MAX Ethernet**

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fourth implied	Slot ID	High byte: Destination drop number
Fifth ... eighth implied	Terminator	FF hex

---

## READ MSTR Operation

### Short Description

An MSTR Read operation transfers data from a specified slave source device to a master destination device on the network. Read and Write use one data master transaction path and may be completed over multiple scans. If you attempt to program the MSTR to Read its own station address, an error will be generated in the first implied register of the MSTR control block. It is possible to attempt a Read operation to a nonexistent register in the slave device. The slave will detect this condition and report it, this may take several scans.

### Network Implementation

The MSTR Read operation can be implemented on the Modbus Plus, TCP/IP Ethernet, and SY/MAX Ethernet networks.

### Control Block Utilization

In a Read operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using:

- Modbus Plus
- TCP/IP Ethernet
- SY/MAX Ethernet

### Control Block for Modbus Plus

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from(1 = 40001, 49 = 40049)
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

**Control Block for TCP/IP Ethernet**

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error: <b>Exception code + 3000</b> : Exception response, where response size is correct <b>4001</b> : Exception response, where response size is incorrect <b>4001</b> : Read/Write
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from (1 = 40001, 49 = 40049)
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... eighth implied	Destination	Each register contains one byte of the 32-bit IP address

**Control Block for SY/MAX Ethernet**

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from (1 = 40001, 49 = 40049)
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fourth implied	Slot ID	High byte: Destination drop number
Fifth ... eighth implied	Terminator	FF hex

## Get Local Statistics MSTR Operation

### Short Description

The Get Local Statistics operation obtains information related to the local node, where the MSTR has been programmed. This operation takes one scan to complete and does not require a data master transaction path.

### Network Implementation

The Get Local Statistics operation (type 3 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available:

- *Modbus Plus Network Statistics, p. 440*
- *TCP/IP Ethernet Statistics, p. 445*

### Control Block Utilization

In a Get local statistics operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using:

- Modbus Plus
- TCP/IP Ethernet

### Control Block for Modbus Plus

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	3
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table (See <i>Modbus Plus Network Statistics</i> , p. 440); the length must be $> 0 \leq$ data area
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table; if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

**Control Block for  
TCP/IP Ethernet**

## Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	3
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table (See <i>TCP/IP Ethernet Statistics</i> , p. 445); the length must be $> 0 \leq \text{data area}$
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table, if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Slot ID	Low byte: Slot address of the network adapter module
Fifth ... Eighth implied	Not applicable	

## Clear Local Statistics MSTR Operation

### Short Description

The Clear local statistics operation clears statistics relative to the local node (where the MSTR has been programmed). This operation takes one scan to complete and does not require a data master transaction path.

**Note:** When you issue the Clear Local Statistics operation, only words 13 ... 22 in the statistics table (See *Modbus Plus Network Statistics*, p. 440) are cleared

### Network Implementation

The Clear Local Statistics operation (type 4 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available:

- *Modbus Plus Network Statistics*, p. 440
- *TCP/IP Ethernet Statistics*, p. 445

### Control Block Utilization

In a Clear local statistics operation, the registers in the MSTR control block (the top node) differ according to the type of network in use:

- Modbus Plus
- TCP/IP Ethernet

### Control Block for Modbus Plus

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	4
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

**Control Block for  
TCP/IP Ethernet**

## Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	4
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth implied	Slot ID	Low byte: Slot address of the network adapter module
Fifth ... Eighth implied		Reserved



---

## Write Global Data MSTR Operation

---

### Short Description

The Write global data operation transfers data to the communications processor in the current node so that it can be sent over the network when the node gets the token. All nodes on the local network link can receive this data. This operation takes one scan to complete and does not require a data master transaction path.

### Network Implementation

The Write global data operation (type 5 in the displayed register of the top node) can be implemented only for Modbus Plus networks.

### Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Write global data operation

Register	Function	Content
Displayed	Operation type	5
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Specifies the number of registers from the data area to be sent to the comm processor; the value of the length must be $\leq 32$ and must not exceed the size of the data area
Third implied		Reserved
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 <b>Note:</b> If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

---

## Read Global Data MSTR Operation

### Short Description

The Read global data operation gets data from the communications processor in any node on the local network link that is providing global data. This operation may require multiple scans to complete if global data is not currently available from the requested node. If global data is available, the operation completes in a single scan. No master transaction path is required.

### Network Implementation

The Read global data operation (type 6 in the displayed register of the top node) can be implemented only for Modbus Plus networks.

### Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Read global data operation

Register	Function	Content
Displayed	Operation type	6
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Specifies the number of words of global data to be requested from the comm processor designated by the routing 1 parameter; the value of the length must be $> 0 \leq 32$ and must not exceed the size of the data area
Third implied	Available words	Contains the number of words available from the requested node; the value is automatically updated by internal software
Fourth implied	Routing 1	The low byte specifies the address of the node whose global data are to be returned (a value between 1 ... 64); if this is the second of two local nodes, set the high byte to a value of 1 <b>Note:</b> If your PLC does not support Modbus Plus option modules (S985s or NOMs), the high byte of the fourth implied register is not used and the highbyte bits must all be set to 0.

## Get Remote Statistics MSTR Operation

**Short Description** The Get Remote Statistics operation obtains information relative to remote nodes on the network. This operation may require multiple scans to complete and does not require a master data transaction path.

**Network Implementation** The Get Remote Statistics operation (type 7 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

**Control Block Utilization** In a Get remote statistics operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using:

- Modbus Plus
- TCP/IP Ethernet

### Control Block for Modbus Plus

#### Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	7
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from an offset, the number of words of statistics to be obtained from a remote node; the length must be $> 0 \leq$ total number of statistics available (54) and must not exceed the size of the data area
Third implied	Offset	Specifies an offset value relative to the first available word in the statistics table (See <i>Modbus Plus Network Statistics</i> , p. 440), the value must not exceed the number of statistic words available.
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device.

The remote comm processor always returns its complete statistics table when a request is made, even if the request is for less than the full table. The MSTR instruction then copies only the amount of words you have requested to the designated 4x registers.

**Control Block for TCP/IP Ethernet**

## Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	7
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table (See <i>TCP/IP Ethernet Statistics</i> , p. 445); the length must be $> 0 \leq \text{data area}$
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table, if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... Eighth implied	Destination	Each register contains one byte of the 32-bit IP address

**Clear Remote Statistics MSTR Operation****Short Description**

The Clear remote statistics operation clears statistics related to a remote network node from the data area in the local node. This operation may require multiple scans to complete and uses a single data master transaction path.

**Note:** When you issue the Clear Remote Statistics operation, only words 13 ... 22 in the statistics table (See *Modbus Plus Network Statistics*, p. 440) are cleared

**Network Implementation**

The Clear remote statistics operation (type 8 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available:

- *Modbus Plus Network Statistics*, p. 440
- *TCP/IP Ethernet Statistics*, p. 445

**Control Block Utilization**

In a Clear remote statistics operation, the registers in the MSTR control block (the top node) contain information that differs according to the type of network in use:

- Modbus Plus
- TCP/IP Ethernet

**Control Block for Modbus Plus**

## Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	8
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

**Control Block for TCP/IP Ethernet**

## Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	8
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... Eighth implied	Destination	Each register contains one byte of the 32-bit IP address

---

## Peer Cop Health MSTR Operation

---

### Short Description

The peer cop health operation reads selected data from the peer cop communications health table and loads that data to specified 4x registers in state RAM. The peer cop communications health table is 12 words long, and the words are indexed via this MSTR operation as words 0 ... 11.

---

### Network Implementation

The peer cop health operation (type 9) in the displayed register of the top node) can be implemented only for Modbus Plus networks.

---

### Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Peer cop health operation:

Register	Function	Content
Displayed	Operation type	9
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data size	Number of words requested from peer cop table (range 1 ... 12)
Third implied	Index	First word from the table to be read (range 0 ... 11, where 0 = the first word in the peer cop table and 11 = the last word in the table)
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 <b>Note:</b> If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

---

### Peer Cop Communications Health Status Information

The peer cop communications health table comprises 12 contiguous registers that can be indexed in an MSTR operation as words 0 ... 11. Each bit in each of the table words is used to represent an aspect of communications health relative to a specific node on the Modbus Plus network.

---

**Bit-to-Network  
Node  
Relationship**

The bits in words 0 ... 3 represent the health of the global input communication expected from nodes 1 ... 64. The bits in words 4 ... 7 represent the health of the output from a specific node. The bits in words 8 ... 11 represent the health of the input to a specific node:

Type of Status	Word Index	Bit-to-network Node Relationship
Global Input	0	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	2	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	3	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific Output	4	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	5	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	6	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	7	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific Input	8	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	9	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	10	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	11	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49

**State of a Peer Cop Health Bit**

The state of a peer cop health bit reflects the current communication status of its associated node. A health bit is set when its associated node accepts inputs for its peer copped input data group or hears that another node has accepted specific output data from the its peer copped output data group. A health bit is cleared when no communication has occurred for its associated data group within the configured peer cop health time-out period.

All health bits are cleared when the Put Peer Cop interface command is executed at PLC start-up time. Table values are not valid until at least one full token rotation cycle has been completed after execution of the Put Peer Cop interface command. The health bit for a given node is always zero when its associated peer cop entry is null.

**Reset Option Module MSTR Operation****Short Description**

The Reset option module operation causes a Quantum NOE option module to enter a reset cycle to reset its operational environment.

**Network Implementation**

The Reset option module operation (type 10 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, accessed via the appropriate network adapter. Modbus Plus networks do not use this operation.

**Control Block Utilization**

In a Reset option module operation, the registers in the MSTR control block (the top node) differ according to the type of network in use:

- TCP/IP Ethernet
- SY/MAX Ethernet

**Control Block for TCP/IP Ethernet**

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	10
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	



**Control Block for SY/MAX Ethernet** Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	10
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fifth ... Eighth implied	Not applicable	

**Read CTE (Config Extension Table) MSTR Operation**
**Short Description**

The Read CTE operation reads a given number of bytes from the Ethernet configuration extension table to the indicated buffer in PLC memory. The bytes to be read begin at a byte offset from the beginning of the CTE. The content of the Ethernet CTE table (See *CTE Display Implementation (Middle Node)*, p. 437) is displayed in the middle node of the MSTR block.

**Network Implementation**

The Read CTE operation (type 11 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, accessed via the appropriate network adapter. Modbus Plus networks do not use this operation.

**Control Block Utilization**

In a Read CTE operation, the registers in the MSTR control block (the top node) differ according to the type of network in use:

- TCP/IP Ethernet
- SY/MAX Ethernet

**Control Block for TCP/IP Ethernet**

## Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	11
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Map index	Either a value displayed in the high byte of the register or not used
	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	

**Control Block for SY/MAX Ethernet**

## Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	11
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data Size	Number of words transferred
Third implied	Base Address	Byte offset in PLC register structure indicating where the CTE bytes will be written
Fourth implied	Low byte	Slot address of the NOE module
	High byte	Terminator (FF hex)
Fifth ... Eighth implied	Not applicable	

**CTE Display Implementation (Middle Node)**

The values in the Ethernet configuration extension table (CTE) are displayed in a series of registers in the middle node of the MSTR instruction when a Read CTE operation is implemented. The middle node contains the first of 11 contiguous 4x registers.

The registers display the following CTE data:

Parameter	Register	Content
Frame type	Displayed	1 = 802.3 2 = Ethernet
IP address	First implied	First byte of the IP address
	Second implied	Second byte of the IP address
	Third implied	Third byte of the IP address
	Fourth implied	Fourth byte of the IP address
Subnetwork mask	Fifth implied	Hi word
	Sixth implied	Low word
Gateway	Seventh implied	First byte of the gateway
	Eighth implied	Second byte of the gateway
	Ninth implied	Third byte of the gateway
	Tenth implied	Fourth byte of the gateway

**Write CTE (Config Extension Table) MSTR Operation****Short Description**

The Write CTE operation writes the configuration CTE table from the data specified in the middle node to an indicated Ethernet configuration extension table or a specified slot.

**Network Implementation**

The Write CTE operation (type 12 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, via the appropriate network adapter. Modbus Plus networks do not use this operation.

**Control Block Utilization**

In a Write CTE operation, the registers in the MSTR control block (the top node) differ according to the type of network in use:

- TCP/IP Ethernet
- SY/MAX Ethernet

**Control Block for TCP/IP Ethernet**

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	12
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Map index	Either a value displayed in the high byte of the register or not used
	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	

**Control Block for SY/MAX Ethernet**

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	12
First implied	Error status (See <i>Run Time Errors</i> , p. 446)	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data Size	Number of words transferred
Third implied	Base Address	Byte offset in PLC register structure indicating where the CTE bytes will be written
Fourth implied	Low byte	Slot address of the NOE module
	High byte	Destination drop number
Fifth implied	Terminator	FF hex
Sixth ... Eighth implied	Not applicable	

**CTE Display  
Implementation  
(Middle Node)**

The values in the Ethernet configuration extension table (CTE) are displayed in a series of registers in the middle node of the MSTR instruction when a Write CTE operation is implemented. The middle node contains the first of 11 contiguous 4x registers.

The registers are used to transfer the following CTE data:

Parameter	Register	Content
Frame type	Displayed	1 = 802.3 2 = Ethernet
IP address	First implied	First byte of the IP address
	Second implied	Second byte of the IP address
	Third implied	Third byte of the IP address
	Fourth implied	Fourth byte of the IP address
Subnetwork mask	Fifth implied	Hi word
	Sixth implied	Low word
Gateway	Seventh implied	First byte of the gateway
	Eighth implied	Second byte of the gateway
	Ninth implied	Third byte of the gateway
	Tenth implied	Fourth byte of the gateway

## Modbus Plus Network Statistics

### Modbus Plus Network Statistics

The following table shows the statistics available on the Modbus Plus network. You may acquire this information by using the appropriate MSTR operation or by using Modbus function code 8.

**Note:** When you issue the Clear local or Clear remote statistics operations, only words 13 ... 22 are cleared.

#### Modbus Plus Network Statistics

Word	Bits	Meaning																																															
00		Node type ID																																															
	0	Unknown node type																																															
	1	PLC node																																															
	2	Modbus bridge node																																															
	3	Host computer node																																															
	4	Bridge Plus node																																															
	5	Peer I/O node																																															
01	0 ... 11	Software version number in hex (to read, strip bits 12-15 from word)																																															
	12 ... 14	Reserved																																															
	15	Defines Word 15 error counters (see Word 15) Most significant bit defines use of error counters in Word 15. Least significant half of upper byte, plus lower byte, contain software version: <div style="text-align: center; margin: 10px 0;"> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px 5px;">15</td><td style="padding: 2px 5px;">14</td><td style="padding: 2px 5px;">13</td><td style="padding: 2px 5px;">12</td><td style="padding: 2px 5px;">11</td><td style="padding: 2px 5px;">10</td><td style="padding: 2px 5px;">9</td><td style="padding: 2px 5px;">8</td><td style="padding: 2px 5px;">7</td><td style="padding: 2px 5px;">6</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">4</td><td style="padding: 2px 5px;">3</td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td> </tr> </table> </div> <div style="margin-left: 100px; margin-top: 5px;"> <table style="border: none;"> <tr> <td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td><td style="border: none;"> </td> </tr> <tr> <td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td><td style="border: none;"></td> </tr> </table> </div> <div style="margin-left: 100px; margin-top: 5px;">                     Software version number (in HEX)                 </div> <div style="margin-left: 100px; margin-top: 10px;">                     Word 15 error counter (see word 15)                 </div>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
02		Network address for this station																																															

Word	Bits	Meaning
03		MAC state variable:
	0	Power up state
	1	Monitor offline state
	2	Duplicate offline state
	3	Idle state
	4	Use token state
	5	Work response state
	6	Pass token state
	7	Solicit response state
	8	Check pass state
	9	Claim token state
10	Claim response state	
04		Peer status (LED code); provides status of this unit relative to the network:
	0	Monitor link operation
	32	Normal link operation
	64	Never getting token
	96	Sole station
128	Duplicate station	
05		Token pass counter; increments each time this station gets the token
06		Token rotation time in ms
07	LO	Data master failed during token ownership bit map
	HI	Program master failed during token ownership bit map
08	LO	Data master token owner work bit map
	HI	Program master token owner work bit map
09	LO	Data slave token owner work bit map
	HI	Program slave token owner work bit map
10	HI	Data slave/get slave command transfer request bit map
11	LO	Program master/get master rsp transfer request bit map
	HI	Program slave/get slave command transfer request bit map
12	LO	Program master connect status bit map
	HI	Program slave automatic logout request bit map
13	LO	Pretransmit deferral error counter
	HI	Receive buffer DMA overrun error counter

Word	Bits	Meaning
14	LO	Repeated command received counter
	HI	Frame size error counter
15		<b>If Word 1 bit 15 is not set</b> , Word 15 has the following meaning:
	LO	Receiver collision-abort error counter
	HI	Receiver alignment error counter
		<b>If Word 1 bit 15 is set</b> , Word 15 has the following meaning:
	LO	Cable A framing error
	HI	Cable B framing error
16	LO	Receiver CRC error counter
	HI	Bad packet-length error counter
17	LO	Bad link-address error counter
	HI	Transmit buffer DMA-underrun error counter
18	LO	Bad internal packet length error counter
	HI	Bad MAC function code error counter
19	LO	Communication retry counter
	HI	Communication failed error counter
20	LO	Good receive packet success counter
	HI	No response received error counter
21	LO	Exception response received error counter
	HI	Unexpected path error counter
22	LO	Unexpected response error counter
	HI	Forgotten transaction error counter
23	LO	Active station table bit map, nodes 1 ... 8
	HI	Active station table bit map, nodes 9 ... 16
24	LO	Active station table bit map, nodes 17 ... 24
	HI	Active station table bit map, nodes 25 ... 32
25	LO	Active station table bit map, nodes 33 ... 40
	HI	Active station table bit map, nodes 41 ... 48
26	LO	Active station table bit map, nodes 49 ... 56
	HI	Active station table bit map, nodes 57 ... 64
27	LO	Token station table bit map, nodes 1 ... 8
	HI	Token station table bit map, nodes 9 ... 16
28	LO	Token station table bit map, nodes 17 ... 24
	HI	Token station table bit map, nodes 25 ... 32



Word	Bits	Meaning
29	LO	Token station table bit map, nodes 33 ... 40
	HI	Token station table bit map, nodes 41 ... 48
30	LO	Token station table bit map, nodes 49 ... 56
	HI	Token station table bit map, nodes 57 ... 64
31	LO	Global data present table bit map, nodes 1 ... 8
	HI	Global data present table bit map, nodes 9 ... 16
32	LO	Global data present table bit map, nodes 17 ... 24
	HI	Global data present table bit map, nodes 25 ... 32
33	LO	Global data present table bit map, nodes 33 ... 40
	HI	Global data present table bit map, nodes 41 ... 48
34	LO	Global data present table map, nodes 49 ... 56
	HI	Global data present table bit map, nodes 57 ... 64
35	LO	Receive buffer in use bit map, buffer 1-8
	HI	Receive buffer in use bit map, buffer 9 ... 16
36	LO	Receive buffer in use bit map, buffer 17 ... 24
	HI	Receive buffer in use bit map, buffer 25 ... 32
37	LO	Receive buffer in use bit map, buffer 33 ... 40
	HI	Station management command processed initiation counter
38	LO	Data master output path 1 command initiation counter
	HI	Data master output path 2 command initiation counter
39	LO	Data master output path 3 command initiation counter
	HI	Data master output path 4 command initiation counter
40	LO	Data master output path 5 command initiation counter
	HI	Data master output path 6 command initiation counter
41	LO	Data master output path 7 command initiation counter
	HI	Data master output path 8 command initiation counter
42	LO	Data slave input path 41 command processed counter
	HI	Data slave input path 42 command processed counter
43	LO	Data slave input path 43 command processed counter
	HI	Data slave input path 44 command processed counter
44	LO	Data slave input path 45 command processed counter
	HI	Data slave input path 46 command processed counter
45	LO	Data slave input path 47 command processed counter
	HI	Data slave input path 48 command processed counter

---

Word	Bits	Meaning
46	LO	Program master output path 81 command initiation counter
	HI	Program master output path 82 command initiation counter
47	LO	Program master output path 83 command initiation counter
	HI	Program master output path 84 command initiation counter
48	LO	Program master command initiation counter
	HI	Program master output path 86 command initiation counter
49	LO	Program master output path 87 command initiation counter
	HI	Program master output path 88 command initiation counter
50	LO	Program slave input path C1 command processed counter
	HI	Program slave input path C2 command processed counter
51	LO	Program slave input path C3 command processed counter
	HI	Program slave input path C4 command processed counter
52	LO	Program slave input path C5 command processed counter
	HI	Program slave input path C6 command processed counter
53	LO	Program slave input path C7 command processed counter
	HI	Program slave input path C8 command processed counter

---

## TCP/IP Ethernet Statistics

### TCP/IP Ethernet Statistics

A TCP/IP Ethernet board responds to Get Local Statistics and Set Local Statistics commands with the following information:

Word	Meaning	
00 ... 02	MAC addr., e.g., if the MAC addr. is 00 00 54 00 12 34, it is displayed as follows:	
	Word	Content
	00	00 00
	01	00 54
	02	34 12
03	Board status	Meaning
	0x0001	Running
	0x4000	APPI LED (1=ON, 0 = OFF)
	0x8000	Link LED
04 and 05	Number of receiver interrupts	
06 and 07	Number of transmitter interrupts	
08 and 09	Transmit-timeout error count	
10 and 11	Collision-detect error count	
12 and 13	Missed packets	
14 and 15	Memory error count	
16 and 17	Number of times driver has restarted lance	
18 and 19	Receive framing error count	
20 and 21	Receiver overflow error count	
22 and 23	Receive CRC error count	
24 and 25	Receive buffer error count	
26 and 27	Transmit buffer error count	
28 and 29	Transmit silo underflow count	
30 and 31	Late collision count	
32 and 33	Lost carrier count	
34 and 35	Number of retries	
36 and 37	IP addr., e.g., if the IP addr. is 198.202.137.113 (or c6 CA 89 71), it is displayed as follows:	
	Word	Content
	36	89 71
	37	C6 CA

---

## Run Time Errors

---

- Runtime Errors** If an error occurs during a MSTR operation, a hexadecimal error code will be displayed in the first implied register in the control block (the top node).  
Function error codes are network-specific:
- *Modbus Plus and SY/MAX Ethernet Error Codes, p. 446*
  - *SY/MAX-specific Error Codes, p. 448*
  - *TCP/IP Ethernet Error Codes, p. 450*
  - *CTE Error Codes for SY/MAX and TCP/IP Ethernet, p. 452*
- 

## Modbus Plus and SY/MAX Ethernet Error Codes

---

- Form of the Function Error Code** The form of the function error code for Modbus Plus and SY/MAX Ethernet transactions is **Mmss**, where
- **M** represents the major code
  - **m** represents the minor code
  - **ss** represents a subcode
- 

- Hexadecimal Error Code** HEX Error Code for Modbus Plus and SY/MAX Ethernet:

Hex Error Code	Meaning
1001	User has aborted the MSTR element
2001	An unsupported operation type has been specified in the control block
2002	One or more control block parameter has been changed while the MSTR element is active (applies only to operations that take multiple scans to complete). Control block parameters may be changed only when the MSTR element is not active.
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid values in the length and offset fields of the control block
2006	Invalid slave device data area
2007	Invalid slave device network area
2008	Invalid slave device network routing
2009	Route equal to your own address
200A	Attempting to obtain more global data words than available
30ss	Modbus slave exception response (See <i>ss HEX Value in Error Code 30ss, p. 447</i> )
4001	Inconsistent Modbus slave response

Hex Error Code	Meaning
5001	Inconsistent network response
6mss)	Routing failure (See <i>ss Hex Value in Error Code 6mss, p. 447</i> )

### ss HEX Value in Error Code 30ss

The ss subfield in error code 30ss is:

ss Hex Value	Meaning
01	Slave device does not support the requested operation
02	Nonexistent slave device registers requested
03	Invalid data value requested
04	Reserved
05	Slave has accepted long-duration program command
06	Function can't be performed now: a long-duration command in effect
07	Slave rejected long-duration program command
08 ... 255	Reserved

### ss Hex Value in Error Code 6mss

The m subfield in error code 6mss is an index into the routing information indicating where an error has been detected (a value of 0 indicates the local node, a 2 the second device on the route, etc.).

The ss subfield in error code 6mss is:

ss Hex Value	Meaning
01	No response received
02	Program access denied
03	Node off-line and unable to communicate
04	Exception response received
05	Router node data paths busy
06	Slave device down
07	Bad destination address
08	Invalid node type in routing path
10	Slave has rejected the command
20	Initiated transaction forgotten by slave device
40	Unexpected master output path received
80	Unexpected response received
F001	Wrong destination node specified for the MSTR operation

## SY/MAX-specific Error Codes

**Types or Errors** Three additional types of errors may be reported in the MSTR instruction when SY/MAX Ethernet is being used.

The error codes have the following designations:

- 71xx errors: Errors detected by the remote SY/MAX device
- 72xx errors: Errors detected by the serve
- 73xx errors: Errors detected by the Quantum translator

### Hexadecimal Error Code SY/MAX-specific

HEX Error Code SY/MAX-specific:

Hex Error Code	Meaning
7101	Illegal opcode detected by the remote SY/MAX device
7103	Illegal address detected by the remote SY/MAX device
7109	Attempt to write a read-only register detected by the remote SY/MAX device
710F	Receiver overflow detected by the remote SY/MAX device
7110	Invalid length detected by the remote SY/MAX device
7111	Remote device inactive, not communicating (occurs after retries and time-out have been exhausted) detected by the remote SY/MAX device
7113	Invalid parameter on a read operation detected by the remote SY/MAX device
711D	Invalid route detected by the remote SY/MAX device
7149	Invalid parameter on a write operation detected by the remote SY/MAX device
714B	Illegal drop number detected by the remote SY/MAX device
7201	Illegal opcode detected by the SY/MAX server
7203	Illegal address detected by the SY/MAX server
7209	Attempt to write to a read-only register detected by the SY/MAX server
720F	Receiver overflow detected by the SY/MAX server
7210	Invalid length detected by the SY/MAX server
7211	Remote device inactive, not communicating (occurs after retries and time-out have been exhausted) detected by the SY/MAX server
7213	Invalid parameter on a read operation detected by the SY/MAX server
721D	Invalid route detected by the SY/MAX server
7249	Invalid parameter on a write operation detected by the SY/MAX server
724B	Illegal drop number detected by the SY/MAX server

---

Hex Error Code	Meaning
7301	Illegal opcode in an MSTR block request by the Quantum translator
7303	Read/Write QSE module status (200 route address out of range)
7309	Attempt to write to a read-only register when performing a status write (200 route)
731D	Invalid rout detected by Quantum translator Valid routes are: <ul style="list-style-type: none"><li>● dest_drop, 0xFF</li><li>● 200, dest_drop, 0xFF</li><li>● 100+drop, dest_drop, 0xFF</li></ul> All other routing values generate an error
734B	One of the following errors has occurred: <ul style="list-style-type: none"><li>● No CTE (configuration extension) table was configured</li><li>● No CTE table entry was created for the QSE Module slot number</li><li>● No valid drop was specified</li><li>● The QSE Module was not reset after the CTE was created</li></ul> <b>Note:</b> After writing and configuring the CTE and downloading it to the QSE Module, you must reset the QSE Module to make the changes take effect. <ul style="list-style-type: none"><li>● When using an MSTR instruction, no valid slot or drop was specified</li></ul>

---

## TCP/IP Ethernet Error Codes

### Error in an MSTR Routine

An error in an MSTR routine over TCP/IP Ethernet may produce one of the following errors in the MSTR control block.

The form of the code is **Mmss**, where

- **M** represents the major code
- **m** represents the minor code
- **ss** represents a subcode

### Hexadecimal Error Code for MSTR Routine over TCP/IP Ethernet

HEX Error Code MSTR routine over TCP/IP Ethernet:

Hex Error Code	Meaning
1001	User has aborted the MSTR element
2001	An unsupported operation type has been specified in the control block
2002	One or more control block parameter has been changed while the MSTR element is active (applies only to operations that take multiple scans to complete). Control block parameters may be changed only when the MSTR element is not active
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid values in the length and offset fields of the control block
2006	Invalid slave device data area
3000	Generic Modbus fail code
30ss	Modbus slave exception response (See <i>ss Hex Value in Error Code 30ss</i> , p. 450)
4001	Inconsistent Modbus slave response

### ss Hex Value in Error Code 30ss

The ss subfield in error code 30ss is:

ss Hex Value	Meaning
01	Slave device does not support the requested operation
02	Nonexistent slave device registers requested
03	Invalid data value requested
04	Reserved
05	Slave has accepted long-duration program command
06	Function can't be performed now: a long-duration command in effect
07	Slave rejected long-duration program command



**HEX Error Code  
TCP/IP Ethernet  
Network**

An error on the TCP/IP Ethernet network itself may produce one of the following errors in the MSTR control block:

Hex Error Code	Meaning
5004	Interrupted system call
5005	I/O error
5006	No such address
5009	The socket descriptor is invalid
500C	Not enough memory
500D	Permission denied
5011	Entry exists
5016	An argument is invalid
5017	An internal table has run out of space
5020	The connection is broken
5023	This operation would block and the socket is nonblocking
5024	The socket is nonblocking and the connection cannot be completed
5025	The socket is nonblocking and a previous connection attempt has not yet completed
5026	Socket operation on a nonsocket
5027	The destination address is invalid
5028	Message too long
5029	Protocol wrong type for socket
502A	Protocol not available
502B	Protocol not supported
502C	Socket type not supported
502D	Operation not supported on socket
502E	Protocol family not supported
502F	Address family not supported
5030	Address is already in use
5031	Address not available
5032	Network is down
5033	Network is unreachable
5034	Network dropped connection on reset
5035	The connection has been aborted by the peer
5036	The connection has been reset by the peer
5037	An internal buffer is required, but cannot be allocated
5038	The socket is already connected

Hex Error Code	Meaning
5039	The socket is not connected
503A	Can't send after socket shutdown
503B	Too many references; can't splice
503C	Connection timed out
503D	The attempt to connect was refused
5040	Host is down
5041	The destination host could not be reached from this node
5042	Directory not empty
5046	NL_INIT returned -1
5047	The MTU is invalid
5048	The hardware length is invalid
5049	The route specified cannot be found
504A	Collision in select call; these conditions have already been selected by another task
504B	The task id is invalid
F001	In Reset mode

---

## CTE Error Codes for SY/MAX and TCP/IP Ethernet

---

### CTE Error Codes for SY/MAX and TCP/IP Ethernet

HEX Error Code MSTR routine over TCP/IP Ethernet:

Hex Error Code	Meaning
7001	The is no Ethernet configuration extension
7002	The CTE is not available for access
7003	The offset is invalid
7004	The offset + length is invalid
7005	Bad data field in the CTE

---

---

## MU16: Multiply 16 Bit

89

---

### At a Glance

#### Introduction

This chapter describes the instruction MU16.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	454
Representation	454

## Short Description

---

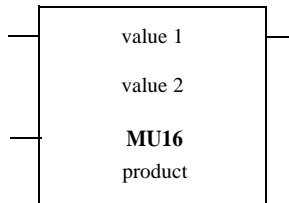
**Function Description** The MU16 instruction performs signed or unsigned multiplication on the 16-bit values in the top and middle nodes, then posts the product in two contiguous holding registers in the bottom node.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables value 1 x value 2
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Multiplicand, can be displayed explicitly as an integer (range 1 ... 65 535, enter e.g. #65535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Multiplier, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
product (bottom node)	4x	INT, UINT	First of two contiguous holding registers: displayed register contains half of the product and the implied register contains the other half
Top output	0x	None	Echoes the state of the top input

---

---

## MUL: Multiply

90

---

### At a Glance

#### Introduction

This chapter describes the instruction MUL.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	456
Representation	456
Example	456

## Short Description

---

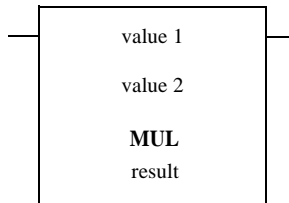
**Function Description** The MUL instruction multiplies unsigned value 1 (its top node) by unsigned value 2 (its middle node) and stores the product in two contiguous holding registers in the bottom node.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = value 1 multiplied by value 2
value 1 (top node)	3x, 4x	UINT	Multiplicand, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
value 2 (middle node)	3x, 4x	UINT	Multiplier, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
result (bottom node)	4x	UINT	Product (first of two contiguous holding registers; displayed: high-order digits; implied: low-order digits)
Top output	0x	None	Echoes the state of the top input

---

## Example

---

**Product of Instruction MUL** For example, if value 1 = 8 000 and value 2 = 2, the product is 16 000. The displayed register contains the value 0001 (the high-order half of the product), and implied register contains the value 6 000 (the low-order half of the product).

---

---

## NBIT: Bit Control

91

---

### At a Glance

#### Introduction

This chapter describes the instruction NBIT.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	458
Representation	458

## Short Description

---

**Function Description**

The normal bit (NBIT) instruction lets you control the state of a bit from a register by specifying its associated bit number in the bottom node. The bits being controlled are similar to coils, when a bit is turned ON, it stays ON until a control signal turns it OFF.

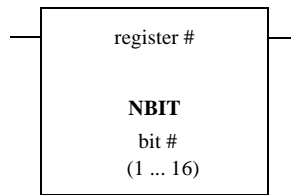
**Note:** The NBIT instruction does not follow the same rules of network placement as 0x-referenced coils do. An NBIT instruction cannot be placed in column 11 of a network and it can be placed to the left of other logic nodes on the same rungs of the ladder.

## Representation

---

**Symbol**

Representation of the instruction



**Parameter Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = sets the specified bit to 1 OFF = clears the specified bit to 0
register # (top node)	4x	WORD	Holding register whose bit pattern is being controlled
bit # (bottom node)		INT, UINT	Indicates which one of the 16 bits is being controlled
Top output	0x	None	Echoes the state of the top input: ON = top input ON and specified bit set to 1 OFF = top input OFF and specified bit set to 0



---

## NCBT: Normally Closed Bit

92

---

### At a Glance

#### Introduction

This chapter describes the instruction NCBT.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	460
Representation	460

## Short Description

---

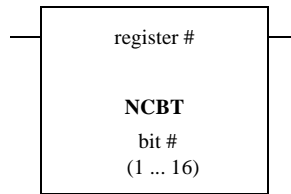
**Function Description** The normally closed bit (NCBT) instruction lets you sense the logic state of a bit in a register by specifying its associated bit number in the bottom node. The bit is representative of an N.C contact. It passes power from the top output when the specified bit is OFF and the top input is ON.

---

## Representation

---

**Symbol** Representation of the instruction



**Parameter Description** Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables bit sensing
register # (top node)	3x, 4x	WORD	Register whose bit pattern is being used to represent N.C. contacts
bit # (bottom node)		INT, UINT	(Indicates which one of the 16 bits is being sensed)
Top output	0x	None	ON = top input is ON and specified bit is OFF (logic state 0)

---

---

## NOBT: Normally Open Bit

93

---

### At a Glance

#### Introduction

This chapter describes the instruction NOBT.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	462
Representation	462

NOBT: Normally Open Bit

---

## Short Description

---

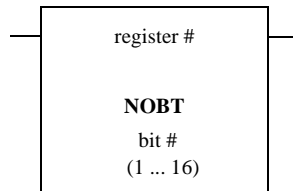
**Function Description** The normally open bit (NOBT) instruction lets you sense the logic state of a bit in a register by specifying its associated bit number in the bottom node. The bit is representative of an N.O contact.

---

## Representation

---

**Symbol** Representation of the instruction



## Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables bit sensing
register # (top node)	3x, 4x	WORD	Register whose bit pattern is being used to represent N.O. contacts
bit # (bottom node)		INT, UINT	(Indicates which one of the 16 bits is being sensed)
Top output	0x	None	ON = top input is ON and specified bit is ON (logic state 1)

---

---

## NOL: Network Option Module for Lonworks

94

---

### At a Glance

#### Introduction

This chapter describes the instruction NOL.

#### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	464
Representation	466
Detailed Description	465

---


## Short Description

---

### Function Requirements

The following steps are necessary before using this instruction:

Step	Action
1	Add loadable <b>NSUP.exe</b> to the controller's configuration <b>Note:</b> This loadable needs only be loaded once to support multiple loadables, such as ECS.exe and XMIT.exe.

	<b>CAUTION</b>
	<p><b>The outputs of the instruction turn on, regardless of the input states</b></p> <p>When the NSUP loadable is not installed or is installed after the NOL loadable or is installed in a Quantum PLC with an executive &lt; V2.0, all three outputs turn on, regardless of the input states.</p> <p><b>Failure to observe this precaution can result in injury or equipment damage.</b></p>

Step	Action
2	Unpack and install the DX Loadable NOL; further information you will find in the chapter <i>Installation of DX Loadables</i> , p. 41.

### Function Description

The NOL instruction is provided to facilitate the movement of the large amount of data between the NOL module and the controller register space. The NOL Module is mapped for 16 input registers (3X) and 16 output registers (4X). Of these registers, two input and two output registers are for handshaking between the NOL Module and the instruction. The remaining fourteen input and fourteen output registers are used to transport the data.

---

## Detailed Description

### Register Block (Middle Node)

This block provides the registers for configuration and status information, the registers for the health status bits and the registers for the actual data of the Standard Network Variable Types (SNVTs).  
Register Block

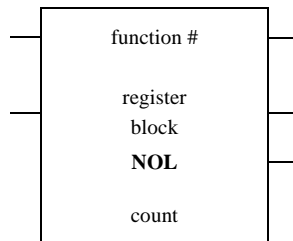
	Register	Content
Configuration and Status information	Displayed and first implied	I/O Map input base (3x)
	Second and third implied	I/O Map output base (4x)
	Fourth implied	Enable health bits
	Fifth implied	Number of input registers
	Sixth implied	Number of output registers
	Seventh implied	Number of discrete input registers
	Eighth implied	Number of discrete output registers
	Ninth implied	Config checksum (CRC)
	Tenth implied	NOL version
	Eleventh implied	Module firmware version
	Twelfth implied	NOL DX version
	Thirteenth implied	Module DX version
		14th to 15th implied
SNVTs Health Bit Status (if enabled in DX-Zoom screen)	16th to 31st implied	Health bits of each programmable network variable
SNVTs Actual Data	Enable Health Bit = NO: from 16th implied up	Data is stored in 4 groups: <ul style="list-style-type: none"> <li>● Discrete inputs</li> <li>● Register inputs</li> <li>● Discrete outputs</li> <li>● Register outputs</li> </ul> These groups of data are set up consecutively and start on word boundaries.
	Enable Health Bit = YES: from 32nd implied up	

The first 16 registers with configuration and status information can be programmed and monitored via the NOL DX Zoom screen. For setting up the link to the NOL module the only parameters that need to be entered are the beginning 3x and 4x registers used when I/O mapping the NOL module. Further information you will find in the documentation *Network Option Module for LonWorks*.

**Count (Bottom Node)** Defines the total number of registers required by the function block. This value must be set to a value equal to or greater than the number of data registers required to transfer and store the network data being used by the NOL module. If the count value is not large enough for the required data, the error output will be set.

## Representation

**Symbol** Representation of the instruction



## Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = Enables the NOL function
Middle input	0x, 1x	None	ON = Initialize: causes the instruction to re-sync with the module
function # (top node)	4x	INT, UINT, WORD	Function number selects the function of the NOL block Function 0 transfers data to/from the module. Any other function number yields an error.
register block (middle node)	4x	INT, UINT, WORD	Register block (first of 16 contiguous registers)
count (bottom node)		INT, UINT	Total number of registers required by the instruction
Top output	0x	None	ON = instruction enabled and no error
Middle output	0x	None	New data Set for one sweep when the entire data block from the module has been written to the register area.
Bottom output	0x	None	ON = Error



---

## OR: Logical OR

95

---

### At a Glance

#### Introduction

This chapter describes the instruction OR.

#### What's in this chapter?

This chapter contains the following topics:

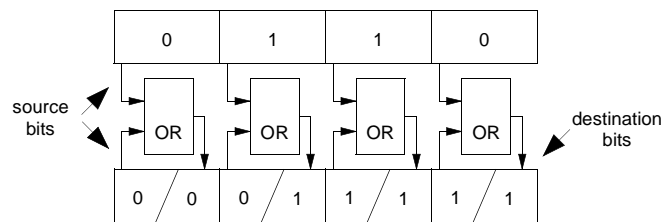
Topic	Page
Short Description	468
Representation	469
Parameter Description	469

## Short Description

---

### Function Description

The OR instruction performs a Boolean OR operation on the bit patterns in the source and destination matrices. The ORed bit pattern is then posted in the destination matrix, overwriting its previous contents.



#### WARNING



#### Overriding of any disabled coils within the destination matrix without enabling them

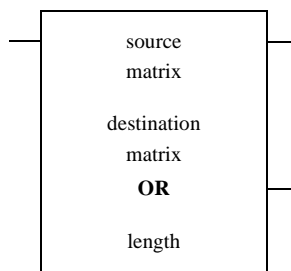
OR will override any disabled coils within the destination matrix without enabling them. This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's state can be changed by the OR operation.

**Failure to observe this precaution can result in severe injury or equipment damage.**

## Representation

### Symbol

Representation of the instruction



### Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Initiates OR
source matrix (top node)	0x, 1x, 3x, 4x	ANY_BIT	First reference in the source matrix.
destination matrix (middle node)	0x, 4x	ANY_BIT	First reference in the destination matrix
length (bottom node)		INT, UINT	Matrix length, range: 1 ... 100.
Top output	0x	None	Echoes state of the top input

## Parameter Description

### Matrix Length (Bottom Node)

The integer entered in the bottom node specifies the matrix length, i.e. the number of registers or 16-bit words in the two matrices. The matrix length can be in the range 1 ... 100. A length of 2 indicates that 32 bits in each matrix will be ORed.

OR: Logical OR

---

---

## Index



### A

AD16, 55  
ADD, 57  
Add 16 Bit, 55  
Addition, 57  
    AD16, 55  
    ADD, 57  
Advanced Calculations, 472  
Analog Input, 479  
Analog Output, 489  
Analog Values, 15  
AND, 59  
ASCII Functions  
    READ, 615  
    WRIT, 707  
Average Weighted Inputs Calculate, 493

### B

Base 10 Antilogarithm, 139  
Base 10 Logarithm, 215  
BCD, 63  
Binary to Binary Code, 63  
Bit Control, 457  
Bit pattern comparison  
    CMPR, 89  
Bit Rotate, 75  
BLKM, 65  
BLKT, 69  
Block Move, 65  
Block Move with Interrupts Disabled, 73  
Block to Table, 69

BMDI, 73  
BROT, 75

### C

Calculated preset formula, 499  
Central Alarm Handler, 485  
Changing the Sign of a Floating Point Number, 155  
Check Sum, 85  
CHS, 79  
CKSM, 85  
Closed Loop Control, 15  
CMPR, 89  
Coils, 43  
Communications  
    MSTR, 411  
COMP, 93  
Compare Register, 89  
Complement a Matrix, 93  
Comprehensive ISA Non Interacting PID, 519  
Configure Hot Standby, 79  
Contacts, 43  
Conversion  
    BCD to binary, 63  
    binary to BCD, 63

## Counters / Timers

- T.01 Timer, 687
- T0.1 Timer, 689
- T1.0 Timer, 691
- T1MS Timer, 693
- UCTR, 705

## Counters/Timers

- DCTR, 97

**D**

## Data Logging for PCMCIA Read/Write

Support, 107

DCTR, 97

Derivative Rate Calculation over a Specified Time, 567

DIOH, 99

Distributed I/O Health, 99

DIV, 103

Divide, 103

Divide 16 Bit, 117

DLOG, 107

Double Precision Addition, 127

Double Precision Division, 187

Double Precision Multiplication, 223

Double Precision Subtraction, 259

Down Counter, 97

DRUM, 113

DRUM Sequencer, 113

DV16, 117

**E**

EMTH, 121

EMTH Subfunction

- EMTH-ADDDP, 127
- EMTH-ADDFP, 131, 135
- EMTH-ANLOG, 139
- EMTH-ARCOS, 143
- EMTH-ARSIN, 147
- EMTH-ARTAN, 151
- EMTH-CHSIN, 155
- EMTH-CMPFP, 159
- EMTH-CMPIF, 163
- EMTH-CNVDR, 167

- EMTH-CNVFI, 171
- EMTH-CNVIF, 175
- EMTH-CNVDR, 179
- EMTH-COS, 183
- EMTH-DIVDP, 187
- EMTH-DIVFI, 191
- EMTH-DIVFP, 195
- EMTH-DIVIF, 199
- EMTH-ERLOG, 203
- EMTH-EXP, 207
- EMTH-LNFP, 211
- EMTH-LOG, 215
- EMTH-LOGFP, 219
- EMTH-MULDP, 223
- EMTH-MULFP, 227
- EMTH-MULIF, 231
- EMTH-PI, 235
- EMTH-POW, 239
- EMTH-SINE, 243
- EMTH-SQRFP, 247
- EMTH-SQRT, 251
- EMTH-SQRTP, 255
- EMTH-SUBDP, 259
- EMTH-SUBFI, 263
- EMTH-SUBFP, 267
- EMTH-SUBIF, 271
- EMTH-TAN, 275
- EMTH-ADDDP, 127
- EMTH-ADDFP, 131
- EMTH-ADDIF, 135
- EMTH-ANLOG, 139
- EMTH-ARCOS, 143
- EMTH-ARSIN, 147
- EMTH-ARTAN, 151
- EMTH-CHSIN, 155
- EMTH-CMPFP, 159
- EMTH-CMPIF, 163
- EMTH-CNVDR, 167
- EMTH-CNVFI, 171
- EMTH-CNVIF, 175
- EMTH-CNVDR, 179
- EMTH-COS, 183
- EMTH-DIVDP, 187
- EMTH-DIVFI, 191
- EMTH-DIVFP, 195
- EMTH-DIVIF, 199

- 
- EMTH-ERLOG, 203
  - EMTH-EXP, 207
  - EMTH-LNFP, 211
  - EMTH-LOG, 215
  - EMTH-LOGFP, 219
  - EMTHMULDP, 223
  - EMTH-MULFP, 227
  - EMTH-MULIF, 231
  - EMTH-PI, 235
  - EMTH-POW, 239
  - EMTH-SINE, 243
  - EMTH-SQRFP, 247
  - EMTH-SQRT, 251
  - EMTH-SQRTP, 255
  - EMTH-SUBDP, 259
  - EMTH-SUBFI, 263
  - EMTH-SUBFP, 267
  - EMTH-SUBIF, 271
  - EMTH-TAN, 275
  - Engineering Unit Conversion and Alarms, 297
  - ESI, 279
  - EUCA, 297
  - Exclusive OR, 731
  - Extended Math, 121
  - Extended Memory Read, 723
  - Extended Memory Write, 727
- F**
- Fast I/O Instructions
    - BMDI, 73
    - ID, 343
    - IE, 347
    - IMIO, 351
    - IMOD, 357
    - ITMR, 365
  - FIN, 309
  - First In, 309
  - First Out, 313
  - First-order Lead/Lag Filter, 537
  - Floating Point - Integer Subtraction, 263
  - Floating Point Addition, 131
  - Floating Point Arc Cosine of an Angle (in Radians), 143
  - Floating Point Arc Tangent of an Angle (in Radians), 151
  - Floating Point Arcsine of an Angle (in Radians), 147
  - Floating Point Common Logarithm, 219
  - Floating Point Comparison, 159
  - Floating Point Conversion of Degrees to Radians, 167
  - Floating Point Conversion of Radians to Degrees, 179
  - Floating Point Cosine of an Angle (in Radians), 183
  - Floating Point Divided by Integer, 191
  - Floating Point Division, 195
  - Floating Point Error Report Log, 203
  - Floating Point Exponential Function, 207
  - Floating Point Multiplication, 227
  - Floating Point Natural Logarithm, 211
  - Floating Point Sine of an Angle (in Radians), 243
  - Floating Point Square Root, 247, 251
  - Floating Point Subtraction, 267
  - Floating Point Tangent of an Angle (in Radians), 275
  - Floating Point to Integer, 317
  - Floating Point to Integer Conversion, 171
  - Formatted Equation Calculator, 509
  - Formatting Messages, 29
  - Four Station Ratio Controller, 571
  - FOUT, 313
  - FTOI, 317
- H**
- History and Status Matrices, 319
  - HLTH, 319
  - Hot standby
    - CHS, 79
- I**
- IBKR, 333
  - IBKW, 335
  - ICMP, 337
  - ID, 343
  - IE, 347
  - IMIO, 351

Immediate I/O, 351  
IMOD, 357  
Indirect Block Read, 333  
Indirect Block Write, 335  
Input Compare, 337  
Input Selection, 581  
Installation of DX Loadables, 41  
Instruction  
    Coils, Contacts and Interconnects, 43  
Instruction Groups, 5  
    ASCII Communication Instructions, 7  
    Coils, Contacts and Interconnects, 14  
    Counters and Timers Instructions, 7  
    Fast I/O Instructions, 8  
    Loadable DX, 9  
    Math Instructions, 9  
    Matrix Instructions, 11  
    Miscellaneous, 12  
    Move Instructions, 13  
    Overview, 6  
    Skips/Specials, 13  
    Special Instructions, 14  
Integer - Floating Point Subtraction, 271  
Integer + Floating Point Addition, 135  
Integer Divided by Floating Point, 199  
Integer to Floating Point, 371  
Integer x Floating Point Multiplication, 231  
Integer-Floating Point Comparison, 163  
Integer-to-Floating Point Conversion, 175  
Integrate Input at Specified Interval, 515  
Interconnects, 43  
Interrupt Disable, 343  
Interrupt Enable, 347  
Interrupt Handling, 37  
Interrupt Module Instruction, 357  
Interrupt Timer, 365  
ISA Non Interacting PI, 551  
ITMR, 365  
ITOF, 371

## J

JSR, 373  
Jump to Subroutine, 373

## L

LAB, 375  
Label for a Subroutine, 375  
Limiter for the Pv, 525  
LL984  
    AD16, 55  
    ADD, 57  
    AND, 59  
    BCD, 63  
    BLKM, 65  
    BLKT, 69  
    BMDI, 73  
    BROT, 75  
    CHS, 79  
    CKSM, 85  
    Closed Loop Control / Analog Values, 15  
    CMPR, 89  
    Coils, Contacts and Interconnects, 43  
    COMP, 93  
    DCTR, 97  
    DIOH, 99  
    DIV, 103  
    DLOG, 107  
    DRUM, 113  
    DV16, 117  
    EMTH, 121  
    EMTH-ADDDP, 127  
    EMTH-ADDFP, 131  
    EMTH-ADDIF, 135  
    EMTH-ANLOG, 139  
    EMTH-ARCOS, 143  
    EMTH-ARSIN, 147  
    EMTH-ARTAN, 151  
    EMTH-CHSIN, 155  
    EMTH-CMPFP, 159  
    EMTH-CMPIF, 163  
    EMTH-CNVDR, 167  
    EMTH-CNVFI, 171  
    EMTH-CNVIF, 175  
    EMTH-CNVRD, 179  
    EMTH-COS, 183  
    EMTH-DIVDP, 187  
    EMTH-DIVFI, 191  
    EMTH-DIVFP, 195  
    EMTH-DIVIF, 199



---

EMTH-ERLOG, 203  
EMTH-EXP, 207  
EMTH-LNFP, 211  
EMTH-LOG, 215  
EMTH-LOGFP, 219  
EMTH-MULDP, 223  
EMTH-MULFP, 227  
EMTH-MULIF, 231  
EMTH-PI, 235  
EMTH-POW, 239  
EMTH-SINE, 243  
EMTH-SQRFP, 247  
EMTH-SQRT, 251  
EMTH-SQRTP, 255  
EMTH-SUBDP, 259  
EMTH-SUBFI, 263  
EMTH-SUBFP, 267  
EMTH-SUBIF, 271  
EMTH-TAN, 275  
ESI, 279  
EUCA, 297  
FIN, 309  
Formatting Messages for ASCII READ/  
WRIT Operations, 29  
FOUT, 313  
FTOI, 317  
HLTH, 319  
IBKR, 333  
IBKW, 335  
ICMP, 337  
ID, 343  
IE, 347  
IMIO, 351  
IMOD, 357  
Interrupt Handling, 37  
ITMR, 365  
ITOF, 371  
JSR, 373  
LAB, 375  
LOAD, 379  
MAP 3, 383  
MBIT, 391  
MBUS, 395  
MRTM, 405  
MSTR, 411  
MU16, 453  
MUL, 455  
NBIT, 457  
NCBT, 459  
NOBT, 461  
NOL, 463  
OR, 467  
PCFL, 471  
PCFL-AIN, 479  
PCFL-ALARM, 485  
PCFL-AOUT, 489  
PCFL-AVER, 493  
PCFL-CALC, 499  
PCFL-DELAY, 503  
PCFL-EQN, 509  
PCFL-INTEG, 515  
PCFL-KPID, 519  
PCFL-LIMIT, 525  
PCFL-LIMV, 529  
PCFL-LKUP, 533  
PCFL-LLAG, 537  
PCFL-MODE, 541  
PCFL-ONOFF, 545  
PCFL-PI, 551  
PCFL-PID, 555  
PCFL-RAMP, 561  
PCFL-RATE, 567  
PCFL-RATIO, 571  
PCFL-RMPLN, 577  
PCFL-SEL, 581  
PCFL-TOTAL, 585  
PEER, 591  
PID2, 595  
R --> T, 609  
RBIT, 613  
READ, 615  
RET, 621  
SAVE, 623  
SBIT, 627  
SCIF, 629  
SENS, 635  
SKPC, 639  
SKPR, 643  
SRCH, 647  
STAT, 651  
SU16, 675  
SUB, 677

Subroutine Handling, 39  
 T.01 Timer, 687  
 T-->R, 679  
 T-->T, 683  
 T0.1 Timer, 689  
 T1.0 Timer, 691  
 T1MS Timer, 693  
 TBLK, 699  
 TEST, 703  
 UCTR, 705  
 WRIT, 707  
 XMIT, 713  
 XMRD, 723  
 XMWT, 727  
 XOR, 731  
 LOAD, 379  
 Load Flash, 379  
 Load the Floating Point Value of "Pi", 235  
 Loadable DX  
   CHS, 79  
   DRUM, 113  
   ESI, 279  
   EUCA, 297  
   HLTH, 319  
   ICMP, 337  
   Installation, 41  
   MAP 3, 383  
   MBUS, 395  
   MRTM, 405  
   NOL, 463  
   PEER, 591  
   XMIT, 713  
 Logarithmic Ramp to Set Point, 577  
 Logical And, 59  
 Logical OR, 467  
 Look-up Table, 533

## M

MAP 3, 383  
 MAP Transaction, 383  
 Master, 411

Math  
   AD16, 55  
   ADD, 57  
   BCD, 63  
   DIV, 103  
   DV16, 117  
   FTOI, 317  
   ITOF, 371  
   MU16, 453  
   MUL, 455  
   SU16, 675  
   SUB, 677  
   TEST, 703  
 Matrix  
   AND, 59  
   BROT, 75  
   CMPR, 89  
   COMP, 93  
   MBIT, 391  
   NBIT, 457  
   NCBT, 459, 461  
   OR, 467  
   RBIT, 613  
   SBIT, 627  
   SENS, 635  
   XOR, 731  
 MBIT, 391  
 MBUS, 395  
 MBUS Transaction, 395  
 Miscellaneous  
   CKSM, 85  
   DLOG, 107  
   EMTH, 121  
   EMTH-ADDDP, 127  
   EMTH-ADDFP, 131  
   EMTH-ADDIF, 135  
   EMTH-ANLOG, 139  
   EMTH-ARCOS, 143, 183  
   EMTH-ARSIN, 147  
   EMTH-ARTAN, 151  
   EMTH-CHSIN, 155  
   EMTH-CMPFP, 159  
   EMTH-CMPIF, 163  
   EMTH-CNVDR, 167  
   EMTH-CNVFI, 171

- EMTH-CNVIF, 175
  - EMTH-CNVRD, 179
  - EMTH-DIVDP, 187
  - EMTH-DIVFI, 191
  - EMTH-DIVFP, 195
  - EMTH-DIVIF, 199
  - EMTH-ERLOG, 203
  - EMTH-EXP, 207
  - EMTH-LNFP, 211
  - EMTH-LOG, 215
  - EMTH-LOGFP, 219
  - EMTH-MULDP, 223
  - EMTH-MULFP, 227
  - EMTH-MULIF, 231
  - EMTH-PI, 235
  - EMTH-POW, 239
  - EMTH-SINE, 243
  - EMTH-SQRFP, 247
  - EMTH-SQRT, 251
  - EMTH-SQRTP, 255
  - EMTH-SUBDP, 259
  - EMTH-SUBFI, 263
  - EMTH-SUBFP, 267
  - EMTH-SUBIF, 271
  - EMTH-TAN, 275
  - LOAD, 379
  - MSTR, 411
  - SAVE, 623
  - SCIF, 629
  - XMRD, 723
  - XMWT, 727
  - Modbus Plus
    - MSTR, 411
  - Modbus Plus Network Statistics
    - MSTR, 440
  - Modify Bit, 391
  - Move
    - BLKM, 65
    - BLKT, 69
    - FIN, 309
    - FOUT, 313
    - IBKR, 333
    - IBKW, 335
    - R --> T, 609
    - SRCH, 647
    - T-->R, 679
    - T-->T, 683
    - TBLK, 699
  - MRTM, 405
  - MSTR, 411
    - Clear Local Statistics, 425
    - Clear Remote Statistics, 430
    - CTE Error Codes for SY/MAX and TCP/IP Ethernet, 452
    - Get Local Statistics, 423
    - Get Remote Statistics, 429
    - Modbus Plus and SY/MAX Ethernet Error Codes, 446
    - Modbus Plus Network Statistics, 440
    - Peer Cop Health, 432
    - Read CTE (Config Extension Table), 435
    - Read Global Data, 428
    - Reset Option Module, 434
    - SY/MAX-specific Error Codes, 448
    - TCP/IP Ethernet Error Codes, 450
    - TCP/IP Ethernet Statistics, 445
    - Write CTE (Config Extension Table), 437
    - Write Global Data, 427
  - MU16, 453
  - MUL, 455
  - Multiply, 455
  - Multiply 16 Bit, 453
  - Multi-Register Transfer Module, 405
- ## N
- NBIT, 457
  - NCBT, 459
  - Network Option Module for Lonworks, 463
  - NOBT, 461
  - NOL, 463
  - Normally Closed Bit, 459
  - Normally Open Bit, 461
- ## O
- ON/OFF Values for Deadband, 545
  - One Hundredth Second Timer, 687
  - One Millisecond Timer, 693
  - One Second Timer, 691
  - One Tenth Second Timer, 689
  - OR, 467

**P**

- PCFL, 471
- PCFL Subfunctions
  - General, 17
- PCFL-AIN, 479
- PCFL-ALARM, 485
- PCFL-AOUT, 489
- PCFL-AVER, 493
- PCFL-CALC, 499
- PCFL-DELAY, 503
- PCFL-EQN, 509
- PCFL-INTEG, 515
- PCFL-KPID, 519
- PCFL-LIMIT, 525
- PCFL-LIMV, 529
- PCFL-LKUP, 533
- PCFL-LLAG, 537
- PCFL-MODE, 541
- PCFL-ONOFF, 545
- PCFL-PI, 551
- PCFL-PID, 555
- PCFL-RAMP, 561
- PCFL-RATE, 567
- PCFL-RATIO, 571
- PCFL-RMPLN, 577
- PCFL-SEL, 581
- PCFL-Subfunction
  - PCFL-AIN, 479
  - PCFL-ALARM, 485
  - PCFL-AOUT, 489
  - PCFL-AVER, 493
  - PCFL-CALC, 499
  - PCFL-DELAY, 503
  - PCFL-EQN, 509
  - PCFL-INTEG, 515
  - PCFL-KPID, 519
  - PCFL-LIMIT, 525
  - PCFL-LIMV, 529
  - PCFL-LKUP, 533
  - PCFL-LLAG, 537
  - PCFL-MODE, 541
  - PCFL-ONOFF, 545
  - PCFL-PI, 551
  - PCFL-PID, 555
  - PCFL-RAMP, 561
  - PCFL-RATE, 567
  - PCFL-RATIO, 571
  - PCFL-RMPLN, 577
  - PCFL-SEL, 581
  - PCFL-TOTAL, 585
- PCFL-TOTAL, 585
- PEER, 591
- PEER Transaction, 591
- PID Algorithms, 555
- PID Example, 21
- PID2, 595
- PID2 Level Control Example, 25
- Process Control Function Library, 471
- Process Square Root, 255
- Process Variable, 16
- Proportional Integral Derivative, 595
- Put Input in Auto or Manual Mode, 541

**R**

- R --> T, 609
- Raising a Floating Point Number to an Integer Power, 239
- Ramp to Set Point at a Constant Rate, 561
- RBIT, 613
- READ, 615
  - MSTR, 421
- Read, 615
- READ/WRITE Operations, 29
- Register to Table, 609
- Regulatory Control, 472
- Reset Bit, 613
- RET, 621
- Return from a Subroutine, 621

**S**

- SAVE, 623
- Save Flash, 623
- SBIT, 627
- SCIF, 629
- Search, 647
- SENS, 635
- Sense, 635
- Sequential Control Interfaces, 629
- Set Bit, 627

Set Point Variable, 16  
Skip (Constants), 639  
Skip (Registers), 643  
Skips / Specials  
    RET, 621  
    SKPC, 639  
    SKPR, 643  
Skips/Specials  
    JSR, 373  
    LAB, 375  
SKPC, 639  
SKPR, 643  
Special  
    DIOH, 99  
    PCFL, 471  
    PCFL-, 489  
    PCFL-AIN, 479  
    PCFL-ALARM, 485  
    PCFL-AVER, 493  
    PCFL-CALC, 499  
    PCFL-DELAY, 503  
    PCFL-EQN, 509  
    PCFL-KPID, 519  
    PCFL-LIMIT, 525  
    PCFL-LIMV, 529  
    PCFL-LKUP, 533  
    PCFL-LLAG, 537  
    PCFL-MODE, 541  
    PCFL-ONOFF, 545  
    PCFL-PI, 551  
    PCFL-PID, 555  
    PCFL-RAMP, 561  
    PCFL-RATE, 567  
    PCFL-RATIO, 571  
    PCFL-RMPLN, 577  
    PCFL-SEL, 581  
    PCFL-TOTAL, 585  
    PCPCFL-INTEGFL, 515  
    PID2, 595  
    STAT, 651  
SRCH, 647  
STAT, 651  
Status, 651  
SU16, 675  
SUB, 677  
Subroutine Handling, 39

Subtract 16 Bit, 675  
Subtraction, 677  
Support of the ESI Module, 279

## T

T.01 Timer, 687  
T-->R, 679  
T-->T, 683  
T0.1 Timer, 689  
T1.0 Timer, 691  
T1MS Timer, 693  
Table to Block, 699  
Table to Register, 679  
Table to Table, 683  
TBLK, 699  
TCP/IP Ethernet Statistics  
    MSTR, 445  
TEST, 703  
Test of 2 Values, 703  
Time Delay Queue, 503  
Totalizer for Metering Flow, 585

## U

UCTR, 705  
Up Counter, 705

## V

Velocity Limiter for Changes in the Pv, 529

## W

WRIT, 707  
Write, 707  
    MSTR, 419

## X

XMIT, 713  
XMIT Communication Block, 713  
XMRD, 723  
XMWT, 727  
XOR, 731

Index

---