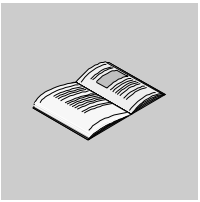
Concept
Block Library LL984

840 USE 506 00 eng Version 2.6

33002261.00

Table of Contents



About the book 11

Part I General Information 13
Introduction 13

Chapter 1 Instructions 15
Parameter Assignment of Instuctions 15

Chapter 2 Instruction Groups 17
At a Glance 17
Instruction Groups 18
ASCII Functions 19
Counters and Timers Instructions 19
Fast I/O Instructions 20
Loadable DX 21
Math Instructions 21
Matrix Instructions 23
Miscellaneous 24
Move Instructions 25
Skips/Specials 25
Special Instructions 26
Coils, Contacts and Interconnects 26

Chapter 3 Closed Loop Control / Analog Values 27
At a Glance 27
Closed Loop Control / Analog Values 28
PCFL Subfunctions 29
A PID Example 33
PID2 Level Control Example 37

Chapter 4 Formatting Messages for ASCII READ/WRIT Operations 41
At a Glance 41
Formatting Messages for ASCII READ/WRIT Operations 42
Format Specifiers 43
Special Set-up Considerations for Control/Monitor Signals Format 46

Chapter 5 Interrupt Handling 49
Interrupt Handling 49

Chapter 6 Subroutine Handling 51
Subroutine Handling 51

Chapter 7 Installation of DX Loadables 53
Installation of DX Loadables 53

Chapter 8 Coils, Contacts and Interconnects 55
At a Glance 55
Coils 56
Contacts 58
Interconnects (Shorts) 60

Part II Instruction Descriptions 61
At a Glance 61

Chapter 9 AD16: Ad 16 Bit 67
At a Glance 67
Short Description 68
Representation 68

Chapter 10 ADD: Addition 69

Chapter 11 AND: Logical And 71

Chapter 12 BCD: Binary to Binary Code 75

Chapter 13 BLKM: Block Move 77

Chapter 14 BLKT: Block to Table 81

Chapter 15 BMDI: Block Move with Interrupts Disabled 85

Chapter 16 BROT: Bit Rotate 87

Chapter 17 CHS: Configure Hot Standby 91

Chapter 18 CKSM: Check Sum 97

Chapter 19 CMPR: Compare Register 101

Chapter 20 COMP: Complement a Matrix 105

Chapter 21 DCTR: Down Counter 109

Chapter 22 DIOH: Distributed I/O Health 111

Chapter 23 DIV: Divide 115

Chapter 24 DLOG: Data Logging for PCMCIA Read/Write Support. 119
Chapter 25 DRUM: DRUM Sequencer. 125
Chapter 26 DV16: Divide 16 Bit 129
Chapter 27 EMTH: Extended Math 133
Chapter 28 EMTH-ADDDP: Double Precision Addition 139
Chapter 29 EMTH-ADDFP: Floating Point Addition 143
Chapter 30 EMTH-ADDIF: Integer + Floating Point Addition 147
Chapter 31 EMTH-ANLOG: Base 10 Antilogarithm 151
Chapter 32 EMTH-ARCOS: Floating Point Arc Cosine of an Angle (in Radians) 155
Chapter 33 EMTH-ARSIN: Floating Point Arcsine of an Angle (in Radians) 159
Chapter 34 EMTH-ARTAN: Floating Point Arc Tangent of an Angle (in Radians) 163
Chapter 35 EMTH-CHSIN: Changing the Sign of a Floating Point Number 167
Chapter 36 EMTH-CMPFP: Floating Point Comparison 171
Chapter 37 EMTH-CMPIF: Integer-Floating Point Comparison 175
Chapter 38 EMTH-CNVDR: Floating Point Conversion of Degrees to Radians 179
Chapter 39 EMTH-CNVFI: Floating Point to Integer Conversion 183
Chapter 40 EMTH-CNVIF: Integer-to-Floating Point Conversion. 187
Chapter 41 EMTH-CNVRD: Floating Point Conversion of Radians to Degrees 191
Chapter 42 EMTH-COS: Floating Point Cosine of an Angle (in Radians) 195
Chapter 43 EMTH-DIVDP: Double Precision Division. 199
Chapter 44 EMTH-DIVFI: Floating Point Divided by Integer. 203
Chapter 45 EMTH-DIVFP: Floating Point Division. 207

Chapter 46 EMTH-DIVIF: Integer Divided by Floating Point 211
Chapter 47 EMTH-ERLOG: Floating Point Error Report Log. 215
Chapter 48 EMTH-EXP: Floating Point Exponential Function. 219
Chapter 49 EMTH-LNFP: Floating Point Natural Logarithm 223
Chapter 50 EMTH-LOG: Base 10 Logarithm 227
Chapter 51 EMTH-LOGFP: Floating Point Common Logarithm 231
Chapter 52 EMTH-MULDP: Double Precision Multiplication. 235
Chapter 53 EMTH-MULFP: Floating Point Multiplication. 239
Chapter 54 EMTH-MULIF: Integer x Floating Point Multiplication 243
Chapter 55 EMTH-PI: Load the Floating Point Value of "Pi" 247
Chapter 56 EMTH-POW: Raising a Floating Point Number to an Integer Power 251
Chapter 57 EMTH-SINE: Floating Point Sine of an Angle (in Radians) . 255
Chapter 58 EMTH-SQRFP: Floating Point Square Root. 259
Chapter 59 EMTH-SQRT: Floating Point Square Root 263
Chapter 60 EMTH-SQRTP: Process Square Root. 267
Chapter 61 EMTH-SUBDP: Double Precision Subtraction 271
Chapter 62 EMTH-SUBFI: Floating Point - Integer Subtraction 275
Chapter 63 EMTH-SUBFP: Floating Point Subtraction 279
Chapter 64 EMTH-SUBIF: Integer - Floating Point Subtraction 283
Chapter 65 EMTH-TAN: Floating Point Tangent of an Angle (in Radians) 287
Chapter 66 ESI: Support of the ESI Module 291
Chapter 67 EUCA: Engineering Unit Conversion and Alarms 309
Chapter 68 FIN: First In. 321
Chapter 69 FOUT: First Out 325
Chapter 70 FTOI: Floating Point to Integer 329

Chapter 71 HLTH: History and Status Matrices. 331
Chapter 72 IBKR: Indirect Block Read 345
Chapter 73 IBKW: Indirect Block Write 347
Chapter 74 ICMP: Input Compare 349
Chapter 75 ID: Interrupt Disable 355
Chapter 76 IE: Interrupt Enable. 359
Chapter 77 IMIO: Immediate I/O 363
Chapter 78 IMOD: Interrupt Module Instruction 369
Chapter 79 ITMR: Interrupt Timer 377
Chapter 80 ITOF: Integer to Floating Point 383
Chapter 81 JSR: Jump to Subroutine. 385
Chapter 82 LAB: Label for a Subroutine 387
Chapter 83 LOAD: Load Flash 391
Chapter 84 MAP 3: MAP Transaction 395
Chapter 85 MBIT: Modify Bit 403
Chapter 86 MBUS: MBUS Transaction 407
Chapter 87 MRTM: Multi-Register Transfer Module 417
Chapter 88 MSTR: Master 423
Chapter 89 MU16: Multiply 16 Bit 465
Chapter 90 MUL: Multiply 467
Chapter 91 NBIT: Bit Control. 469
Chapter 92 NCBT: Normally Closed Bit 471
Chapter 93 NOBT: Normally Open Bit 473
Chapter 94 NOL: Network Option Module for Lonworks 475
Chapter 95 OR: Logical OR 479
Chapter 96 PCFL: Process Control Function Library 483

Chapter 97 PCFL-AIN: Analog Input 491
Chapter 98 PCFL-ALARM: Central Alarm Handler 497
Chapter 99 PCFL-AOUT: Analog Output 501
Chapter 100 PCFL-AVER: Average Weighted Inputs Calculate 505
Chapter 101 PCFL-CALC: Calculated preset formula 511
Chapter 102 PCFL-DELAY: Time Delay Queue 515
Chapter 103 PCFL-EQN: Formatted Equation Calculator 521
Chapter 104 PCFL-INTEG: Integrate Input at Specified Interval 527
Chapter 105 PCFL-KPID: Comprehensive ISA Non Interacting PID 531
Chapter 106 PCFL-LIMIT: Limiter for the Pv 537
Chapter 107 PCFL-LIMV: Velocity Limiter for Changes in the Pv 541
Chapter 108 PCFL-LKUP: Look-up Table 545
Chapter 109 PCFL-LLAG: First-order Lead/Lag Filter 549
Chapter 110 PCFL-MODE: Put Input in Auto or Manual Mode 553
Chapter 111 PCFL-ONOFF: ON/OFF Values for Deadband 557
Chapter 112 PCFL-PI: ISA Non Interacting PI 563
Chapter 113 PCFL-PID: PID Algorithms 567
Chapter 114 PCFL-RAMP: Ramp to Set Point at a Constant Rate 573
Chapter 115 PCFL-RATE: Derivative Rate Calculation over a Specified Timeme 579
Chapter 116 PCFL-RATIO: Four Station Ratio Controller 583
Chapter 117 PCFL-RMPLN: Logarithmic Ramp to Set Point 589
Chapter 118 PCFL-SEL: Input Selection 593
Chapter 119 PCFL-TOTAL: Totalizer for Metering Flow 597
Chapter 120 PEER: PEER Transaction 603
Chapter 121 PID2: Proportional Integral Derivative 607
Chapter 122 R —> T: Register to Table 621

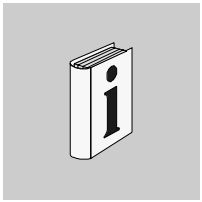
Chapter 123 RBIT: Reset Bit 625
Chapter 124 READ: Read. 627
Chapter 125 RET: Return from a Subroutine. 633
Chapter 126 SAVE: Save Flash 635
Chapter 127 SBIT: Set Bit 639
Chapter 128 SCIF: Sequential Control Interfaces 641
Chapter 129 SENS: Sense 647
Chapter 130 SKPC: Skip (Constants) 651
Chapter 131 SKPR: Skip (Registers) 655
Chapter 132 SRCH: Search 659
Chapter 133 STAT: Status 663
Chapter 134 SU16: Subtract 16 Bit 687
Chapter 135 SUB: Subtraction 689
Chapter 136 T–>R: Table to Register 691
Chapter 137 T–>T: Table to Table 695
Chapter 138 T.01 Timer: One Hundredth Second Timer. 699
Chapter 139 T0.1 Timer: One Tenth Second Timer 701
Chapter 140 T1.0 Timer: One Second Timer 703
Chapter 141 T1MS Timer: One Millisecond Timer. 705
Chapter 142 TBLK: Table to Block 711
Chapter 143 TEST: Test of 2 Values 715
Chapter 144 UCTR: Up Counter 717
Chapter 145 WRIT: Write 719
Chapter 146 XMIT: XMIT Communication Block 725
Chapter 147 XMRD: Extended Memory Read 735
Chapter 148 XMWT: Extended Memory Write 739

Chapter 149 XOR: Exclusive OR 743

Glossary 747

Index 771

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

General Information

I

Introduction

At a Glance

In this part you will find general information about the instruction groups and the use of instructions.

What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
1	Instructions	15
2	Instruction Groups	17
3	Closed Loop Control / Analog Values	27
4	Formatting Messages for ASCII READ/WRIT Operations	41
5	Interrupt Handling	49
6	Subroutine Handling	51
7	Installation of DX Loadables	53
8	Coils, Contacts and Interconnects	55

Instructions

1

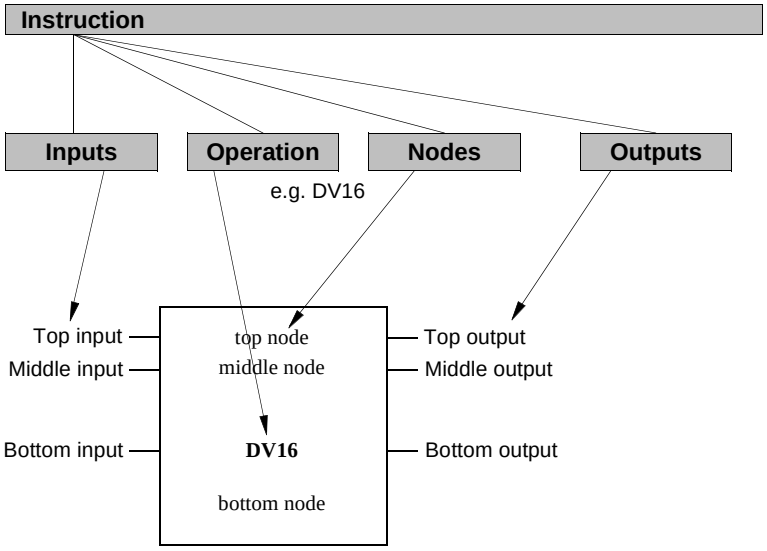
Parameter Assignment of Instuctions

General

Programming for electrical controls involves a user who implements Operational Coded instructions in the form of visual objects organized in a recognizable ladder form. The program objects designed, at the user level, is converted to computer usable OP codes during the download process. the Op codes are decoded in the CPU and acted upon by the controllers firmware functions to implement the desired control.
Each instruction is composed of an operation, nodes required for the operation and in- and outputs.

Parameter Assignment

Parameter assignment with the instruction DV16 as an example:



Operation

The operation determines which functionality is to be executed by the instruction, e.g. shift register, conversion operations.

Nodes, In- and Outputs

The nodes and in- and outputs determines what the operation will be executed with.

Instruction Groups

2

At a Glance

Introduction

In this chapter you will find an overwiew of the instruction groups and their accompanying instructions.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Instruction Groups	18
ASCII Functions	19
Counters and Timers Instructions	19
Fast I/O Instructions	20
Loadable DX	21
Math Instructions	21
Matrix Instructions	23
Miscellaneous	24
Move Instructions	25
Skips/Specials	25
Special Instructions	26
Coils, Contacts and Interconnects	26

Instruction Groups

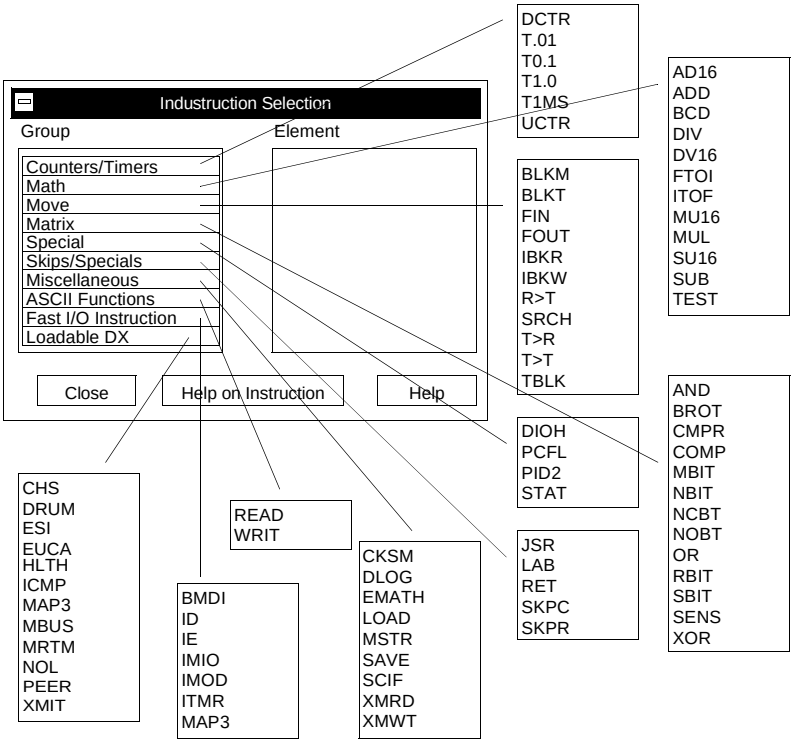
General

All instructions are attached to one of the following groups:

- ASCII Functions (See *ASCII Functions, p. 19*)
- Counters/Timers (See *Counters and Timers Instructions, p. 19*)
- Fast I/O Instructions (See *Fast I/O Instructions, p. 20*)
- Loadable DX (See *Loadable DX, p. 21*)
- Math (See *Math Instructions, p. 21*)
- Matrix (See *Matrix Instructions, p. 23*)
- Miscellaneous (See *Miscellaneous, p. 24*)
- Move (See *Move Instructions, p. 25*)
- Skips/Specials (See *Skips/Specials, p. 25*)
- Special (See *Special Instructions, p. 26*)
- *Coils, Contacts and Interconnects, p. 26*

Overview of all Instructions

Overwiew of instructions per instruction group



ASCII Functions

ASCII Functions

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
READ	Read ASCII messages	yes	no	no	no
WRIT	Write ASCII messages	yes	no	no	no

PLCs that support ASCII messaging use instructions called READ and WRIT to handle the sending of messages to display devices and the receiving of messages from input devices. These instructions provide the routines necessary for communication between the ASCII message table in the PLC's system memory and an interface module at the Remote I/O drops.
Further information you will find in the chapter *Formatting Messages for ASCII READ/WRIT Operations, p. 41*.

Counters and Timers Instructions

Counters and Timers Instructions

The table shows the counters and timers instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
UCTR	Counts up from 0 to a preset value	yes	yes	yes	no
DCTR	Counts down from a preset value to 0	yes	yes	yes	no
T1.0	Timer that increments in seconds	yes	yes	yes	no
T0.1	Timer that increments in tenths of a second	yes	yes	yes	no
T.01	Timer that increments in hundredths of a second	yes	yes	yes	no
T1MS	Timer that increments in one millisecond	yes (CPU 242 02 only)	yes	yes	no

Fast I/O Instructions

Fast I/O Instructions

The following instructions are designed for a variety of functions known generally as fast I/O updating:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
BMDI	Block move with interrupts disabled	yes	yes	no	no
ID	Disable interrupt	yes	yes	no	no
IE	Enable interrupt	yes	yes	no	no
IMIO	Immediate I/O instruction	yes	yes	no	no
IMOD	Interrupt module instruction	yes	no	no	no
ITMR	Interval timer interrupt	no	yes	no	no

Further information you will find in the chapter *Interrupt Handling, p. 49*.

Note: The Fast I/O Instructions are only available after configuring a CPU without extension.

Loadable DX

Loadable DX

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
CHS	Hot standby (Quantum)	yes	no	no	no
DRUM	DRUM sequenzer	yes	yes	no	no
ESI	Support of the ESI module 140 ESI 062 10	yes	no	no	no
EUCA	Engineering unit conversion and alarms	yes	yes	no	no
HLTH	History and status matrices	yes	yes	no	no
ICMP	Input comparison	yes	yes	no	no
MAP3	MAP 3 Transaction	no	no	no	no
MBUS	MBUS Transaction	no	no	no	no
MRTM	Multi-register transfer module	yes	yes	no	no
NOL	Transfer to/from the NOL Module	yes	no	no	no
PEER	PEER Transaction	no	no	no	no
XMIT	RS 232 Master Mode	yes	yes	yes	no

Further information you will find in *Installation of DX Loadables, p. 53*.

Math Instructions

Math Instructions

Two groups of instructions that support basic math operations are available. The first group comprises four integer-based instructions: ADD, SUB, MUL and DIV.

The second group contains five comparable instructions, AD16, SU16, TEST, MU16 and DV16, that support signed and unsigned 16-bit math calculations and comparisons.

Three additional instructions, ITOF, FTOI and BCD, are provided to convert the formats of numerical values (from integer to floating point, floating point to integer, binary to BCD and BCD to binary). Conversion operations are usful in expanded math.

Integer Based Instructions

This part of the group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
ADD	Addition	yes	yes	yes	no
DIV	Division	yes	yes	yes	no
MUL	Multiplication	yes	yes	yes	no
SUB	Subtraction	yes	yes	yes	no

Comparable Instructions

This part of the group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
AD16	Add 16 bit	yes	yes	yes	no
DV16	Divide 16 bit	yes	yes	yes	no
MU16	Multiply 16 bit	yes	yes	yes	no
SU16	Subtract 16 bit	yes	yes	yes	no
TEST	Test of 2 values	yes	yes	yes	no

Format Conversion

This part of the group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
BCD	Conversion from binary to binary code or binary code to binary	yes	yes	yes	no
FTOI	Conversion from floating point to integer	yes	yes	yes	no
ITOF	Conversion from integer to floating point	yes	yes	yes	no

Matrix Instructions

Matrix Instructions

A matrix is a sequence of data bits formed by consecutive 16-bit words or registers derived from tables. DX matrix functions operate on bit patterns within tables.

Just as with move instructions, the minimum table length is 1 and the maximum table length depends on the type of instruction you use and on the size of the CPU (24-bit) in your PLC.

Groups of 16 discretes can also be placed in tables. The reference number used is the first discrete in the group, and the other 15 are implied. The number of the first discrete must be of the first of 16 type 000001, 100001, 000017, 100017, 000033, 100033, ... , etc..

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
AND	Logical AND	yes	yes	yes	no
BROT	Bit rotate	yes	yes	yes	no
CMPR	Compare register	yes	yes	yes	no
COMP	Complement a matrix	yes	yes	yes	no
MBIT	Modify bit	yes	yes	yes	no
NBIT	Bit control	yes	yes	no	no
NCBT	Normally open bit	yes	yes	no	no
NOBT	Normally closed bit	yes	yes	no	no
OR	Logical OR	yes	yes	yes	no
RBIT	Reset bit	yes	yes	no	no
SBIT	Set bit	yes	yes	no	no
SENS	Sense	yes	yes	yes	no
XOR	Exclusive OR	yes	yes	yes	no

Miscellaneous

Miscellaneous

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
CKSM	Check sum	yes	yes	yes	no
DLOG	Data Logging for PCMCIA Read/Write Support	no	yes	no	no
EMTH	Extended Math Functions	yes	yes	yes	no
LOAD	Load flash	yes (CPU 434 12/ 534 14 only)	yes	yes (CCC 960 x0/ 980 x0 only)	no
MSTR	Master	yes	yes	yes	no
SAVE	Save flash	yes (CPU 434 12/ 534 14 only)	yes	yes (CCC 960 x0/ 980 x0 only)	no
SCIF	Sequential control interfaces	yes	yes	no	no
XMRD	Extended memory read	yes	no	no	no
XMWT	Extended memory write	yes	no	no	no

Move Instructions

Move Instructions

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
BLKM	Block move	yes	yes	yes	no
BLKT	Table to block move	yes	yes	yes	no
FIN	First in	yes	yes	yes	no
FOUT	First out	yes	yes	yes	no
IBKR	Indirect block read	yes	yes	no	no
IBKW	Indirect block write	yes	yes	no	no
R → T	Register to tabel move	yes	yes	yes	no
SRCH	Search table	yes	yes	yes	no
T → R	Table to register move	yes	yes	yes	no
T → T	Table to table move	yes	yes	yes	no
TBLK	Table to block move	yes	yes	yes	no

Skips/Specials

Skips/Specials

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
JSR	Jump to subroutine	yes	yes	yes	no
LAB	Label for a subroutine	yes	yes	yes	no
RET	Return from a subroutine	yes	yes	yes	no
SKPC	Skip (constant)	yes	yes	yes	no
SKPR	Skip (register)	yes	yes	yes	no

The SKP instruction is a standard instruction in all PLCs. It should be used with caution.

	DANGER
	Inputs and outputs that normally effect control may be unintentionally skipped (or not skipped). SKP is a dangerous instruction that should be used carefully. If inputs and outputs that normally effect control are unintentionally skipped (or not skipped), the result can create hazardous conditions for personnel and application equipment. **Failure to observe this precaution will result in death or serious injury.**

Special Instructions

Special Instructions

These instructions are used in special situations to measure statistical events on the overall logic system or create special loop control situations.

This group provides the following instructions:

Instruction	Meaning	Available at PLC family			
		Quantum	Compact	Momentum	Atrium
DIOH	Distributed I/O health	yes	no	no	no
PCFL	Process control function library	yes	yes	no	no
PID2	Proportional integral derivative	yes	yes	yes	no
STAT	Status	yes	yes	yes	no

Coils, Contacts and Interconnects

Coils, Contacts and Interconnects

Coils, Contacts and Interconnects are availabel at all PLC families:

- Normal coil
- Memory-retentive, or latched, coil
- Normally open (N.O.) contact
- Normally closed (N.C.) contact
- Positive transitional (P.T.) contact
- Negative transitional (N.T.) contact
- Horizontal Short
- Vertical Short

Closed Loop Control / Analog Values

3

At a Glance

Introduction

In this chapter you will find general information about configuring closed loop control and using analog values.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Closed Loop Control / Analog Values	28
PCFL Subfunctions	29
A PID Example	33
PID2 Level Control Example	37

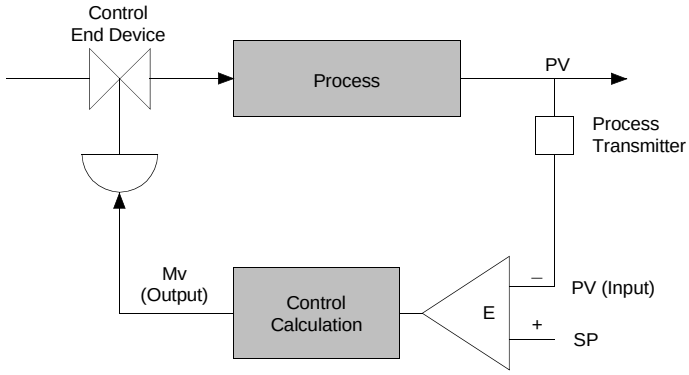
Closed Loop Control / Analog Values

General

An analog closed loop control system is one in which the deviation from an ideal process condition is measured, analyzed and adjusted in an attempt to obtain and maintain zero error in the process condition. Provided with the Enhanced Instruction Set is a proportional-integral-derivative function block called PID2, which allows you to establish closed loop (or negative feedback) control in ladder logic.

Definition of Set Point and Process Variable

The desired (zero error) control point, which you will define in the PID2 block, is called the set point (SP). The conditional measurement taken against SP is called the process variable (PV). The difference between the SP and the PV is the deviation or error (E). E is fed into a control calculation that produces a manipulated variable (Mv) used to adjust the process so that PV = SP (and, therefore, E = 0).



PCFL Subfunctions

General

The PCFL instruction gives you access to a library of process control functions utilizing analog values.
PCFL operations fall into three major categories:

- Advanced Calculations
- Signal Processing
- Regulatory Control

Advanced Calculations

Advanced calculations are used for general mathematical purposes and are not limited to process control applications. With advanced calculations, you can create custom signal processing algorithms, derive states of the controlled process, derive statistical measures of the process, etc.
Simple math routines have already been offered in the EMTH instruction. The calculation capability included in PCFL is a textual equation calculator for writing custom equations instead of programming a series of math operations one by one.

Signal Processing

Signal processing functions are used to manipulate process and derived process signals. They can do this in a variety of ways; they linearize, filter, delay and otherwise modify a signal. This category would include functions such as an Analog Input/Output, Limiters, Lead/Lag and Ramp generators.

Regulatory Control

Regulatory functions perform closed loop control in a variety of applications. Typically, this is a PID (proportional integral derivative) negative feedback control loop. The PID functions in PCFL offer varying degrees of functionality. Function PID has the same general functionality as the PID2 instruction but uses floating point math and represents some options differently. PID is beneficial in cases where PID2 is not suitable because of numerical concerns such as round-off.

Explanation of Formula Elements

Meaning of formula elements in the following formulas:

Formula elements	Meaning
Y	Manipulated variable output
YP	Proportional part of the calculation
YI	Integral part of the calculation
YD	Derivative part of the calculation
Bias	Constant added to input
BT	Bumpless transfer register
SP	Set point
KP	Proportional gain
Dt	Time since last solve
TI	Integral time constant
TD	Derivative time constant
TD1	Derivative time lag
XD	Error term, deviation
XD_1	Previous error term
X	Process input
X_1	Previous process input

General Equations

The following general equations are valid:

Equation	Condition /Requirement
$Y = YP + YI + YD + BIAS$	Integral bit ON
$Y = YP + YD + BIAS + BT$	Integral bit OFF
$Y_{high} \leq Y \leq Y_{low}$	High/low limits
with	
$YP, YI, YD = f(XD)$	
$XD = SP - X \pm (GRZ \times (1 - KGRZ))$	Gain reduction
$XD = SP - X$	Gain reduction zone not used

Proportional Calculations

The following equations are valid:

Equation	Condition /Requirement
$YP = KP \times XD$	Proportional bit ON
$YP = 0$	

Integral Calculation

The following equations are valid:

Equation	Condition /Requirement
$YI = YI + KP \times \frac{\Delta t}{TI} \times \frac{XD_1 + XD}{2}$	Integral bit ON
$YI = 0$	

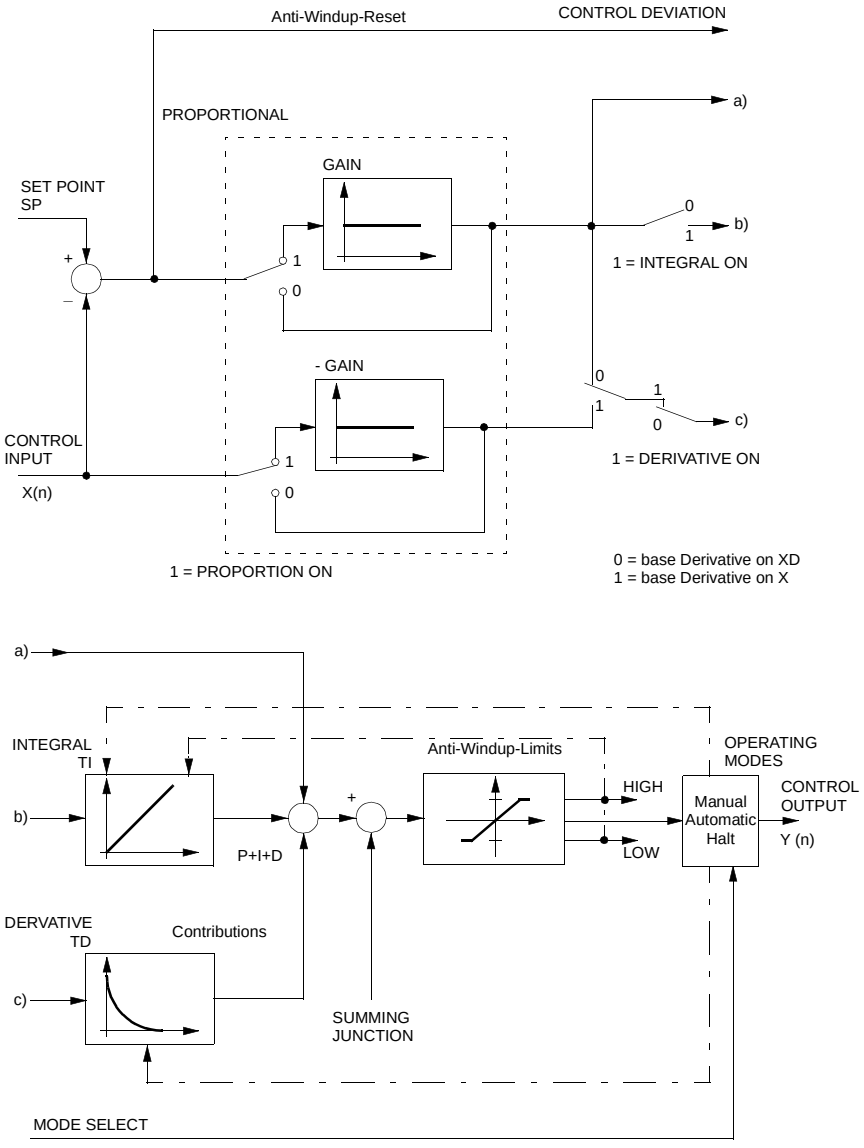
Derivative Calculation

The following equations are valid:

Equation	Condition /Requirement
$DXD = X_1 - X$	Base derivative or PV
$DXD = XD - X_1$	
$YD = \frac{(TD1 \times YD) + (TD \times KP \times DXD)}{\Delta t + TD1}$	Derivative bit ON
$D = 0$	

Structure Diagram

Structure Diagram

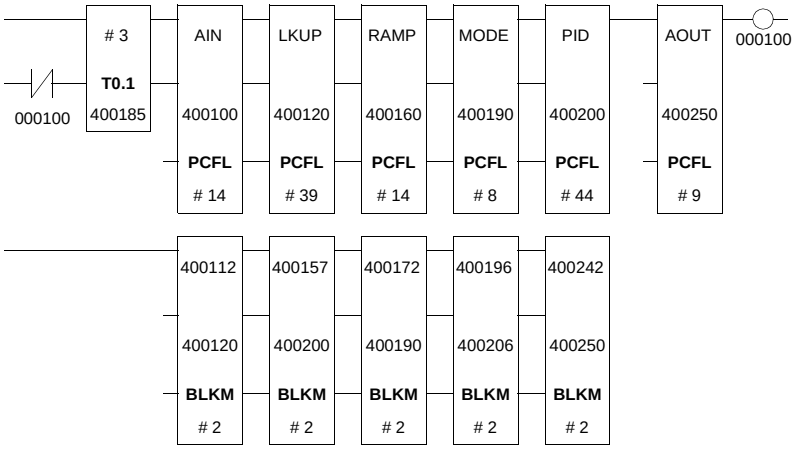


A PID Example

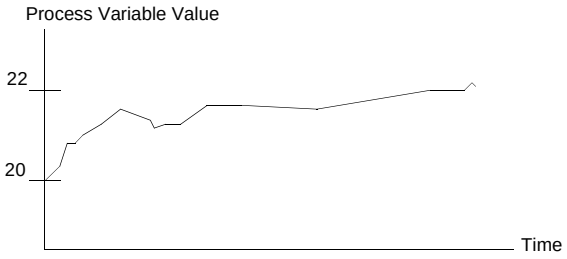
Description

This example illustrates how a typical PID loop could be configured using PCFL function PID. The calculation begins with the AIN function, which takes raw input simulated to cause the output to run between approximately 20 and 22 when the engineering unit scale is set to 0 ... 100.

LL984 Ladder Diagram

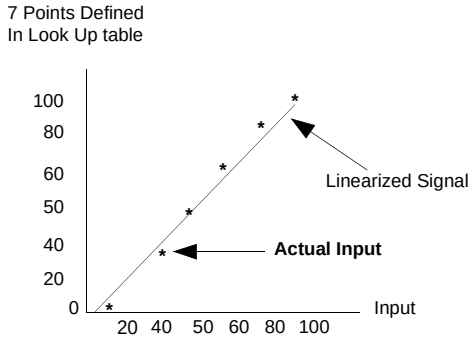


The process variable over time should look something like this:



Main PID Ladder Logic

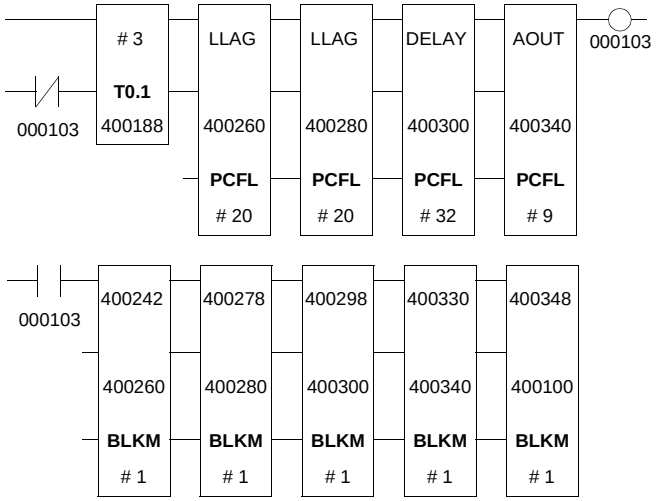
The AIN output is block moved to the LKUP function, which is used to scale the input signal. We do this because the input sensor is not likely to produce highly linear readings; the result is an ideal linear signal:



The look-up table output is block moved to the PID function. RAMP is used to control the rise (or fall) of the set point for the PID controller with regard to the rate of ramp and the solution interval. In this example, the set point is established in another logic section to simulate a remote setting. The MODE function is placed after the RAMP so that we can switch between the RAMP-generated set point or a manual value.

Simulated Process

The PID function is actually controlling the process simulated by this logic (value in 400100: **878**(Dec)):



The process simulator is comprised of two LLAG functions that act as a filter and input to a DELAY queue that is also a PCFL function block. This arrangement is the equivalent of a second-order process with dead time.

The solution intervals for the LLAG filters do not affect the process dynamics and were chosen to give fast updates. The solution interval for the DELAY queue is set at 1000 ms with a delay of 5 intervals,i.e. 5 s. The LLAG filters each have lead terms of 4 s and lag terms of 10 s. The gain for each is 1.0.

In process control terms the transfer function can be expressed as:

$$Gp(S) = \frac{(4S + 1)(4S + 1)e^{-5S}}{(10S + 1)(10S + 1)}$$

The AOUT function is used only to convert the simulated process output control value into a range of 0 ... 4 095, which simulates a field device. This integer signal is used as the process input in the first network.

PID Parameters

The PID controller is tuned to control this process at 20.0, using the Ziegler-Nichols tuning method. The resulting controller gain is 2.16, equivalent to a proportional band of 46.3%.

The integral time is set at 12.5 s/repeat (4.8 repeats/ min). The derivative time is initially 3 s, then reduced to 0.3 s to de-emphasize the derivative effect.
An AOUT function is used after the PID. It conditions the PID control output by scaling the signal back to an integer for use as the control value.

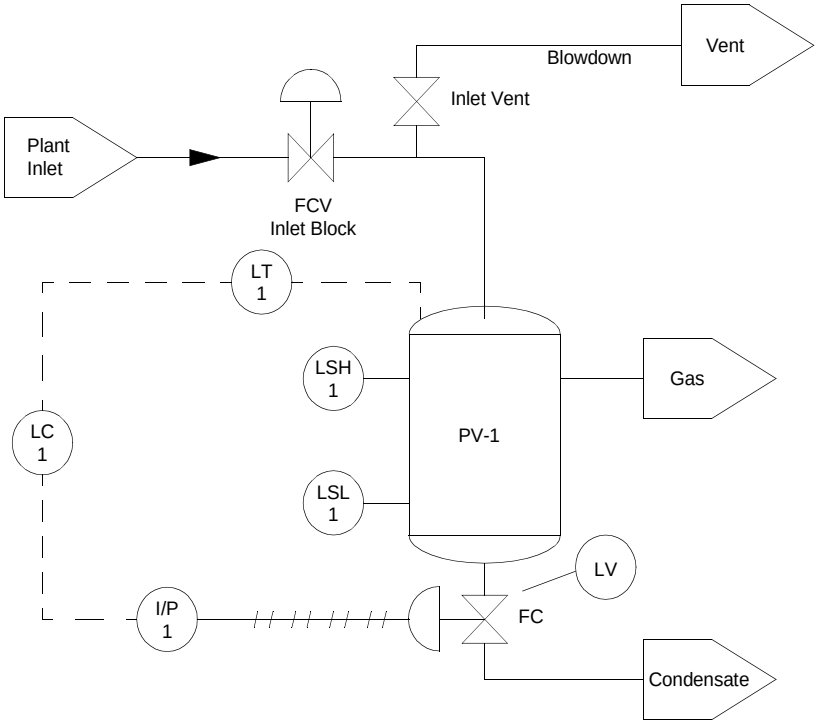
The entire control loop is preceded by a 0.1 s timer. The target solution interval for the entire loop is 1 s, and the full solve is 1 s. However, the nontime-dependent functions that are used (AIN, LKUP, MODE, and AOUT) do not need to be solved every scan. To reduce the scan time impact, these functions are scheduled to solve less frequently. The example has a loop solve every 3 s, reducing the average scan time dramatically.

Note: It is still important to be aware of the maximum scan impact. When programming other loops, you will not want all of the loops to solve on the same scan

PID2 Level Control Example

Description

Here is a simplified P&I diagram for an inlet separator in a gas processing plant. There is a two-phase inlet stream: liquid and gas.



LT-1 4 ... 20 mA level transmitter

I/P-1 4 ... 20 mA current to pneumatic converter

LV-1 control valve, fail CLOSED

LSH-1 high level switch, normally closed

LSL-1 low level switch, normally open

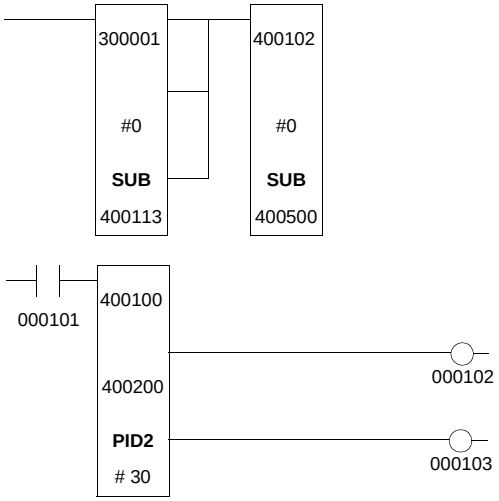
LC-1 level controller

I/P-1 Mv to control the flow into tank T-1

The liquid is dumped from the tank to maintain a constant level. The control objective is to maintain a constant level in the separator. The phases must be separated before processing; separation is the role of the inlet separator, PV-1. If the level controller, LC-1, fails to perform its job, the inlet separator could fill, causing liquids to get into the gas stream; this could severely damage devices such as gas compressors.

Ladder Logic Diagram

The level is controlled by device LC-1, a Quantum controller connected to an analog input module; I/P-1 is connected to an analog output module. We can implement the control loop with the following 984 ladder logic:



The first SUB block is used to move the analog input from LT-1 to the PID2 analog input register, 40113. The second SUB block is used to move the PID2 output Mv to the I/O mapped output I/P-1. Coil 00101 is used to change the loop from AUTO to MANUAL mode, if desired. For AUTO mode, it should be ON.

Register Content

Specify the set point in mm for input scaling (E.U.). The full input range will be 0 ... 4000 mm (for 0 ... 4095 raw analog). Specify the register content of the top node in the PID2 block as follows:

Register	Content Numeric	Content Meaning	Comments
400100		Scaled PV (mm)	PID2 writes this
400101	2000	Scaled SP (mm)	Set to 2000 mm (half full) initially
400102	0000	Loop output (0 ... 4095	PID2 writes this; keep it set to 0 to be safe
400103	3500	Alarm High Set Point (mm)	If the level rises above 3500 mm, coil 000102 goes ON
400104	1000	Alarm Low Set Point (mm)	If the level drops below 1000 mm, coil 000103 goes ON
400105	0100	PB (%)	The actual value depends on the process dynamics
400106	0500	Integral constant (5.00 repeats/min)	The actual value depends on the process dynamics
400107	0000	Rate time constant (per min)	Setting this to 0 turns off the derivative mode
400108	0000	Bias (0 ... 4095)	This is set to 0, since we have an integral term
400109	4095	High windup limit (0 ... 4095)	Normally set to the maximum
400110	0000	Low windup limit (0 ... 4095)	Normally set to the minimum
400111	4000	High engineering range (mm)	The scaled value of the process variable when the raw input is at 4095
400112	0000	Low engineering range (mm)	The scaled value of the process variable when the raw input is at 0
400113		Raw analog measure (0 ... 4095)	A copy of the input from the analog input module register (300001) copied by the first SUB
400114	0000	Offset to loop counter register	Zero disables this feature. Normally, this is not used
400115	0000	Max loops solved per scan	See register 400114

Register	Content Numeric	Content Meaning	Comments
400116	0102	Pointer to reset feedback	If you leave this as zero, the PID2 function automatically supplies a pointer to the loop output register. If the actual output (400500) could be changed from the value supplied by PID2, then this register should be set to 500 (400500) to calculate the integral properly
400117	4095	Output clamp high (0 ... 4095)	Normally set to maximum
400118	0000	Output clamp low (0 ... 4095)	Normally set to minimum
400119	0015	Rate Gain Limit Constant (2 ... 30)	Normally set to about 15. The actual value depends on how noisy the input signal is. Since we are not using derivative mode, this has no effect on PID2
400120	0000	Pointer to track input	Used only if the PRELOAD feature is used. If the PRELOAD is not used, this is normally zero

The values in the registers in the 400200 destination block are all set by the PID2 block.

Formatting Messages for ASCII READ/WRIT Operations

4

At a Glance

Introduction

In this chapter you will find general information about formatting messages for ASCII READ/WRIT operations.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Formatting Messages for ASCII READ/WRIT Operations	42
Format Specifiers	43
Special Set-up Considerations for Control/Monitor Signals Format	46

Formatting Messages for ASCII READ/WRIT Operations

General

The ASCII messages used in the READ and WRIT instructions can be created via your panel software using the format specifiers described below. Format specifiers are character symbols that indicate:

- The ASCII characters used in the message
- Register content displayed in ASCII character format
- Register content displayed in hexadecimal format
- Register content displayed in integer format
- Subroutine calls to execute other message formats

Overview Format Specifiers

The following format specifiers can be used;

Specifier	Meaning
/	ASCII return (CR) and linefeed (LF)
" "	Enclosure for octal control code
' '	Enclosure for ASCII text characters
X	Space indicator
( )	Repeat contents of the parentheses
I	Integer
L	Leading zeros
A	Alphanumeric
O	Octal
B	Binary
H	Hexadecimal

Format Specifiers

Format Specifier /

ASCII return (CR) and linefeed (LF)

Field width	None (defaults to 1)
Prefix	None (defaults to 1)
Input format	Outputs CR, LF; no ASCII characters accepted
Output format	Outputs CR, LF

Format Specifier " "

Enclosure for octal control code

Field width	Three digits enclosed in double quotes
Prefix	None
Input format	Accepts three octal control characters
Output format	Outputs three octal control characters

Format Specifier / ´

Enclosure for ASCII text characters

Field width	1 ... 128 characters
Prefix	None (defaults to 1)
Input format	Inputs number of upper and/or lower case printable characters specified by the field width
Output format	Outputs number of upper and/or lower case printable characters specified by the field width

Format Specifier X

Space indicator, e.g., 14X indicates 14 spaces left open from the point where the specifier occurs.

Field width	None (defaults to 1)
Prefix	1 ... 99 spaces
Input format	Inputs specified number of spaces
Output format	Outputs specified number of spaces

Format Specifier ( )

Repeat contents of the parentheses, e.g., `2 (4X, I5)` says repeat 4X, I5 two times

Field width	None
Prefix	1 ... 255
Input format	Repeat format specifiers in parentheses the number of times specified by the prefix
Output format	Repeat format specifiers in parentheses the number of times specified by the prefix

Format Specifier I

Integer, e.g., `I5` specifies five integer characters

Field width	1 ... 8 characters
Prefix	1 ... 99
Input format	Accepts ASCII characters 0 ... 9. If the field width is not satisfied, the most significant characters in the field are padded with zeros
Output format	Outputs ASCII characters 0 ... 9. If the field width is not satisfied, the most significant characters in the field are padded with zeros. The overflow field consists of asterisks.

Format Specifier L

Leading zeros, e.g., `L5` specifies five leading zeros

Field width	1 ... 8 characters
Prefix	1 ... 99
Input format	Accepts ASCII characters 0 ... 9. If the field width is not satisfied, the most significant characters in the field are padded with zeros
Output format	Outputs ASCII characters 0 ... 9. If the field width is not satisfied, the most significant characters in the field are padded with zeros. The overflow field consists of asterisks.

Format Specifier A

Alphanumeric, e.g., `A27` specifies 27 alphanumeric characters, no suffix allowed

Field width	None (defaults to 1)
Prefix	1 ... 99
Input format	Accepts any 8-bit character except reserved delimiters such as CR, LF, ESC, BKSPC, DEL.
Output format	Outputs any 8-bit character

Format Specifier O

Octal, e.g., O2 specifies two octal characters

Field width	1 ... 6 characters
Prefix	1 ... 99
Input format	Accepts ASCII characters 0 ... 7. If the field width is not satisfied, the most significant characters are padded with zeros.
Output format	Outputs ASCII characters 0 ... 7. If the field width is not satisfied, the most significant characters are padded with zeros. No overflow indicators.

Format Specifier B

Binary, e.g., B4 specifies four binary characters

Field width	1 ... 16 characters
Prefix	1 ... 99
Input format	Accepts ASCII characters 0 and 1. If the field width is not satisfied, the most significant characters are padded with zeros.
Output format	Outputs ASCII characters 0 and 1. If the field width is not satisfied, the most significant characters are padded with zeros. No overflow indicators.

Format Specifier H

Hexadecimal, e.g., H2 specifies two hex characters

Field width	1 ... 4 characters
Prefix	1 ... 99
Input format	Accepts ASCII characters 0 ... 9 and A ... F. If the field width is not satisfied, the most significant characters are padded with zeros.
Output format	Outputs ASCII characters 0 ... 9 and A ... F. If the field width is not satisfied, the most significant characters are padded with zeros. No overflow indicators.

Special Set-up Considerations for Control/Monitor Signals Format

General

To control and monitor the signals used in the messaging communication, specify code 1002 in the first register of the control block (the register displayed in the top node). Via this format, you can control the RTS and CTS lines on the port used for messaging.

Note: In this format, only the local port can be used for messaging, i.e., a parent PLC cannot monitor or control the signals on a child port. Therefore, the port number specified in the fifth implied node of the control block must always be 1.

The first three registers in the data block (the displayed register and the first and second implied registers in the middle node) have predetermined content:

Register	Content
Displayed	Stores the control mask word
First implied	Stores the control data word
Second implied	Stores the status word

These three data block registers are required for this format, and therefore the allowable range for the length value (specified in the bottom node) is 3 ... 255.

Control Mask Word

Usage of word:

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

Bit	Function
1	1 = port can be taken 0 = port cannot be taken
2 - 15	Not used
16	1 = control RTS 0 = do not control RTS

Control Data Word

Usage of word:

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

Bit	Function
1	1 = take port 0 = return port
2 - 15	Not used
16	1 = activate RTS 0 = deactivate RTS

Status Word

Usage of word:

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

Bit	Function
1	1 = port taken
2	1 = port ACTIVE as Modbus slave
3 - 13	Not used
14	1 = DSR ON
15	1 = CTS ON
16	1 = RTS ON

Interrupt Handling

5

Interrupt Handling

Interrupt-related Performance

The interrupt-related instructions operate with minimum processing overhead. The performance of interrupt-related instructions is especially critical. Using a interval timer interrupt (ITMR) instruction adds about 6% to the scan time of the scheduled ladder logic, this increase does not include the time required to execute the interrupt handler subroutine associated with the interrupt.

Interrupt Latency Time

The following table shows the minimum and maximum interrupt latency times you can expect:

ITMR overhead	No work to do	60 ms/ms
Response time	Minimum	98 ms
	Maximum during logic solve and Modbus command reception	400 ms
Total overhead (not counting normal logic solve time)		155 ms

These latency times assume only one interrupt at a time.

Interrupt Priorities

The PLC uses the following rules to choose which interrupt handler to execute in the event that multiple interrupts are received simultaneously:

- An interrupt generated by an interrupt module has a higher priority than an interrupt generated by a timer.
- Interrupts from modules in lower slots of the local backplane have priority over interrupts from modules in the higher slots.

If the PLC is executing an interrupt handler subroutine when a higher priority interrupt is received, the current interrupt handler is completed before the new interrupt handler is begun.

Instructions that Cannot Be Used in an Interrupt Handler

The following (nonreenterant) ladder logic instructions cannot be used inside an interrupt handler subroutine:

- MSTR
- READ / WRIT
- PCFL / EMTH
- T1.0, T0.1, T.01 and T1MS timers (will not set error bit 2, timer results invalid)
- Equation Networks
- User loadables (will not set error bit 2)

If any of these instructions are placed in an interrupt handler, the subroutine will be aborted, the error output on the ITMR or IMOD instruction that generated the interrupt will go ON, and bit 2 in the status register will be set.

Interrupt with BMDI/ID/IE

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.

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

The BMDI instruction can be used to mask both a timer-generated and local I/O-generated interrupts, perform a single block data move, then unmask the interrupts. It allows for the exchange of a block of data either within the subroutine or at one or more places in the scheduled logic program.

BMDI instructions can be used to reduce the time between the disable and enable of interrupts. For example, BMDI instructions can be used to protect the data used by the interrupt handler when the data is updated or read by Modbus, Modbus Plus, Peer Cop or Distributed I/O (DIO).

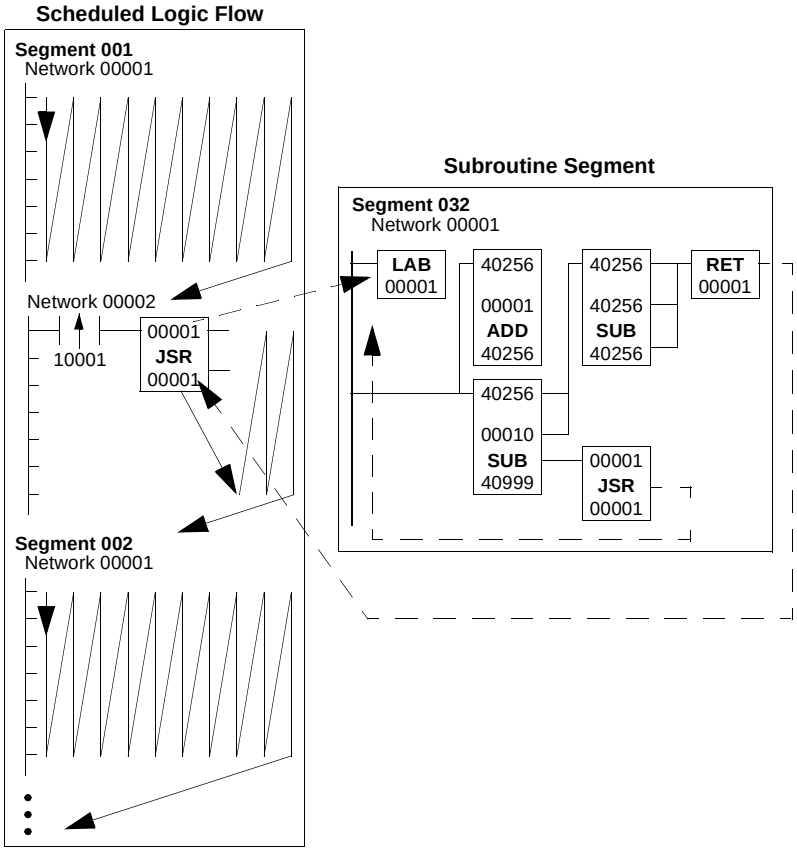
Subroutine Handling

6

Subroutine Handling

JSR / LAB Method

The example below shows a series of three user logic networks, the last of which is used for an up-counting subroutine. Segment 32 has been removed from the order-of-solve table in the segment scheduler:



When input 100001 to the JSR block in network 2 of segment 1 transitions from OFF to ON, the logic scan jumps to subroutine #1 in network 1 of segment 32.

The subroutine will internally loop on itself ten times, counted by the ADD block. The first nine loops end with the JSR block in the subroutine (network 1 of segment 32) sending the scan back to the LAB block. Upon completion of the tenth loop, the RET block sends the logic scan back to the scheduled logic at the JSR node in network 2 of segment 1.

Installation of DX Loadables

7

Installation of DX Loadables

How to install the DX Loadables

The DX loadable instructions are only available if you have installed them. With the installation of the Concept software, DX loadables are located on your hard disk. Now you have to unpack and install the loadables you want to use as follows:

Step	Action
1	With the menu command `Project` → `Configurator` you open the configurator
2	With `Configure` → `Loadables...` you open the dialog box `Loadables`
3	Press the command button `Unpack...` to open the standard Windows dialog box `Unpack Loadable File` where the multifile loadables (DX loadables) can be selected. Select the loadable file you need, click the button `OK` and it is inserted into the list box `Available:`.
4	Now press the command button `Install=>` to install the loadable selected in the list box `Available:`. The installed loadable will be displayed in the list box `Installed:`.
5	Press the command button `Edit...` to open the dialog box `Loadable Instruction Configuration`. Change the opcode if necessary or accept the default. You can assign an opcode to the loadable in the list box `Opcode` in order to enable user program access through this code. An opcode that is already assigned to a loadable, will be identified by a *. Click the button `OK`.
6	Click the button `OK` in the dialog box `Loadables`. Configuration loadables count is adjusted. The installed loadable is available for programming at the menu `Objects` → `List Instructions` → `DX Loadable`.

Coils, Contacts and Interconnects

8

At a Glance

Introduction

In this chapter you will find information about Coils, Contacts and Interconnects (Shorts.)

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Coils	56
Contacts	58
Interconnects (Shorts)	60

Coils

Definition of Coils

A coil is a discrete output that is turned ON and OFF by power flow in the logic program. A single coil is tied to a 0x reference in the PLC's state RAM. Because output values are updated in state RAM by the PLC, a coil may be used internally in the logic program or externally via the I/O map to a discrete output unit in the control system. When a coil is ON, it either passes power to a discrete output circuit or changes the state of an internal relay contact in state RAM.

There are two types of coils:

- A normal coil
- A memory-retentive, or latched, coil

Normal Coil

A normal coil is a discrete output shown as a 0x reference.

A normal coil is ON or OFF, depending on power flow in the program.

A ladder logic network can contain up to seven coils, no more than one per row. When a coil is placed in a row, no other logic elements or instruction nodes can appear to the right of the coil's logic-solve position in the row. Coils are the only ladder logic elements that can be inserted in column 11 of a network.

To define a discrete reference for the coil, select it in the editor and click to open a dialog box called `Coil`.

Symbol

	WARNING
	Forcing of Coils When a discrete input (1x) is disabled, signals from its associated input field device have no control over its ON/OFF state. When a discrete output (0x) is disabled, the PLC's logic scan has no control over the ON/OFF state of the output. When a discrete input or output has been disabled, you can change its current ON/OFF state with the Force command. There is an important exception when you disable coils. Data move and data matrix instructions that use coils in their destination node recognize the current ON/OFF state of all coils in that node, whether they are disabled or not. If you are expecting a disabled coil to remain disabled in such an instruction, you may cause unexpected or undesirable effects in your application. When a coil or relay contact has been disabled, you can change its state using the Force ON or Force OFF command. If a coil or relay is enabled, it cannot be forced. **Failure to observe this precaution can result in severe injury or equipment damage.**

Retentive Coil

If a retentive (latched) coil is energized when the PLC loses power, the coil will come back up in the same state for one scan when the PLC's power is restored.

To define a discrete reference for the coil, select it in the editor and click to open a dialog box called `Retentative coil (latch)`.

Symbol

Contacts

Definition of Contacts

Contacts are used to pass or inhibit power flow in a ladder logic program. They are discrete, i.e., each consumes one I/O point in ladder logic. A single contact can be tied to a 0x or 1x reference number in the PLC's state RAM, in which case each contact consumes one node in a ladder network.

Four kinds of contacts are available:
- Normally open (N.O.) contacts
- Normally closed (N.C.) contacts
- Positive transitional (P.T.) contacts
- Negative transitional (N.T.) contacts

Contact Normally Open

A normally open (NO) contact passes power when it is ON.
To define a discrete reference for the NO contact, select it in the editor and click to open a dialog called `Normally open contact`.

Symbol

Contact Normally Closed

A normally closed (NC) contact passes power when it is OFF.

To define a discrete reference for the NC contact, double ckick on it in the ladder node to open a dialog called `Normally closed contact`.

Symbol

Contact Pos Trans

A positive transitional (PT) contact passes power for only one scan as it transitions from OFF to ON.

To define a discrete reference for the PT contact, select it in the editor and click to open a dialog called `Positive transition contact`.

Symbol

Contact Neg Trans

A negative transitional (NT) contact passes power for only one scan as it transitions from ON to OFF.

To define a discrete reference for the NT contact, select it in the editor and click to open a dialog called `Contact negative transition` .

Symbol

Interconnects (Shorts)

Definition of Interconnects (Shorts)

Shorts are simply straight-line connections between contacts and/or instructions in a ladder logic network. Shorts may be inserted horizontally or vertically in a network.

Two kinds of shorts are available:
- Horizontal Short
- Vertical Short

Horizontal Short

A short is a straight-line connection between contacts and/or nodes in an instruction through which power flow can be controlled.

A horizontal short is used to extend logic out across a row in a network without breaking the power flow. Each horizontal short consumes one node in the network, and uses a word of memory in the PLC.

Symbol

Vertical Short

A vertical short connects contacts or nodes in an instruction positioned one above the other in a column. Vertical shorts can also connect inputs or outputs in an instruction to create either-or conditions. When two contacts are connected by a vertical short, power is passed when one or both contacts receive power.

The vertical short is unique in two ways:
- It can coexist in a network node with another element or nodal value
- It does not consume any PLC memory

Symbol

Instruction Descriptions

II

At a Glance

Introduction

The instruction descriptions are arranged alphabetically according to their abbreviations.

What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
9	AD16: Ad 16 Bit	67
10	ADD: Addition	69
11	AND: Logical And	71
12	BCD: Binary to Binary Code	75
13	BLKM: Block Move	77
14	BLKT: Block to Table	81
15	BMDI: Block Move with Interrupts Disabled	85
16	BROT: Bit Rotate	87
17	CHS: Configure Hot Standby	91
18	CKSM: Check Sum	97
19	CMPR: Compare Register	101
20	COMP: Complement a Matrix	105
21	DCTR: Down Counter	109
22	DIOH: Distributed I/O Health	111
23	DIV: Divide	115
24	DLOG: Data Logging for PCMCIA Read/Write Support	119
25	DRUM: DRUM Sequencer	125
26	DV16: Divide 16 Bit	129
27	EMTH: Extended Math	133
28	EMTH-ADDDP: Double Precision Addition	139
29	EMTH-ADDFP: Floating Point Addition	143
30	EMTH-ADDIF: Integer + Floating Point Addition	147
31	EMTH-ANLOG: Base 10 Antilogarithm	151
32	EMTH-ARCOS: Floating Point Arc Cosine of an Angle (in Radians)	155
33	EMTH-ARSIN: Floating Point Arcsine of an Angle (in Radians)	159
34	EMTH-ARTAN: Floating Point Arc Tangent of an Angle (in Radians)	163
35	EMTH-CHSIN: Changing the Sign of a Floating Point Number	167
36	EMTH-CMPFP: Floating Point Comparison	171
37	EMTH-CMPIF: Integer-Floating Point Comparison	175
38	EMTH-CNVDR: Floating Point Conversion of Degrees to Radians	179
39	EMTH-CNVFI: Floating Point to Integer Conversion	183

Chapter	Chaptername	Page
40	EMTH-CNVIF: Integer-to-Floating Point Conversion	187
41	EMTH-CNVRD: Floating Point Conversion of Radians to Degrees	191
42	EMTH-COS: Floating Point Cosine of an Angle (in Radians)	195
43	EMTH-DIVDP: Double Precision Division	199
44	EMTH-DIVFI: Floating Point Divided by Integer	203
45	EMTH-DIVFP: Floating Point Division	207
46	EMTH-DIVIF: Integer Divided by Floating Point	211
47	EMTH-ERLOG: Floating Point Error Report Log	215
48	EMTH-EXP: Floating Point Exponential Function	219
49	EMTH-LNFP: Floating Point Natural Logarithm	223
50	EMTH-LOG: Base 10 Logarithm	227
51	EMTH-LOGFP: Floating Point Common Logarithm	231
52	EMTH-MULDP: Double Precision Multiplication	235
53	EMTH-MULFP: Floating Point Multiplication	239
54	EMTH-MULIF: Integer x Floating Point Multiplication	243
55	EMTH-PI: Load the Floating Point Value of "Pi"	247
56	EMTH-POW: Raising a Floating Point Number to an Integer Power	251
57	EMTH-SINE: Floating Point Sine of an Angle (in Radians)	255
58	EMTH-SQRFP: Floating Point Square Root	259
59	EMTH-SQRT: Floating Point Square Root	263
60	EMTH-SQRTP: Process Square Root	267
61	EMTH-SUBDP: Double Precision Subtraction	271
62	EMTH-SUBFI: Floating Point - Integer Subtraction	275
63	EMTH-SUBFP: Floating Point Subtraction	279
64	EMTH-SUBIF: Integer - Floating Point Subtraction	283
65	EMTH-TAN: Floating Point Tangent of an Angle (in Radians)	287
66	ESI: Support of the ESI Module	291
67	EUCA: Engineering Unit Conversion and Alarms	309
68	FIN: First In	321
69	FOUT: First Out	325
70	FTOI: Floating Point to Integer	329
71	HLTH: History and Status Matrices	331
72	IBKR: Indirect Block Read	345

Chapter	Chaptername	Page
73	IBKW: Indirect Block Write	347
74	ICMP: Input Compare	349
75	ID: Interrupt Disable	355
76	IE: Interrupt Enable	359
77	IMIO: Immediate I/O	363
78	IMOD: Interrupt Module Instruction	369
79	ITMR: Interrupt Timer	377
80	ITOF: Integer to Floating Point	383
81	JSR: Jump to Subroutine	385
82	LAB: Label for a Subroutine	387
83	LOAD: Load Flash	391
84	MAP 3: MAP Transaction	395
85	MBIT: Modify Bit	403
86	MBUS: MBUS Transaction	407
87	MRTM: Multi-Register Transfer Module	417
88	MSTR: Master	423
89	MU16: Multiply 16 Bit	465
90	MUL: Multiply	467
91	NBIT: Bit Control	469
92	NCBT: Normally Closed Bit	471
93	NOBT: Normally Open Bit	473
94	NOL: Network Option Module for Lonworks	475
95	OR: Logical OR	479
96	PCFL: Process Control Function Library	483
97	PCFL-AIN: Analog Input	491
98	PCFL-ALARM: Central Alarm Handler	497
99	PCFL-AOUT: Analog Output	501
100	PCFL-AVER: Average Weighted Inputs Calculate	505
101	PCFL-CALC: Calculated preset formula	511
102	PCFL-DELAY: Time Delay Queue	515
103	PCFL-EQN: Formatted Equation Calculator	521
104	PCFL-INTEG: Integrate Input at Specified Interval	527
105	PCFL-KPID: Comprehensive ISA Non Interacting PID	531
106	PCFL-LIMIT: Limiter for the Pv	537

Chapter	Chaptername	Page
107	PCFL-LIMV: Velocity Limiter for Changes in the Pv	541
108	PCFL-LKUP: Look-up Table	545
109	PCFL-LLAG: First-order Lead/Lag Filter	549
110	PCFL-MODE: Put Input in Auto or Manual Mode	553
111	PCFL-ONOFF: ON/OFF Values for Deadband	557
112	PCFL-PI: ISA Non Interacting PI	563
113	PCFL-PID: PID Algorithms	567
114	PCFL-RAMP: Ramp to Set Point at a Constant Rate	573
115	PCFL-RATE: Derivative Rate Calculation over a Specified Timeme	579
116	PCFL-RATIO: Four Station Ratio Controller	583
117	PCFL-RMPLN: Logarithmic Ramp to Set Point	589
118	PCFL-SEL: Input Selection	593
119	PCFL-TOTAL: Totalizer for Metering Flow	597
120	PEER: PEER Transaction	603
121	PID2: Proportional Integral Derivative	607
122	R –––> T: Register to Table	621
123	RBIT: Reset Bit	625
124	READ: Read	627
125	RET: Return from a Subroutine	633
126	SAVE: Save Flash	635
127	SBIT: Set Bit	639
128	SCIF: Sequential Control Interfaces	641
129	SENS: Sense	647
130	SKPC: Skip (Constants)	651
131	SKPR: Skip (Registers)	655
132	SRCH: Search	659
133	STAT: Status	663
134	SU16: Subtract 16 Bit	687
135	SUB: Subtraction	689
136	T–––>R: Table to Register	691
137	T–––>T: Table to Table	695
138	T.01 Timer: One Hundredth Second Timer	699
139	T0.1 Timer: One Tenth Second Timer	701
140	T1.0 Timer: One Second Timer	703

Chapter	Chaptername	Page
141	T1MS Timer: One Millisecond Timer	705
142	TBLK: Table to Block	711
143	TEST: Test of 2 Values	715
144	UCTR: Up Counter	717
145	WRIT: Write	719
146	XMIT: XMIT Communication Block	725
147	XMRD: Extended Memory Read	735
148	XMWT: Extended Memory Write	739
149	XOR: Exclusive OR	743

AD16: Ad 16 Bit

9

At a Glance

Introduction

This chapter describes the instruction AD16.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	68
Representation	68

Short Description

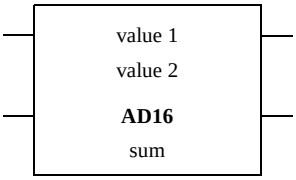
Function Description

The AD16 instruction performs signed or unsigned 16-bit addition on value 1 (its top node) and value 2 (its middle node), then posts the sum in a 4x holding register 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 = add value 1 and value 2
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Addend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Addend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
sum (bottom node)	4x	INT, UINT	Sum of 16 bit addition
Top output	0x	None	ON = successful completion of the operation
Bottom output	0x	None	ON = overflow in the sum:

ADD: Addition

10

At a Glance

Introduction

This chapter describes the instruction ADD.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	70
Representation	70

Short Description

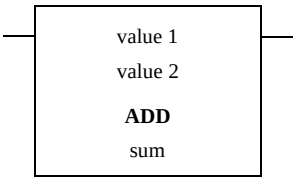
Function Description

The ADD instruction adds unsigned value 1 (its top node) to unsigned value 2 (its middle node) and stores the sum in a holding register 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 = add value 1 and value 2
value 1 (top node)	3x, 4x	INT, UINT	Addened, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Addend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
sum (bottom node)	4x	INT, UINT	Sum
Top output	0x	None	ON = overflow in the sum: sum > 9 999

AND: Logical And

11

At a Glance

Introduction

This chapter describes the instruction AND.

What's in this chapter?

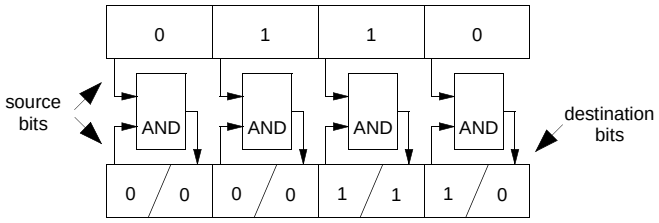
This chapter contains the following topics:

Topic	Page
Short Description	72
Representation	73
Parameter Description	73

Short Description

Function Description

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

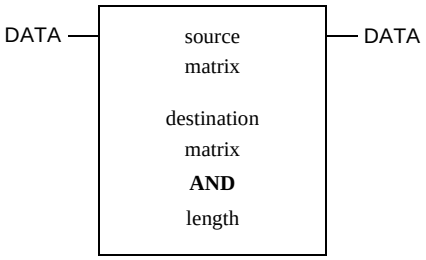


STOP	**WARNING**
	Overriding of any disabled coils within the destination matrix without enabling them. AND will override any disabled coils within the destination matrix without enabling them.This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's state can be changed by the AND 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 AND
source matrix (top node)	0x, 1x, 3x, 4x	BOOL, WORD	First reference in the source matrix
destination matrix (middle node)	0x, 4x	BOOL, WORD	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 ANDed.

BCD: Binary to Binary Code

12

At a Glance

Introduction

This chapter describes the instruction BCD.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	76
Representation	76

Short Description

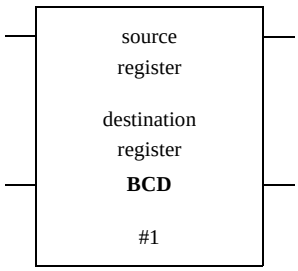
Function Description

The BCD instruction can be used to convert a binary value to a binary coded decimal (BCD) value or a BCD value to a binary value. The type of conversion to be performed is controlled by the state of the bottom input.

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 = enable conversion
Bottom input	0x, 1x	None	ON = BCD → binary conversion OFF = binary → BCD conversion
Source register (top node)	3x, 4x	INT, UINT	Source register where the numerical value to be converted is stored
Destination register (middle node)	4x	INT, UINT	Destination register where the converted numerical value is posted
#1 (bottom node)		INT, UINT	Constant value, can not be changed
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = error in the conversion operation

BLKM: Block Move

13

At a Glance

Introduction

This chapter describes the instruction BLKM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	78
Representation	79

Short Description

Function Description

The BLKM (block move) instruction copies the entire contents of a source table to a destination table in one scan.

WARNING

Overriding of any disabled coils within a destination table without enabling them.

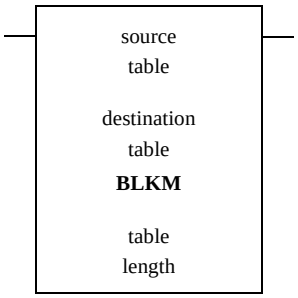
BLKM will override any disabled coils within a destination table without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the BLKM instruction.

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	ON = initiates block move
source table (top node)	0x, 1x, 3x, 4x	ANY_BIT	Source table that will have its contents copied in the block move
destination table (middle node)	0x, 4x	ANY_BIT	Destination table where the contents of the source table will be copied in the block move
table length (bottom node)		INT, UINT	Table size (number of registers or 16-bit words) for both the source and destination tables; they are of equal length. Range: 1 ... 100.
Top output	0x	None	Echos the state of the top input

BLKT: Block to Table

14

At a Glance

Introduction

This chapter describes the instruction BLKT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	82
Representation	83
Parameter Description	84

Short Description

Function Description

The BLKT (block-to-table) instruction combines the functions of R→T and BLKM in a single instruction. In one scan, it can copy data from a source block to a destination block in a table. The source block is of a fixed length. The block within the table is of the same length, but the overall length of the table is limited only by the number of registers in your system configuration.

WARNING

All the 4x registers in your PLC can be corrupted with data copied from the source block.

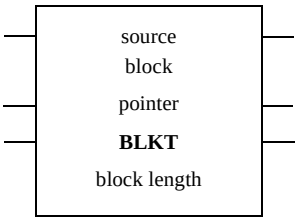
BLKT is a powerful instruction that can corrupt all the 4x registers in your PLC with data copied from the source block. You should use external logic in conjunction with the middle or bottom input to confine the value in the pointer to a safe range.

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	ON = initiates the DX move
Middle input	0x, 1x	None	ON = hold pointer
Bottom input	0x, 1x	None	ON = reset pointer to zero
source block (top node)	4x	BYTE, WORD	First holding register in the block of contiguous registers whose content will be copied to a block of registers in the destination table.
pointer (middle node)	4x	BYTE, WORD	Pointer to the destination table
block length (bottom node)		INT, UINT	Block length (number of 4x registers) of the source block and of the destination block. Range: 1 ... 100.
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = error / move not possible

Parameter Description

Middle and Bottom Input

The middle and bottom input can be used to control the pointer so that source data is not copied into registers that are needed for other purposes in the logic program. When the middle input is ON, the value in the pointer register is frozen while the BLKT operation continues. This causes new data being copied to the destination to overwrite the block data copied on the previous scan.
When the bottom input is ON, the value in the pointer register is reset to zero. This causes the BLKT operation to copy source data into the first block of registers in the destination table.

Pointer (Middle Node)

The 4x register entered in the middle node is the pointer to the destination table. The first register in the destination table is the next contiguous register after the pointer, e.g. if the pointer register is 400107, then the first register in the destination table is 400108.

Note: The destination table is segmented into a series of register blocks, each of which is the same length as the source block. Therefore, the size of the destination table is a multiple of the length of the source block, but its overall size is not specifically defined in the instruction. If left uncontrolled, the destination table could consume all the 4x registers available in the PLC configuration.

The value stored in the pointer register indicates where in the destination table the source data will begin to be copied. This value specifies the block number within the destination table.

BMDI: Block Move with Interrupts Disabled

15

At a Glance

Introduction

This chapter describes the instruction BMDI.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	86
Representation	86

Short Description

Function Description

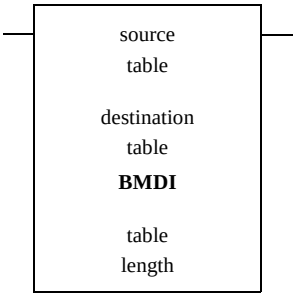
Note: This instruction is only available after configuring a CPU without extension.

The BMDI instruction masks the interrupt, initiates a block move (BLKM) operation, then unmasks the interrupts.
Further Information you will find in the chapter "*Interrupt Handling, p. 49*".

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 = masks interrupt, initiates a block move, then unmasks the interrupts
source table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	Source table that will have its contents copied in the block move
destination table (middle node)	0x, 4x	INT, UINT, WORD	Destination table where the contents of the source table will be copied in the block move
table length (bottom node)		INT, UINT	Integer value, specifies the table size, i.e. the number of registers, in the source and destination tables (they are of equal length). Range: 1 ... 100.
Top output	0x	None	Echoes the state of the top input

BROT: Bit Rotate

16

At a Glance

Introduction

This chapter describes the instruction BROT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	88
Representation	89
Parameter Description	90

Short Description

Function Description

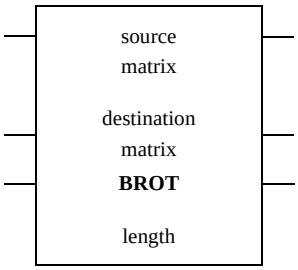
The BROT (bit rotate) instruction shifts the bit pattern in a source matrix, then posts the shifted bit pattern in a destination matrix. The bit pattern shifts left or right by one position per scan.

	WARNING
	Overriding of any disabled coils within a destination matrix without enabling them. BROT will override any disabled coils within a destination matrix without enabling them. This can cause injury if a coil has been disabled for repair or maintenance if BROT unexpectedly changes the coil's state. **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	ON = shifts bit pattern in source matrix by one
Middle input	0x, 1x	None	ON= shift left OFF = shift right
Bottom input	0x, 1x	None	OFF = exit bit falls out of the destination matrix ON = exit bit wraps to start of the destination matrix
source matrix (top node)	0x, 1x, 3x, 4x	ANY_BIT	First reference in the source matrix, i.e. in the matrix that will have its bit pattern shifted
destination matrix (middle node)	0x, 4x	ANY_BIT	First reference in the destination matrix, i.e. in the matrix that shows the shifted bit pattern
length (bottom node)	0x	INT, UINT	Matrix length; range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	OFF = exit bit is 0 ON = exit bit is 1

Parameter Description

Matrix Length (Bottom Node)	The integer value entered in the bottom node specifies the matrix length, i.e. the number of registers or 16-bit words in each of the two matrices. The source matrix and destination matrix have the same length. The matrix length can range from 1 ... 100, e.g. a matrix length of 100 indicates 1600 bit locations.
Result of the Shift (Middle Output)	The middle output indicates the sense of the bit that exits the source matrix (the leftmost or rightmost bit) as a result of the shift.

CHS: Configure Hot Standby

17

At a Glance

Introduction

This chapter describes the instruction CHS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	92
Representation	93
Detailed Description	94

Short Description

Function Description

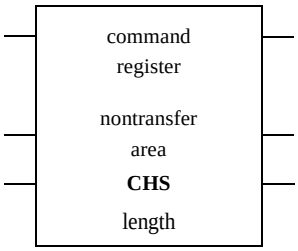
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 53*".

The logic in the CHS loadable is the engine that drives the Hot Standby capability in a Quantum PLC system. Unlike the HSBY instruction, the use of the CHS instruction in the ladder logic program is optional. However, the loadable software itself must be installed in the Quantum PLC in order for a Hot Standby system to be implemented.

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	Execute Hot Standby (unconditionally)
Middle input	0x, 1x	None	ON = Enable command register
Bottom input	0x, 1x	None	ON = Enable nontransfer area OFF = nontransfer area will not be used and the Hot Standby status register will not exist
command register (top node)	4x	INT, UINT, WORD	Hot Standby command register
nontransfer area (middle node)	4x	INT, UINT, WORD	First register in the nontransfer area of state RAM
length (bottom node)		INT, UINT	Number of registers of the Hot Standby nontransfer area in state RAM; range 4 ... 8000
Top output	0x	None	Hot Standby system ACTIVE
Middle output	0x	None	PLC cannot communicate with its CHS module
Bottom output	0x	None	Configuration extension screens are defining the Hot Standby configuration

Detailed Description

Hot Standby System Configuration via the CHS Instruction

Program the CHS instruction in network 1, segment 1 of your ladder logic program and unconditionally connect the top input to the power rail via a horizontal short (as the HSBY instruction is programmed in a 984 Hot Standby system).
This method is particularly useful if you are porting Hot Standby code from a 984 application to a Quantum application. The structure of the CHS instruction is almost exactly the same as the HSBY instruction. You simply remove the HSBY instruction from the 984 ladder logic and replace it with a CHS instruction in the Quantum logic.
If you are using the CHS instruction in ladder logic, the only difference between it and the HSBY instruction is the use of the bottom output. This output senses whether or not method 2 has been used. If the Hot Standby configuration extension screens have been used to define the Hot Standby configuration, the configuration parameters in the screens will override any different parameters defined by the CHS instruction at system startup.
For detailes discussion of the issues related to the configuration extension capabilities of a Quantum Hot Standby system, refer to the *Modicon Quantum Hot Standby System Planning and Installation Guide.*

Parameter Description Execute Hot Standby (Top Input)

When the CHS instruction is inserted in ladder logic to control the Hot Standby configuration parameters, its top input must be connected directly to the power rail by a horizontal short. No control logic, such as contacts, should be placed between the rail and the input to the top node.

WARNING

Erratic behavior in the Hot Standby system

Although it is legal to enable and disable the nontransfer area while the Hot Standby system is running, we strongly discourage this practice. It can lead to erratic behavior in the Hot Standby system.

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

Parameter Description Command Register (Top Node)

The 4x register entered in the top node is the Hot Standby command register; eight bits in this register are used to configure and control Hot Standby system parameters:

Usage of command word:

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

Bit	Function
1 - 5	Not used
6	0 = swap Modbus port 3 address during switchover 1 = no swap
7	0 = swap Modbus port 2 address during switchover 1 = no swap
8	0 = swap Modbus port 1 address during switchover 1 = no swap
9 - 11	Not used
12	0 = allow exec upgrade only after application stops 1 = allow the upgrade without stopping the application
13	0 = force standby offline if there is a logic mismatch 1 = do not force
14	0 = controller B is in **OFFLINE** mode 1 = controller B is in **RUN**
15	0 = controller A is in **OFFLINE** mode 1 = controller A is in **RUN**
16	0 = disable keyswitch override 1 = enable the override

Note: The Hot Standby command register must be outside of the nontransfer area of state RAM.

Parameter Description Nontransfer Area (Middle Node)

The 4x register entered in the middle node is the first register in the nontransfer area of state RAM. The nontransfer area must contain at least four registers, the first three of which have a predefined usage:

Register	Content
Displayed and first implied	Reverse transfer registers for passing information from the standby to the primary PLC
Second implied	*CHS Status Register, p. 96*

The content of the remaining registers is application-specific; the length is defined in the parameter "length" (bottom node).
The 4x registers in the nontransfer area are never transferred from the primary to the standby PLC during the logic scans. One reason for scheduling additional registers in the nontransfer area is to reduce the impact of state RAM transfer on the total system scan time.

CHS Status Register

Usage of status word:

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

Bit	Function
1	1 = the top output is ON (indicating Hot Standby system is active)
2	1 = the middle output is ON (indicating an error condition)
3 - 10	Not used
11	0 = PLC switch is set to A 1 = PLC switch is set to B
12	0 = PLC logic is matched 1 = there is a logic mismatch
13 - 14	The 2 bit value is: ● 0 1 if the other PLC is in **OFFLINE** mode ● 1 0 if other PLC is running in primary mode ● 1 1 if other PLC is running in standby mode
15 - 16	The 2 bit value is: ● 0 1 if this PLC is in **OFFLINE** mode ● 1 0 if this PLC is running in primary mode ● 1 1 if this PLC is running in standby mode

CKSM: Check Sum

18

At a Glance

Introduction

This chapter describes the instruction CKSM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	98
Representation	98
Parameter Description	99

Short Description

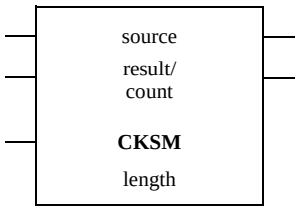
Function Description

Several PLCs that do not support Modbus Plus come with a standard checksum (CKSM) instruction. CKSM has the same opcode as the MSTR instruction and is not provided in executive firmware for PLCs that support Modbus Plus.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input (See *Inputs, p. 99*)	0x, 1x	None	Initiates checksum calculation of source table
Middle input	0x,1x	None	Cksm select 1
Bottom input	0x, 1x	None	Cksm select 2
source (top node)	4x	INT, UINT	First holding register in the source table. The checksum calculation is performed on the registers in this table.
result/count (middle node)	4x	INT, UINT	First of two contiguous registers
length (bottom node)		INT	Number of 4x registers in the source table; range: 1 ... 255
Top output	0x	None	ON = calculation successful
Bottom output	0x	None	ON = implied register count > length or implied register count =0

Parameter Description

Inputs

The states of the inputs indicate the type of checksum calculation to be performed:

CKSM Calculation	Top Input	Middle Input	Bottom Input
Straight Check	ON	OFF	ON
Binary Addition Check	ON	ON	ON
CRC-16	ON	ON	OFF
LRC	ON	OFF	OFF

Result / Count (Middle Node)

The 4x register entered in the middle node is the first of two contiguous 4x registers:

Register	Content
Displayed	Stores the result of the checksum calculation
First implied	Posts a value that specifies the number of registers selected from the source table as input to the calculation. The value posted in the implied register must be ≤ length of source table.

CMPR: Compare Register

19

At a Glance

Introduction

This chapter describes the instruction CMPR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	102
Representation	102
Parameter Description	103

Short Description

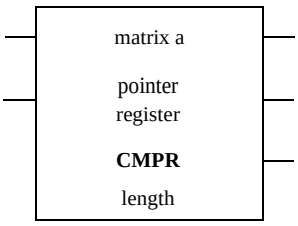
Function Description

The CMPR instruction compares the bit pattern in matrix a against the bit pattern in matrix b for miscompares. In a single scan, the two matrices are compared bit position by bit position until a miscompare is found or the end of the matrices is reached (without miscompares).

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 = intiiates compare operation
Middle input	0x, 1x	None	OFF = restart at last miscompare ON = restart at the beginning
matrix a (top node)	0x, 1x, 3x, 4x	ANY_BIT	First reference in matrix a, one of the two matrices to be compared
pointer register (midlle node)	4x	WORD	Pointer to matrix b: the first register in matrix b is the next contiguous 4x register following the pointer register
length (bottom node)		INT, UINT	Matrix length; range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = miscompare detected
Bottom output	0x	None	ON = miscompared bit in matrix a is 1 OFF = miscompared bit in matrix a is 0

Parameter Description

Pointer Register (Middle Node)	The pointer register entered in the middle node must be a 4x holding register. It is the pointer to matrix b, the other matrix to be compared. The first register in matrix b is the next contiguous 4x register following the pointer register. The value stored inside the pointer register increments with each bit position in the two matrices that is being compared. As bit position 1 in matrix a and matrix b is compared, the pointer register contains a value of 1; as bit position 2 in the matrices are compared, the pointer value increments to 2; etc. When the outputs signal a miscompare, you can check the accumulated count in the pointer register to determine the bit position in the matrices of the miscompare.
Matrix Length (Bottom Node)	The integer value entered in the bottom node specifies a length of the two matrices, i.e. the number of registers or 16-bit words in each matrix. (Matrix a and matrix b have the same length.) The matrix length can range from 1 ... 100, i.e. a length of 2 indicates that matrix a and matrix b contain 32 bits.

COMP: Complement a Matrix

20

At a Glance

Introduction

This chapter describes the instruction COMP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	106
Representation	107
Parameter Description	107

Short Description

Function Description

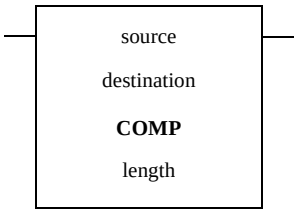
The COMP instruction complements the bit pattern, i.e. changes all 0's to 1's and all 1's to 0's, of a source matrix, then copies the complemented bit pattern into a destination matrix. The entire COMP operation is accomplished in one scan.

	WARNING
	Overriding of any disabled coils in the destination matrix without enabling them. COMP will override any disabled coils in the destination matrix without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can be changed by the COMP 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	ON = initiates the complement operation
source (top node)	0x, 1x, 3x, 4x	ANY_BIT	First reference in the source matrix, which contains the original bit pattern before the complement operation
destination (middle node)	0x, 4x	ANY_BIT	First reference in the destination matrix where the complemented bit pattern will be posted
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 value entered in the bottom node specifies a matrix length, i.e. the number of registers or 16-bit words in the matrices. Matrix length can range from 1 ... 100. A length of 2 indicates that 32 bits in each matrix will be complemented.

DCTR: Down Counter

21

At a Glance

Introduction

This chapter describes the instruction DCTR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	110
Representation	110

Short Description

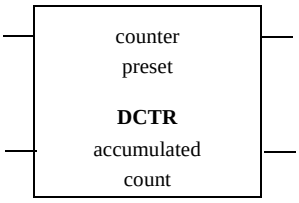
Function Description

The DCTR instruction counts control input transitions from OFF to ON down from a counter preset value to zero.

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	OFF → ON = initiates the counter operation
Bottom input	0x, 1x	None	OFF = accumulated count is reset to preset value ON = counter accumulating
counter preset (top node)	3x, 4x	INT, UINT	Preset value, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated count (bottom node)	4x	INT, UINT	Count value (actual value); which decrements by one on each transition from OFF to ON of the top input until it reaches zero.
Top output	0x	None	ON = accumulated count = 0
Bottom output	0x	None	ON = accumulated count > 0

DIOH: Distributed I/O Health

22

At a Glance

Introduction

This chapter describes the instruction DIOH.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	112
Representation	112
Parameter Description	113

Short Description

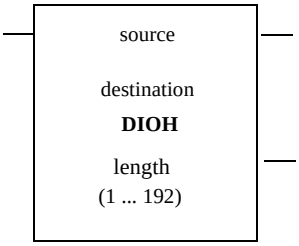
Function Description

The DIOH instruction lets you retrieve health data from a specified group of drops on the distributed I/O network. It accesses the DIO health status table, where health data for modules in up to 189 distributed drops is stored.

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 retrieval of the specified status words from the DIO health table into the destination table
source (top node)		INT, UINT	Source value (four-digit constant in the form **xxyy**)
destination (middle node)	4x	INT, UINT, WORD	First holding register in the destination table, i.e. in a block of contiguous registers where the retrieved health status information is stored
length (bottom node)		INT, UINT	Length of the destination table, range 1 ... 64
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = invalid source entry

Parameter Description

Source Value (Top Node)

The source value entered in the top node is a four-digit constant in the form **xxyy**, where:

Digits	Meaning
xx	Decimal value in the range 00 ... 16, indicating the slot number in which the relevant DIO processor resides. The value 00 can always be used to indicate the Modbus Plus ports on the PLC, regardless of the slot in which it resides.
yy	Decimal value in the range 1 ... 64, indicating the drop number on the appropriate token ring

For example, if you are interested in retrieving drop status starting at distributed drop #1 on a network being handled by a DIO processor in slot 3, enter **0301** in the top node.

Length of Destination Table (Bottom Node)

The integer value entered in the bottom node specifies the length, i.e. the number of 4x registers, in the destination table. The length is in the range 1 ... 64.

Note: If you specify a length that excedes the number of drops available, the instruction will return status information only for the drops available. For example, if you specify the 63rd drop number (yy) in the top node register and then request a length of 5, the instruction will give you only two registers (the 63rd and 64th drop status words) in the destination table.

DIV: Divide

23

At a Glance

Introduction

This chapter describes the instruction DIV.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	116
Representation	117
Example	118

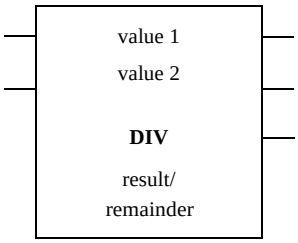
Short Description

Function Description	The DIV instruction divides unsigned value 1 (its top node) by unsigned value 2 (its middle node) and posts the quotient and remainder 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 divided by value 2
Middle input	0x, 1x	None	ON = decimal remainder OFF = fraction remainder
value 1 (top node)	3x, 4x	INT, UINT	Dividend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in two contiguous registers (displayed for hig-horder half, implied for low-order half)
value 2 (middle node)	3x, 4x	INT, UINT	Divisor, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
result / remainder (bottom node)	4x	INT, UINT	First of two contiguous holding registers: displayed: result of division implied: remainder (either a decimal or a fraction, depending on the state of middle input)
Top output	0x	None	ON = division successful
Middle output	0x	None	ON = overflow: if result > 9 999, a 0 value is returned
Bottom output	0x	None	ON = value 2 = 0

Example

Quotient of Instruction DIV

The state of the middle input indicates whether the remainder will be expressed as a decimal or as a fraction. For example, if value 1 = 8 and value 2 = 3, the decimal remainder (middle input ON) is 6666; the fractional remainder (middle input OFF) is 2.

DLOG: Data Logging for PCMCIA Read/Write Support

24

At a Glance

Introduction

This chapter describes the instruction DLOG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	120
Representation	121
Parameter Description	122
Run Time Error Handling	123

Short Description

Function Description

Note: This instruction is only available with the PLC family TSX Compact.

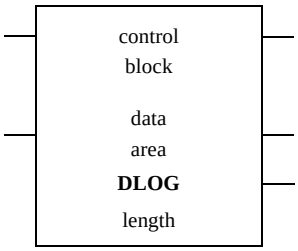
PCMCIA read and write support consists of a configuration extension to be implemented using a DLOG instruction. The DLOG instruction provides the facility for an application to copy data to a PCMCIA flash card, copy data from a PCMCIA flash card, erase individual memory blocks on a PCMCIA flash card, and to erase an entire PCMCIA flash card. The data format and the frequency of data storage are controlled by the application.

Note: The DLOG instruction will only operate with PCMCIA linear flash cards that use AMD flash devices.

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 = DLOG operation enabled, it should remain ON until the operation has completed successfully or an error has occurred.
Middle input	0x, 1x	None	ON = stops the currently active operation
control block (top node)	4x	INT, UINT	First of five contiguous registers in the DLOG control block
data area (middle node)	4x	INT, UINT	First 4x register in a data area used for the source or destination of the specified operation
length (bottom node)		INT, UINT	Maximum number of registers reserved for the data area, range: 0 ... 100.
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = error during DLOG operation (operation terminated unsuccessfully)
Bottom output	0x	None	ON = DLOG operation finishes successfully (operation successful)

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of five contiguous registers in the DLOG control block.
The control block defines the function of the DLOG command, the PCMCIA flash card window and offset, a return status word, and a data word count value.

Register	Function	Content
Displayed	Error Status	Displays DLOG errors in HEX values
First implied	Operation Type	1 = Write to PCMCIA Card 2 = Read to PCMCIA Card 3 = Erase One Block 4 = Erase Entire Card Content
Second implied	Window (Block Identifier)	This register identifies a particular block (PCMCIA memory window) located on the PCMCIA card (1 block=128k bytes) The number of blocks are dependent on the memory size of the PCMCIA card. (e.g.. 0 ... 31 Max. for a 4Meg PCMCIA card).
Third implied	Offset (Byte Address within the Block)	Particular range of bytes located within a particular block on the PCMCIA card. Range: 1 ... 128k bytes
Fourth implied	Count	Number of 4x registers to be written or read to the PCMCIA card. Range: 0 ... 100.

Note: PCMCIA Flash Card address are address on a Window:Offset basis. Windows have a set size of 128k bytes (65 535 words (16-bit values)). No Write or Read operation can cross the boundary from one window to the next. Therefore, offset (third implied register) plus length (fourth implied register) must always be less or equal to 128k bytes (65 535 words).

Data Area (Middle Node)

The 4x register entered in the middle node is the first register in a contiguous block of 4x word registers, that the DLOG instruction will use for the source or destination of the operation specified in the top node's control block.

Operation	State Ram Reference	Function
Write	4x	Source Address
Read	4x	Destination Address
Erase Block	none	None
Erase Card	none	None

Length (Bottom Node)

The integer value entered in the bottom node is the length of the data area, i.e., the maximum number of words (registers) allowed in a transfer to/from the PCMCIA flash card. The length can range from 0 ... 100.

Run Time Error Handling

Error Codes

The displayed register of the control block contains the following DLOG errors in Hex-code.
Hex Error Codes DLOG

Error Code in Hex	Content
1	The count parameter of the control block > the DLOG block length during a WRITE operation (01)
2	PCMCIA card operation failed when intially started (write/read/erase)
3	PCMCIA card operation failed during execution (write/read/erase)

DRUM: DRUM Sequencer

25

At a Glance

Introduction

This chapter describes the instruction DRUM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	126
Representation	127
Parameter Description	127

Short Description

Function Description

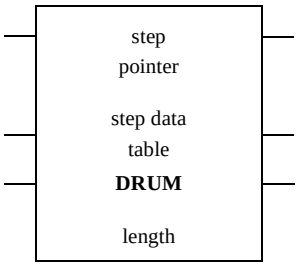
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 53*".

The DRUM instruction operates on a table of 4x registers containing data representing each step in a sequence. The number of registers associated with this step data table depends on the number of steps required in the sequence. You can pre-allocate registers to store data for each step in the sequence, thereby allowing you to add future sequencer steps without having to modify application logic. DRUM incorporates an output mask that allows you to selectively mask bits in the register data before writing it to coils. This is particularly useful when all physical sequencer outputs are not contiguous on the output module. Masked bits are not altered by the DRUM instruction, and may be used by logic unrelated to the sequencer.

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 DRUM sequencer
Middle input	0x, 1x	None	ON = step pointer increments to next step
Bottom input	0x, 1x	None	ON = reset step pointer to 0
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	Echos state of the top input
Middle output	0x	None	ON = step pointer value = length
Bottom output	0x	None	ON = Error

Parameter Description

Step Pointer (Top Node)

The 4x register entered in the top node stores the current step number. The value in this register is referenced by the DRUM instruction each time it is solved. If the middle input to the block is ON, the contents of the register in the top node are incremented to the next step in the sequence before the block is solved.

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 six registers in the step data table hold constant and variable data required to solve the block:

Register	Name	Content
Displayed	masked output data	Loaded by DRUM each time the block is solved; contains the contents of the current step data register masked with the outputmask register
First implied	current step data	Loaded by DRUM each time the block is solved; contains data from the step pointer, causes the block logic to automatically calculate register offsets when accessing step data in the step data table
Second implied	output mask	Loaded by user before using the block, DRUM will not alter output mask contents during logic solve; contains a mask to be applied to the data for each sequencer step
Third implied	machine ID number	Identifies DRUM/ICMP blocks belonging to a specific machine configuration; value range: 0 ... 9 999 (0 = block not configured); all blocks belonging to same machine configuration have the same machine ID number
Fourth implied	profile ID number	Identifies profile data currently loaded to the sequencer; value range: 0... 9 999 (0 = block not configured); all blocks with the same machine ID number must have the same profile ID number
Fifth 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 greater or less than the table length in the bottom node

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 + 6. The length must be greater or equal to the value placed in the steps used register in the middle node.

DV16: Divide 16 Bit

26

At a Glance

Introduction

This chapter describes the instruction DV16.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	130
Representation	130
Example	131

Short Description

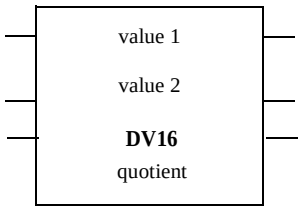
Function Description

The DV16 instruction performs a signed or unsigned division on the 16-bit values in the top and middle nodes (value 1 / value 2), then posts the quotient and remainder in two contiguous 4x 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 / value 2
Middle input	0x, 1x	None	OFF = decimal remainder ON = fractional remainder
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Dividend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in two contiguous registers (displayed for high-order half, implied for low-order half)
value 2 (middle node)	3x, 4x	INT, UINT	Divisor, can be displayed explicitly as an integer (range 1 ... 65 535, enter e.g. #65535) or stored in a register
quotient (bottom node)	4x	INT, UINT	First of two contiguous holding registers: displayed: result of division implied: remainder (either a decimal or a fraction, depending on the state of middle input)
Top output	0x	None	ON = Divide operation completed successfully
Middle output	0x	None	ON = overflow: quotient > 65 535 in unsigned operation -32 768 > quotient > 32 767 in signed operation
Bottom output	0x	None	ON = value 2 = 0

Example

Quotient of Instruction DV16

The state of the middle input indicates whether the remainder will be expressed as a decimal or as a fraction. For example, if value 1 = 8 and value 2 = 3, the decimal remainder (middle input OFF) is 6666; the fractional remainder (middle input ON) is 2.

EMTH: Extended Math

27

At a Glance

Introduction

This chapter describes the instruction EMTH.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	134
Representation	135
Parameter Description	136
Floating Point EMTH Functions	138

Short Description

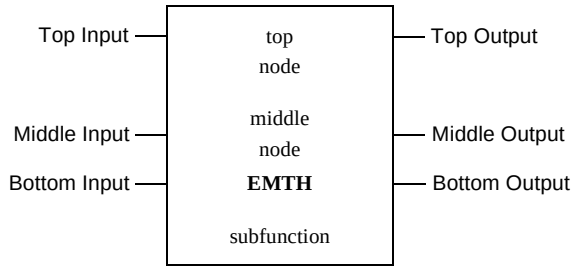
Function Description

This instruction accesses a library of double-precision math, square root and logarithm calculations and floating point (FP) arithmetic functions.
The EMTH instruction allows you to select from a library of 38 extended math functions. Each of the functions has an alphabetical indicator of variable subfunctions that can be selected from a pulldown menu in your panel software and appears in the bottom node. EMTH control inputs and outputs are function-dependent.

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	Depends on the selected EMTH function, see "*Inputs, Outputs and Bottom Node, p. 136*"
Middle input	0x, 1x	None	Depends on the selected EMTH function
Bottom input	0x, 1x	None	Depends on the selected EMTH function
top node	3x, 4x	DINT, UDINT, REAL	Two consecutive registers, usually 4x holding registers but, in the integer math cases, either 4x or 3x registers
middle node	4x	DINT, UDINT, REAL	Two, four, or six consecutive registers, depending on the function you are implementing.
subfunction (bottom node)			An alphabetical lable, identifing the EMTH function, see "*Inputs, Outputs and Bottom Node, p. 136*"
Top output	0x	None	Depends on the selected EMTH function, see "*Inputs, Outputs and Bottom Node, p. 136*"
Middle output	0x	None	Depends on the selected EMTH function
Bottom output	0x	None	Depends on the selected EMTH function

Parameter Description

Inputs, Outputs and Bottom Node

The implementation of inputs to and outputs from the block depends on the EMTH subfunction you select. An alphabetical indicator of variable subfunctions appears in the bottom node identifing the EMTH function you have chosen from the library.

You will find the EMTH subfunctions in the following tables:
- Double Precision Math
- Integer Math
- Floating Point Math

Subfunctions for Double Precision Math

Double Precision Math

EMTH Function	Subfunction	Active Inputs	Active Outputs
Addition	ADDDP	Top	Top and Middle
Subtraction	SUBDP	Top	Top, Middle and Bottom
Multiplication	MULDP	Top	Top and Middle
Division	DIVDP	Top and Middle	Top, Middle and Bottom

Subfunctions for Integer Math

Integer Math

EMTH Function	Subfunction	Active Inputs	Active Outputs
Square root	SQRT	Top	Top and Middle
Process square root	SQRTP	Top	Top and Middle
Logarithm	LOG	Top	Top and Middle
Antilogarithm	ANLOG	Top	Top and Middle

Subfunctions for Floating Point Math

Floating Point Math (See *Floating Point EMTH Functions, p. 138*)

EMTH Function	Subfunction	Active Inputs	Active Outputs
Integer-to-FP conversion	CNVIF	Top	Top
Integer + FP	ADDIF	Top	Top
Integer - FP	SUBIF	Top	Top
Integer x FP	MULIF	Top	Top
Integer / FP	DIVIF	Top	Top
FP - Integer	SUBFI	Top	Top
FP / Integer	DIVFI	Top	Top
Integer-FP comparison	CMPIF	Top	Top
FP-to-Integer conversion	CNVFI	Top	Top and Middle
Addition	ADDFP	Top	Top
Subtraction	SUBFP	Top	Top
Multiplication	MULFP	Top	Top
Division	DIVFP	Top	Top
Comparison	CMPFP	Top	Top, Middle and Bottom
Square root	SQRFP	Top	Top
Change sign	CHSIN	Top	Top
Load Value of p	PI	Top	Top
Sine in radians	SINE	Top	Top
Cosine in radians	COS	Top	Top
Tangent in radians	TAN	Top	Top
Arcsine in radians	ARSIN	Top	Top
Arccosine in radians	ARCOS	Top	Top
Arctangent in radians	ARTAN	Top	Top
Radians to degrees	CNVRD	Top	Top
Degrees to radians	CNVDR	Top	Top
FP to an integer power	POW	Top	Top
Exponential function	EXP	Top	Top
Natural log	LNFP	Top	Top
Common log	LOGFP	Top	Top
Report errors	ERLOG	Top	Top and Middle

Floating Point EMTH Functions

Use of Floating Point Functions

To make use of the floating point (FP) capability, the four-digit integer values used in standard math instructions must be converted to the IEEE floating point format. All calculations are then performed in FP format and the results must be converted back to integer format.

The IEEE Floating Point Standard

EMTH floating point functions require values in 32-bit IEEE floating point format. Each value has two registers assigned to it, the eight most significant bits representing the exponent and the other 23 bits (plus one assumed bit) representing the mantissa and the sign of the value.

Note: Floating point calculations have a mantissa precision of 24 bits, which guarantees the accuracy of the seven most significant digits. The accuracy of the eighth digit in an FP calculation can be inexact.

It is virtually impossible to recognize a FP representation on the programming panel. Therefore, all numbers should be converted back to integer format before you attempt to read them.

Dealing with Negative Floating Point Numbers

Standard integer math calculations do not handle negative numbers explicitly. The only way to identify negative values is by noting that the SUB function block has turned the bottom output ON.
If such a negative number is being converted to floating point, perform the Integer-to-FP conversion (EMTH subfunction CNVIF), then use the Change Sign function (EMTH subfunction CHSIN) to make it negative prior to any other FP calculations.

EMTH-ADDDP: Double Precision Addition

28

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ADDDP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	140
Representation	140
Parameter Description	141

Short Description

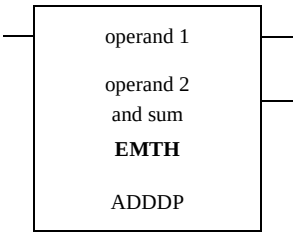
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 136*)".

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 = adds operands and posts sum in designated registers
operand 1 (top node)	4x	DINT, UDINT	Operand 1 (first of two contiguous registers)
operand 2 and sum (middle node)	4x	DINT, UDINT	Operand 2 and sum (first of six contiguous registers)
ADDDP (bottom node)			Selection of the subfunction ADDDP
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = operand out of range or invalid

Parameter Description

Operand 1 (Top Node)

The first of two contiguous 4x registers is entered in the top node. The second 4x register is implied. Operand 1 is stored here.

Register	Content
Displayed	Register stores the low-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999

Operand 2 and Sum (Middle Node)

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

Register	Content
Displayed	Register stores the low-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999
Second implied	The value stored in this register indicates whether an overflow condition exists (a value of 1 = overflow)
Third implied	Register stores the low-order half of the double precision sum.
Fourth implied	Register stores the high-order half of the double precision sum.
Fifth implied	Register is not used in the calculation but must exist in state RAM

EMTH-ADDFP: Floating Point Addition

29

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ADDFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	144
Representation	144
Parameter Description	145

Short Description

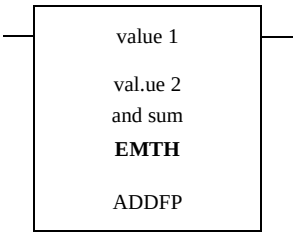
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP addition
value 1 (top node)	4x	REAL	Floating point value 1 (first of two contiguous registers)
value 2 and sum (middle node)	4x	REAL	Floating point value 2 and the sum (first of four contiguous registers)
ADDFP (bottom node)			Selection of the subfunction ADDFP
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value 1 (Top Node)

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

Register	Content
Displayed First implied	Registers store the FP value 1.

Floating Point Value 2 and Sum (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers store the FP value 2.
Second implied Third implied	Registers store the sum of the addition in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-ADDIF: Integer + Floating Point Addition

30

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ADDIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	148
Representation	148
Parameter Description	149

Short Description

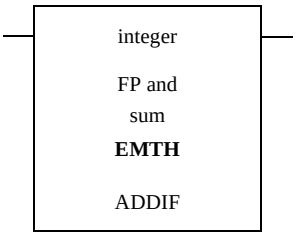
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 integer + FP operation
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
FP and sum (middle node)	4x	REAL	FP value and sum (first of four contiguous registers)
ADDIF (bottom node)			Selection of the subfunction ADDIF
Top output	0x	None	ON = operation successful

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value to be added to the FP value is stored here.

FP Value and Sum (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers store the FP value to be added in the operation.
Second implied Third implied	The sum is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-ANLOG: Base 10 Antilogarithm

31

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ANLOG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	152
Representation	152
Parameter Description	153

Short Description

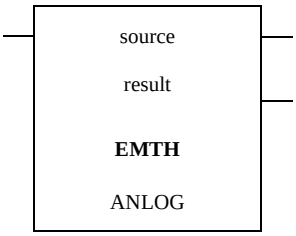
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 136*)".

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 antilog(x) operation
source (top node)	3x, 4x	INT, UINT	Source value
result (middle node)	4x	DINT, UDINT	Result (first of two contiguous registers)
ANLOG (bottom node)			Selection of the subfunction ANLOG
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = an error or value out of range

Parameter Description

Source Value (Top Node)

The top node is a single 4x holding register or 3x input register. The source value, i.e. the value on which the antilog calculation will be performed, is stored here in the fixed decimal format **1.234**. It must be in the range 0 ... 7 999, representing a source value up to a maximum of 7.999.

Result (Middle Node)

The first of two contiguous 4x registers is entered in the middle node. The second register is implied. The result of the antilog calculation is posted here in the fixed decimal format **12345678**:

Register	Content
Displayed	Most significant bits
First implied	Least significant bits

The largest antilog value that can be calculated is 99770006 (9977 posted in the displayed register and 0006 posted in the implied register).

EMTH-ARCOS: Floating Point Arc Cosine of an Angle (in Radians)

32

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ARCOS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	156
Representation	156
Parameter Description	157

Short Description

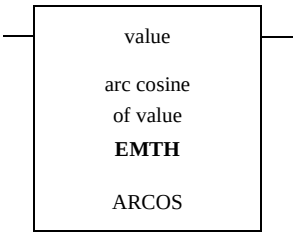
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates arc cosine of the value
value (top node)	4x	REAL	FP value indicating the cosine of an angle (first of two contiguous registers)
arc cosine of value (middle node)	4x	REAL	Arc cosine in radians of the value in the top node (first of four contiguous registers)
ARCOS (bottom node)			Selection of the subfunction ARCOS
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the cosine of an angle between 0 ... p radians is stored here. This value must be in the range of -1.0 ... +1.0;

If the value is not in the range of -1.0 ... +1.0:

- The arc cosine is not computed
- An invalid result is returned
- An error is flagged in the EMTH-ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function

Arc Cosine of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The arc cosine in radians of the FP value in the top node is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-ARSIN: Floating Point Arcsine of an Angle (in Radians)

33

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ARSIN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	160
Representation	160
Parameter Description	161

Short Description

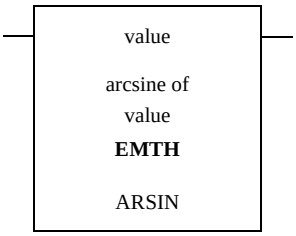
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the arcsine of the value
value (top node)	4x	REAL	FP value indicating the sine of an angle (first of two contiguous registers)
arcsine of value (middle node)	4x	REAL	Arcsine of the value in the top node (first of four contiguous registers)
ARSIN (bottom node)			Selection of the subfunction ARSIN
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the sine of an angle between $-\pi/2$ ... $\pi/2$ radians is stored here. This value (the sine of an angle) must be in the range of -1.0 ... +1.0;

If the value is not in the range of -1.0 ... +1.0:

- The arcsine is not computed
- An invalid result is returned
- An error is flagged in the EMTH-ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function

Arcsine of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The arcsine of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-ARTAN: Floating Point Arc Tangent of an Angle (in Radians)

34

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-ARTAN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	164
Representation	164
Parameter Description	165

Short Description

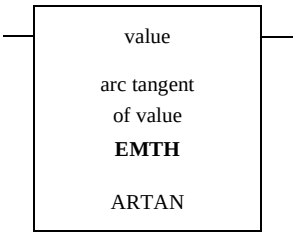
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the arc tangent of the value
value (top node)	4x	REAL	FP value indicating the tangent of an angle (first of two contiguous registers)
arc tangent of value (middle node)	4x	REAL	Arc tangent of the value in the top node (first of four contiguous registers)
ARTAN (bottom node)			Selection of the subfunction ARTAN
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the tangent of an angle between -π/2 ... π/2 radians is stored here. Any valid FP value is allowed.;

Arc Tangent of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The arc tangent in radians of the FP value in the top node is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-CHSIN: Changing the Sign of a Floating Point Number

35

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CHSIN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	168
Representation	168
Parameter Description	169

Short Description

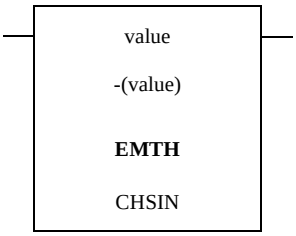
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = changes the sign of FP value
value (top node)	4x	REAL	Floating point value (first of two contiguous registers)
-(value) (middle node)	4x	REAL	Floating point value with changed sign (first of four contiguous registers)
CHSIN (bottom node)			Selection of the subfunction CHSIN
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value (Top Node)

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

Register	Content
Displayed First implied	The FP value whose sign will be changed is stored here.

Floating Point Value with changed sign (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The top node FP value with changed sign is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-CMPFP: Floating Point Comparison

36

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CMPFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	172
Representation	172
Parameter Description	173

Short Description

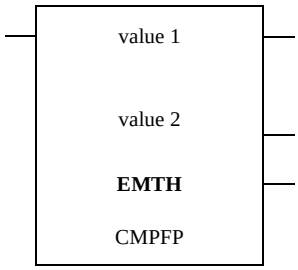
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 comparison
value 1 (top node)	4x	DINT, UDINT	First floating point value (first of two contiguous registers)
value 2 (middle node)	4x	REAL	Second floating point value (first of four contiguous registers)
CMPFP (bottom node)			Selection of the subfunction CMPFP
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = value 1 > value 2 when the bottom output is OFF
Bottom output	0x	None	ON = value 1 < value 2 when the middle output is OFF

Parameter Description

Value 1 (Top Node)

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

Register	Content
Displayed First implied	The first FP value (value 1) to be compared is stored here.

Value 2 (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	The second FP value (value 2) to be compared is stored here.
Second implied Third implied	Registers are not used but their allocation in state RAM is required.

Middle and Bottom Output

When EMTH function CMPFP compares its two FP values, the combined states of the middle and the bottom output indicate their relationship:

Middle Output	Bottom Output	Relationship
ON	OFF	value 1 > value 2
OFF	ON	value 1 < value 2
ON	ON	value 1 = value 2

EMTH-CMPIF: Integer-Floating Point Comparison

37

At a Glance

Introduction

This chapter describes the EMTH EMTH subfunction EMTH-CMPIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	176
Representation	176
Parameter Description	177

Short Description

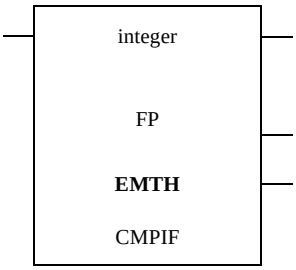
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 comparison
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
FP (middle node)	4x	REAL	Floating point value (first of four contiguous registers)
CMPIF (bottom node)			Selection of the subfunction CMPIF
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = integer > FP when the bottom output is OFF
Bottom output	0x	None	ON = integer < FP when the middle output is OFF

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value to be compared is stored here.

Floating Point Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	The FP value to be compared is stored here.
Second implied Third implied	Registers are not used but their allocation in state RAM is required.

Middle and Bottom Output

When EMTH function CMPIF compares its integer and FP values, the combined states of the middle and the bottom output indicate their relationship:

Middle Output	Bottom Output	Relationship
ON	OFF	integer > FP
OFF	ON	integer < FP
ON	ON	integer = FP

EMTH-CNVDR: Floating Point Conversion of Degrees to Radians

38

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CNVDR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	180
Representation	180
Parameter Description	181

Short Description

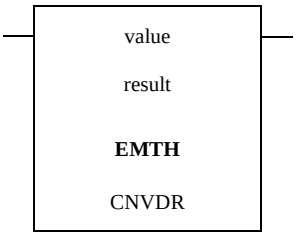
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 conversion of value 1 to value 2 (result)
value (top node)	4x	REAL	Value in FP format of an angle in degrees (first of two contiguous registers)
result (middle node)	4x	REAL	Converted result (in radians) in FP format (first of four contiguous registers)
CNVDR (bottom node)			Selection of the subfunction CNVDR
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	The value in FP format (See *The IEEE Floating Point Standard, p. 138*) of an angle in degrees is stored here.

Result in Radians (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The converted result in FP format (See *The IEEE Floating Point Standard, p. 138*) of the top-node value (in radians) is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-CNVFI: Floating Point to Integer Conversion

39

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CNVFI.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	184
Representation	184
Parameter Description	185
Runtime Error Handling	185

Short Description

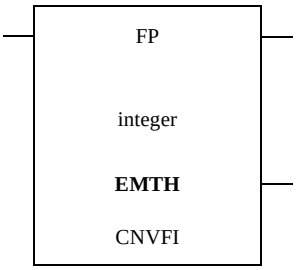
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP to integer conversion
FP (top node)	4x	REAL	Floating point value to be converted (first of two contiguous registers)
integer (middle node)	4x	DINT, UDINT	Integer value (first of four contiguous registers)
CNVFI (bottom node)			Selection of the subfunction CNVFI
Top output	0x	None	ON = operation successful
Bottom output	0x	None	OFF = positive integer value ON = negative integer value

Parameter Description

Integer Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The double precision integer result of the conversion is stored here. This value should be the largest integer value possible that is ≤ the FP value. For example, the FP value 3.5 is converted to the integer value 3, while the FP value -3.5 is converted to the integer value -4.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

Runtime Error Handling

Runtime Errors

If the resultant integer is too large for double precision integer format (> 99 999 999), the conversion still occurs but an error is logged in the EMTH_ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function.

EMTH-CNVIF: Integer-to-Floating Point Conversion

40

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CNVIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	188
Representation	188
Parameter Description	189
Runtime Error Handling	189

Short Description

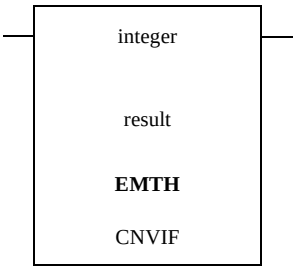
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP to integer conversion
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
result (middle node)	4x	REAL	Result (first of four contiguous registers)
CNVIF (bottom node)			Selection of the subfunction CNVIF
Top output	0x	None	ON = operation successful

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value to be converted to 32-bit FP format (See *The IEEE Floating Point Standard, p. 138*) is stored here.

Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied.

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The FP result of the conversion is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

Runtime Error Handling

Runtime Errors

If an invalid integer value (> 9 999) is entered in either of the two top-node registers, the FP conversion will be performed but an error will be reported and logged in the EMTH_ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function. The result of the conversion may not be correct.

EMTH-CNVRD: Floating Point Conversion of Radians to Degrees

41

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-CNVRD.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	192
Representation	192
Parameter Description	193

Short Description

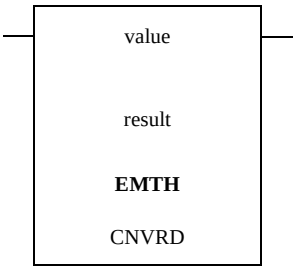
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 conversion of value 1 to value 2
value (top node)	4x	REAL	Value in FP format of an angle in radians (first of two contiguous registers)
result (middle node)	4x	REAL	Converted result (in degrees) in FP format (first of four contiguous registers)
CNVRD (bottom node)			Selection of the subfunction CNVRD
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	The value in FP format (See *The IEEE Floating Point Standard, p. 138*) of an angle in radians is stored here.

Result in Degrees (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied.

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The converted result in FP format (See *The IEEE Floating Point Standard, p. 138*) of the top-node value (in degrees) is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-COS: Floating Point Cosine of an Angle (in Radians)

42

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-COS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	196
Representation	196
Parameter Description	197

Short Description

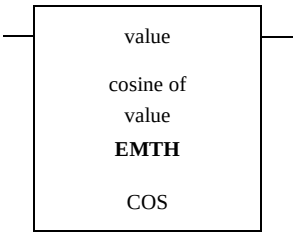
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the cosine of the value
value (top node)	4x	REAL	FP value indicating the value of an angle in radians (first of two contiguous registers)
cosine of value (middle node)	4x	REAL	Cosine of the value in the top node (first of four contiguous registers)
COS (bottom node)			Selection of the subfunction COS
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the value of an angle in radians is stored here. The magnitude of this value must be < 65 536.0.

If the magnitude of this value is ≥ 65 536.0:

- The cosine is not computed
- An invalid result is returned
- An error is flagged in the EMTH-ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function

Cosine of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The cosine of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-DIVDP: Double Precision Division

43

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-DIVDP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	200
Representation	200
Parameter Description	201
Runtime Error Handling	201

Short Description

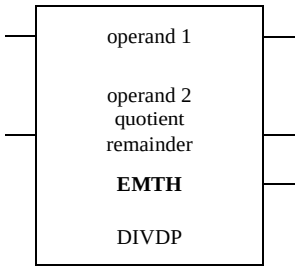
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 136*)".

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 = operand 1 divided by operand 2 and result posted in designated registers."
Middle input	0x, 1x	None	ON = decimal remainder OFF = fractional remainder
operand 1 top node	4x	DINT, UDINT	Operand 1 (first of two contiguous registers)
operand 2 quotient remainder middle node	4x	DINT, UDINT	Operand 2, quotient and remainder (first of six contiguous registers)
DIVDP (bottom node)			Selection of the subfunction DIVDP"
Top output	0x	None	ON = operation successful"
Middle output	0x	None	ON = an operand out of range or invalid
Bottom output	0x	None	ON = operand 2 = 0

Parameter Description

Operand 1 (Top Node)

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

Register	Content
Displayed	Low-order half of operand 1 is stored here.
First implied	High-order half of Operand 1 is stored here.

Each register holds a value in the range 0000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999.

Operand 2, Quotient and Remainder (Middle Node)

The first of six contiguous 4x registers is entered in the middle node. The remaining five registers are implied

Register	Content
Displayed	Register stores the low-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999.
Second implied Third implied	Registers store an eight-digit quotient.
Fourth implied Fifth implied	Registers store the remainder. ● If it is expressed as a **decimal**, it is four digits long and only the fourth implied register is used. ● If it is expressed as a **fraction**, it is eight digits long and both registers are used

Runtime Error Handling

Runtime Errors

Since division by 0 is illegal, a 0 value causes an error, an error trapping routine sets the remaining middle-node registers to 0000 and turns the bottom output ON.

EMTH-DIVFI: Floating Point Divided by Integer

44

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-DIVFI.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	204
Representation	204
Parameter Description	205

Short Description

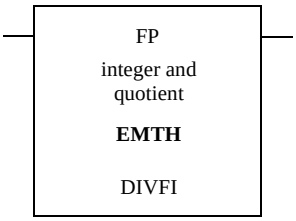
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP / integer operation
FP (top node)	4x	REAL	Floating point value (first of two contiguous registers)
integer and quotient (middle node)	4x	DINT, UDINT	Integer value and quotient (first of four contiguous registers)
DIVFI (bottom node)			Selection of the subfunction DIVFI
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value (Top Node)

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

Register	Content
Displayed First implied	The FP value to be divided by the integer value is stored here.

Integer Value and Quotient (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied.

Register	Content
Displayed First implied	The double precision integer value that divides the FP value is posted here.
Second implied Third implied	The quotient is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-DIVFP: Floating Point Division

45

At a Glance

Introduction

This chapter describes the instrcution EMTH-DIVFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	208
Representation	208
Parameter Description	209

Short Description

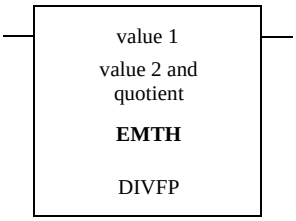
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 value 1 / value 2 operation
value 1 (top node)	4x	REAL	Floating point value 1 (first of two contiguous registers)
value 2 and quotient (middle node)	4x	REAL	Floating point value 2 and the quotient (first of four contiguous registers)
DIVFP (bottom node)			Selection of the subfunction DIVFP
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value 1 (Top Node)

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

Register	Content
Displayed First implied	FP value 1, which will be divided by the value 2, is stored here.

Floating Point Value 2 and Quotient (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	FP value 2, the value by which value 1 is divided, is stored here
Second implied Third implied	The quotient is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-DIVIF: Integer Divided by Floating Point

46

At a Glance

Introduction

This chapter describes the instrcution EMTH-DIVIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	212
Representation	212
Parameter Description	213

Short Description

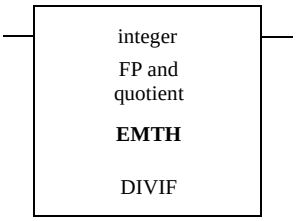
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 integer / FP operation
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
FP and quotient (middle node)	4x	REAL	FP value and quotient (first of four contiguous registers)
DIVIF (bottom node)			Selection of the subfunction DIVIF
Top output	0x	None	ON = operation successful

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value to be divided by the FP value is stored here.

Floating Point Value and Quotient (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied.

Register	Content
Displayed First implied	The FP value to be divided in the operation is posted here.
Second implied Third implied	The quotient is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-ERLOG: Floating Point Error Report Log

47

At a Glance

Introduction

This chapter describes the instrcution EMTH-ERLOG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	216
Representation	216
Parameter Description	217

Short Description

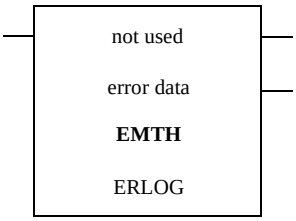
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = retrieves a log of error types since last invocation
not used (top node)	4x	INT, UINT, DINT, UDINT, REAL	Not used in the operation (first of two contiguous registers)
error data (middle node)	4x	INT, UINT, DINT, UDINT, REAL	Error log register (first of four contiguous registers)
ERLOG (bottom node)			Selection of the subfunction ERLOG
Top output	0x	None	ON = retrieval successful
Middle output	0x	None	ON = nonzero values in error log register OFF = all zeros in error log register

Parameter Description

Not used (Top Node)

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

Register	Content
Displayed First implied	These two registers are not used in the operation but their allocation in state RAM is required.

Error Data (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied.

Register	Content
Displayed First implied	These two registers are not used but their allocation in state RAM is required.
Second implied	Error log register, see table (See *Error Log Register, p. 217*).
Third implied	This register has all its bits cleared to zero.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since these registers must be allocated but none are used.

Error Log Register

Usage of error log register:

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

Bit	Function
1 - 8	Function code of last error logged
9 - 11	Not used
12	Integer/FP conversion error
13	Exponential function power too large
14	Invalid FP value or operation
15	FP overflow
16	FP underflow

If the bit is set to 1, then the specific error condition exists for that bit.

EMTH-EXP: Floating Point Exponential Function

48

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-EXP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	220
Representation	220
Parameter Description	221

Short Description

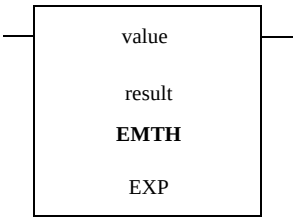
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates exponential function of the value
value (top node)	4x	REAL	Value in FP format (first of two contiguous registers)
result (middle node)	4x	REAL	Exponential of the value in the top node (first of four contiguous registers)
EXP (bottom node)			Selection of the subfunction EXP
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	A value in FP format (See *The IEEE Floating Point Standard, p. 138*) in the range -87.34 ... +88.72 is stored here. If the value is out of range, the result will either be 0 or the maximum value. No error will be flagged.

Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	These registers are not used but their allocation in state RAM is required
Second implied Third implied	The exponential of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-LNFP: Floating Point Natural Logarithm

49

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-LNFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	224
Representation	224
Parameter Description	225

Short Description

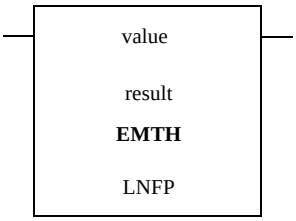
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the natural log of the value
value (top node)	4x	REAL	Value > 0 in FP format (first of two contiguous registers)
result (middle node)	4x	REAL	Natural logarithm of the value in the top node (first of four contiguous registers)
LNFP (bottom node)			Selection of the subfunction LNFP
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	A value > 0 is stored here in FP format (See *The IEEE Floating Point Standard, p. 138*). If the value ≤ 0, an invalid result will be returned in the middle node and an error will be logged in the EMTH-ERLOG function.

Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	These registers are not used but their allocation in state RAM is required
Second implied Third implied	The natural logarithm of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-LOG: Base 10 Logarithm

50

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-LOG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	228
Representation	228
Parameter Description	229

Short Description

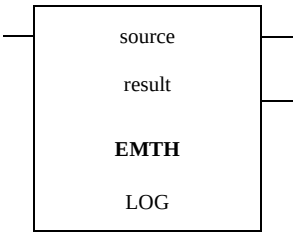
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 136*)".

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 log(x) operation
source (top node)	3x, 4x	DINT, UDINT	Source value (first of two contiguous registers)
result (middle node)	4x	INT, UINT	Result
LOG (bottom node)			Selection of the subfunction LOG
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = an error or value out of range

Parameter Description

Source Value (Top Node)

The first of two contiguous 3x or 4x registers is entered in the top node. The second register is implied. The source value upon which the log calculation will be performed is stored in these registers.
If you specify a **4x register**, the source value may be in the range 0 ... 99 999 99:

Register	Content
Displayed	The high-order half of the value is stored here.
First implied	The low-order half of the value is stored here.

If you specify a **3x register**, the source value may be in the range 0 ... 9 999:

Register	Content
Displayed	The source value upon which the log calculation will be performed is stored here
First implied	This register is required but not used.

Result (Middle Node)

The middle node contains a single 4x holding register where the result of the base 10 log calculation is posted. The result is expressed in the fixed decimal format **1.234**, and is truncated after the third decimal position.
The largest result that can be calculated is 7.999, which would be posted in the middle register as 7999.

EMTH-LOGFP: Floating Point Common Logarithm

51

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-LOGFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	232
Representation	232
Parameter Description	233

Short Description

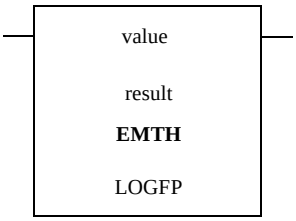
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the common log of the value
value (top node)	4x	REAL	Value > 0 in FP format (first of two contiguous registers)
result (middle node)	4x	REAL	Common logarithm of the value in the top node (first of four contiguous registers)
LOGFP (bottom node)			Selection of the subfunction LOGFP
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	A value > 0 is stored here in FP format (See *The IEEE Floating Point Standard, p. 138*). If the value ≤ 0, an invalid result will be returned in the middle node and an error will be logged in the EMTH-ERLOG function.

Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	These registers are not used but their allocation in state RAM is required
Second implied Third implied	The common logarithm of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-MULDP: Double Precision Multiplication

52

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-MULDP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	236
Representation	236
Parameter Description	237

Short Description

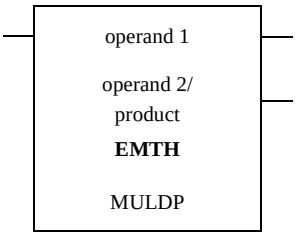
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 136*)".

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 = operand 1 x operand 2 and product posted in designated registersoperand 1
operand 1 (top node)	4x	DINT, UDINT	Operand 1 (first of two contiguous registers)
operand 2 / product (middle node)	4x	DINT, UDINT	Operand 2 and product (first of six contiguous registers)
MULDP (bottom node)			Selection of the subfunction MULDP
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON = operand out of range

Parameter Description

Operand 1 (Top Node)

The first of two contiguous 4x registers is entered in the top node. The second 4x register is implied. Operand 1 is stored here.

Register	Content
Displayed	Register stores the low-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999

Operand 2 and Product (Middle Node)

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

Register	Content
Displayed	Register stores the low-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 2, respectively, for a combined double precision value in the range 0 ... 99 999 999
Second implied Third implied Fourth implied Fifth implied	These registers store the double precision product in the range 0 ... 9 999 999 999 999 999

EMTH-MULFP: Floating Point Multiplication

53

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-MULFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	240
Representation	240
Parameter Description	241

Short Description

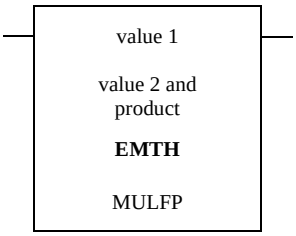
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP multiplication
value 1 (top node)	4x	REAL	Floating point value 1 (first of two contiguous registers)
value 2 and product (middle node)	4x	REAL	Floating point value 2 and the product (first of four contiguous registers)
MULFP (bottom node)			Selection of the subfunction MULFP
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value 1 (Top Node)

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

Register	Content
Displayed First implied	FP value 1 in the multiplication operation is stored here.

Floating Point Value 2 and Product (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	FP value 2 in the multiplication operation is stored here.
Second implied Third implied	The product of the multiplication is stored here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-MULIF: Integer x Floating Point Multiplication

54

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-MULIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	244
Representation	244
Parameter Description	245

Short Description

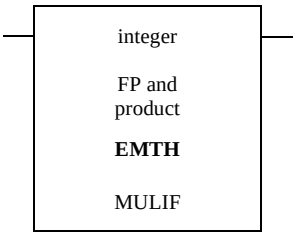
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 integer x FP operation
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
FP and product (middle node)	4x	REAL	FP value and product (first of four contiguous registers)
MULIF (bottom node)			Selection of the subfunction MULIF
Top output	0x	None	ON = operation successful

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value to be multiplied by the FP value is stored here.

FP Value and Product (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	The FP value to be multiplied in the operation is stored here.
Second implied Third implied	The product of the multiplication is stored here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-PI: Load the Floating Point Value of "Pi"

55

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-PI (Load the Floating Point Value of π).

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	248
Representation	248
Parameter Description	249

Short Description

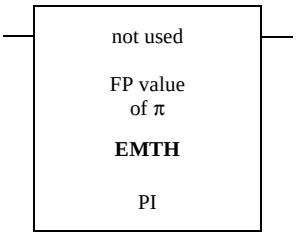
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = loads FP value of π to middle node register
not used (top node)	4x	REAL	First of two contiguous registers
FP value of π (middle node)	4x	REAL	FP value of π (first of four contiguous registers)
PI (bottom node)			Selection of the subfunction PI
Top output	0x	None	ON = operation successful

Parameter Description

Not used (Top Node)

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

Register	Content
Displayed First implied	These registers are not used but their allocation in state RAM is required.

Floating Point Value of π (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed First implied	These registers are not used but their allocation in state RAM is required.
Second implied Third implied	The FP value of π is posted here.

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-POW: Raising a Floating Point Number to an Integer Power

56

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-POW.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	252
Representation	252
Parameter Description	253

Short Description

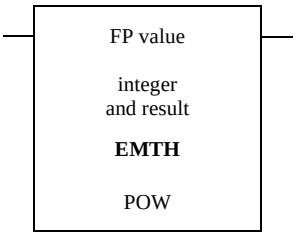
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates FP value raised to the power of integer value
FP value (top node)	4x	REAL	FP value (first of two contiguous registers)
integer and result (middle node)	4x	INT, UINT	Integer value and result (first of four contiguous registers)
POW (bottom node)			Selection of the subfunction POW
Top output	0x	None	ON = operation successful

Parameter Description

FP Value (Top Node)

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

Register	Content
Displayed First implied	The FP value to be raised to the integer power is stored here.

Integer and Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied:

Register	Content
Displayed	The bit values in this register must all be cleared to zero.
First implied	An integer value representing the power to which the top-node value will be raised is stored here.
Second implied Third implied	The result of the FP value being raised to the power of the integer value is stored here.

EMTH-SINE: Floating Point Sine of an Angle (in Radians)

57

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SINE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	256
Representation	256
Parameter Description	257

Short Description

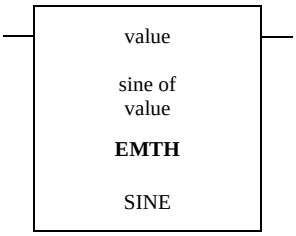
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the sine of the value
value (top node)	4x	REAL	FP value indicating the value of an angle in radians (first of two contiguous registers)
sine of value (middle node)	4x	REAL	Sine of the value in the top node (first of four contiguous registers)
SINE (bottom node)			Selection of the subfunction SINE
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the value of an angle in radians is stored here. The magnitude of this value must be < 65 536.0.

If the magnitude is ≥ 65 536.0:

- The sine is not computed
- An invalid result is returned
- An error is flagged in the EMTH-ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function

Sine of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The sine of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-SQRFP: Floating Point Square Root

58

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SQRFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	260
Representation	260
Parameter Description	261

Short Description

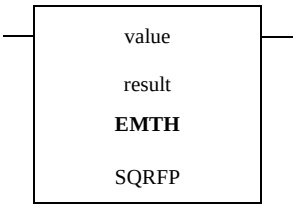
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 square root on FP value
value (top node)	4x	REAL	Floating point value (first of two contiguous registers)
result (middle node)	4x	REAL	Result in FP format (first of four contiguous registers)
SQRFP (bottom node)			Selection of the subfunction SQRFP
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value (Top Node)

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

Register	Content
Displayed First implied	The FP value on which the square root operation is performed is stored here.

Result (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The result of the square root operation is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

EMTH-SQRT: Floating Point Square Root

59

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SQRT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	264
Representation	264
Parameter Description	265

Short Description

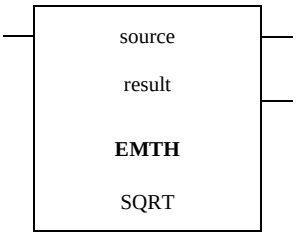
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 136*)".

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 standard square root operation
source (top node)	3x, 4x	DINT, UDINT	Source value (first of two contiguous registers)
result (middle node)	4x	DINT, UDINT	Result (first of two contiguous registers)
SQRT (bottom node)			Selection of the subfunction SQRT
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON =source value out of range

Parameter Description

Source Value (Top Node)

The first of two contiguous 3x or 4x registers is entered in the top node. The second register is implied. The source value, i.e. the value for which the square root will be derived, is stored here.
If you specify a **4x register**, the source value may be in the range 0 ... 99 999 99:

Register	Content
Displayed	The high-order half of the value is stored here.
First implied	The low-order half of the value is stored here.

If you specify a **3x register**, the source value may be in the range 0 ... 9 999:

Register	Content
Displayed	The square root calculation is done on only the value in the displayed register
First implied	This register is required but not used.

Result (Middle Node)

Enter the first of two contiguous 4x registers in the middle node. The second register is implied. The result of the standard square root operation is stored here in the fixed-decimal format: **`1234.5600.:`**.

Register	Content
Displayed	This register stores the four-digit value to the left of the first decimal point.
First implied	This register stores the four-digit value to the right of the first decimal point.

Note: Numbers after the second decimal point are truncated; no round-off calculations are performed.

EMTH-SQRTP: Process Square Root

60

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SQRTP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	268
Representation	268
Parameter Description	269
Example	270

Short Description

Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 136*)".

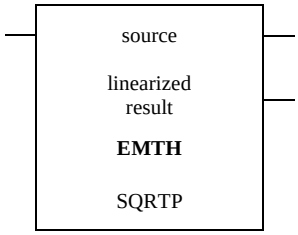
The process square root function tailors the standard square root function for closed loop analog control applications. It takes the result of the standard square root result, multiplies it by 63.9922 (the square root of 4 095) and stores that linearized result in the middle-node registers.

The process square root is often used to linearize signals from differential pressure flow transmitters so that they may be used as inputs in closed loop control operations.

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 process square root operation
source (top node)	3x, 4x	DINT, UDINT	Source value (first of two contiguous registers)
linearized result (middle node)	4x	DINT, UDINT	Linearized result (first of two contiguous registers)
SQRTP (bottom node)			Selection of the subfunction SQRPT
Top output	0x	None	ON = operation successful
Middle output	0x	None	ON =source value out of range

Parameter Description

Source Value (Top Node)

The first of two contiguous 3x or 4x registers is entered in the top node. The second register is implied. The source value, i.e. the value for which the square root will be derived, is stored here. In order to generate values that have meaning, the source value must not exceed 4 095.
If you specify a **4x register**:

Register	Content
Displayed	Not used
First implied	The source value will be stored here

If you specify a **3x register**:

Register	Content
Displayed	The source value will be stored here
First implied	Not used.

Linearized Result (Middle Node)

The first of two contiguous 4x registers is entered in the middle node. The second register is implied. The linearized result of the process square root operation is stored here n the fixed-decimal format **`1234.5600`**..

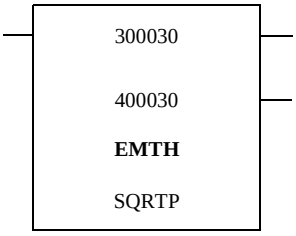
Register	Content
Displayed	This register stores the four-digit value to the left of the first decimal point.
First implied	This register stores the four-digit value to the right of the first decimal point.

Note: Numbers after the second decimal point are truncated; no round-off calculations are performed.

Example

Process Square Root Function

This example gives a quick overview of how the process square root is calculated.
Instruction



Suppose a source value of 2000 is stored in register 300030 of EMTH function SQRTP.
First, a standard square root operation is performed:

$\sqrt{2000} = 0044.72$

Then this result is multiplied by 63.9922, yielding a linearized result of 2861.63:

$0044.72 \times 63.9922 = 2861.63$

The linearized result is placed in the two registers in the middle node:

Register	Part of the result
400030	`2861` (four-digit value to the left of the first decimal point)
400031	`6300` (four-digit value to the right of the first decimal point)

EMTH-SUBDP: Double Precision Subtraction

61

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SUBDP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	272
Representation	272
Parameter Description	273

Short Description

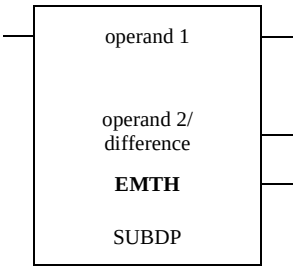
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 136*)".

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 = subtracts operand 2 from operand 1 and posts difference in designated registers
operand 1 (top node)	4x	DINT, UDINT	Operand 1 (first of two contiguous registers)
operand 2/ difference (middle node)	4x	DINT, UDINT	Operand 2 and difference (first of six contiguous registers)
SUBDP (bottom node)			Selection of the subfunction SUBDP
Top output	0x	None	ON = operand 1 > operand 2
Middle output	0x	None	ON = operand 1 = operand 2
Bottom output	0x	None	ON = operand 1 < operand 2

Parameter Description

Operand 1 (Top Node)

The first of two contiguous 4x registers is entered in the top node. The second 4x register is implied. Operand 1 is stored here.

Register	Content
Displayed	Register stores the low-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 1 Range 0 000 ... 9 999, for a combined double precision value in the range 0 ... 99 999 999

Operand 2 and Product (Middle Node)

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

Register	Content
Displayed	Register stores the low-order half of operand 2 for a combined double precision value in the range 0 ... 99 999 999
First implied	Register stores the high-order half of operand 2 for a combined double precision value in the range 0 ... 99 999 999
Second implied	This register stores the low-order half of the absolute difference in double precision format
Third implied	This register stores the high-order half of the absolute difference in double precision format
Fourth implied	0 = operands in range 1 = operands out of range
Fifth implied	This register is not used in the calculation but must exist in state RAM.

EMTH-SUBFI: Floating Point - Integer Subtraction

62

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SUBFI.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	276
Representation	276
Parameter Description	277

Short Description

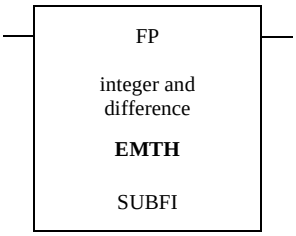
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP - integer operation
FP (top node)	4x	REAL	Floating point value (first of two contiguous registers)
integer and difference (middle node)	4x	DINT, UDINT	Integer value and difference (first of four contiguous registers)
SUBFI (bottom node)			Selection of the subfunction SUBFI
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value (Top Node)

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

Register	Content
Displayed First implied	The FP value from which the integer value is subtracted is stored here.

Sine of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers store the double precision integer value to be subtracted from the FP value.
Second implied Third implied	The difference is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-SUBFP: Floating Point Subtraction

63

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SUBFP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	280
Representation	280
Parameter Description	281

Short Description

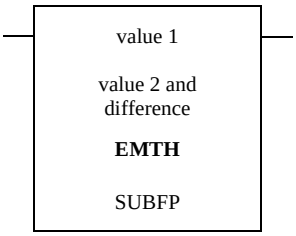
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 FP value 1 - value 2 subtraction
value 1 (top node)	4x	REAL	Floating point value 1 (first of two contiguous registers)
value 2 and difference (middle node)	4x	REAL	Floating point value 2 and the difference (first of four contiguous registers)
SUBFP (bottom node)			Selection of the subfunction SUBFP
Top output	0x	None	ON = operation successful

Parameter Description

Floating Point Value 1 (Top Node)

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

Register	Content
Displayed First implied	FP value 1 (the value from which value 2 will be subtracted) is stored here.

Floating Point Value 2 (Top Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	FP value 2 (the value to be subtracted from value 1) is stored in these registers
Second implied Third implied	The difference of the subtraction is stored here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-SUBIF: Integer - Floating Point Subtraction

64

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-SUBIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	284
Representation	284
Parameter Description	285

Short Description

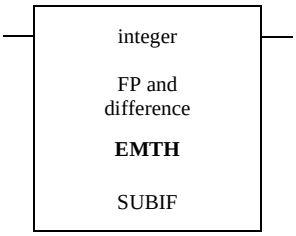
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 integer - FP operation
integer (top node)	4x	DINT, UDINT	Integer value (first of two contiguous registers)
FP and difference (middle node)	4x	REAL	FP value and difference (first of four contiguous registers)
SUBIF (bottom node)			Selection of the subfunction SUBIF
Top output	0x	None	ON = operation successful

Parameter Description

Integer Value (Top Node)

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

Register	Content
Displayed First implied	The double precision integer value from which the FP value is subtracted is stored here.

FP Value and Difference (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers store the FP value to be subtracted from the integer value.
Second implied Third implied	The difference is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

EMTH-TAN: Floating Point Tangent of an Angle (in Radians)

65

At a Glance

Introduction

This chapter describes the EMTH subfunction EMTH-TAN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	288
Representation	289
Parameter Description	290

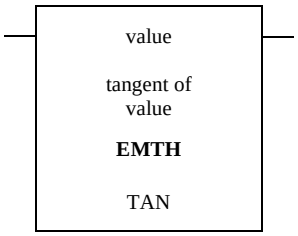
Short Description

Function Description	This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 137*)".

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 = calculates the tangent of the value
value (top node)	4x	REAL	FP value indicating the value of an angle in radians (first of two contiguous registers)
tangent of value (middle node)	4x	REAL	Tangent of the value in the top node (first of four contiguous registers)
TAN (bottom node)			Selection of the subfunction TAN
Top output	0x	None	ON = operation successful

Parameter Description

Value (Top Node)

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

Register	Content
Displayed First implied	An FP value indicating the value of an angle in radians is stored here. The magnitude of this value must be < 65 536.0.

If the magnitude is ≥ 65 536.0:

- The tangent is not computed
- An invalid result is returned
- An error is flagged in the EMTH-ERLOG (See *EMTH-ERLOG: Floating Point Error Report Log, p. 215*) function

Tangent of Value (Middle Node)

The first of four contiguous 4x registers is entered in the middle node. The remaining three registers are implied

Register	Content
Displayed First implied	Registers are not used but their allocation in state RAM is required.
Second implied Third implied	The tangent of the value in the top node is posted here in FP format (See *The IEEE Floating Point Standard, p. 138*).

Note: To preserve registers, you can make the 4x reference numbers assigned to the displayed register and the first implied register in the middle node equal to the register references in the top node, since the first two middle-node registers are not used.

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	292
Representation	293
Parameter Description	294
READ ASCII Message (Subfunction 1)	297
WRITE ASCII Message (Subfunction 2)	301
GET DATA (Subfunction 3)	302
PUT DATA (Subfunction 4)	303
ABORT (Middle Input ON)	307
Run Time Errors	308

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. 53*".

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.

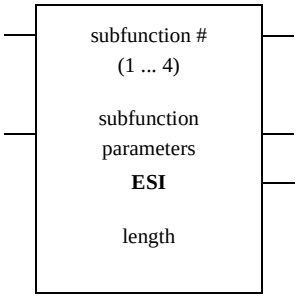
Further Information you will find in the *Quantum 140 ESI 062 10 ASCII Interface Module User Guide*

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

Top Input

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.

Middle Input

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.

Subfunction # (Top Node)

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 *READ ASCII Message, p. 297* followed by multiple GET DATA commands
2	Multiple PUT DATA commands followed by one command *WRITE ASCII Message, p. 301*
3	Zero or more commands *GET DATA, p. 302*
4	Zero or more commands *PUT DATA, p. 303*

Note: A fifth command, ABORT ASCII Message (See *ABORT, p. 307*), 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 ramaining 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
Note: 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. 308*)).

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.

Ouptuts

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 it is completed before the subfunction is executed. For more details see error checking (See *Run Time Errors, p. 308*).

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. 295*). 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	

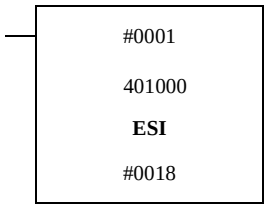
Response Structure

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:

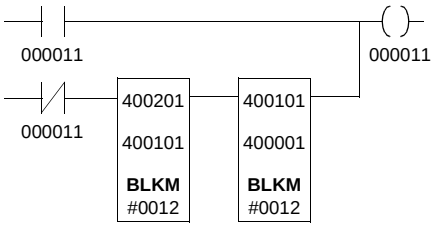
Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	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

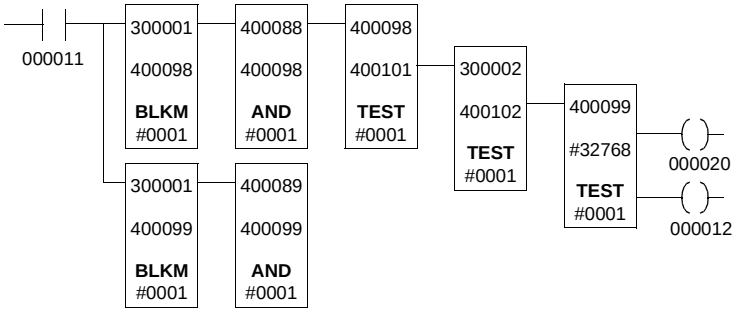


Contents of registers

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

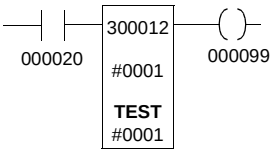
Register	Value (hex)	Description
400098	nnnn	Workspace for response word
400099	nnnn	Workspace for response word
400088	7FFF	Response word mask
400089	8000	Status word valid bit mask

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. 295*). 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

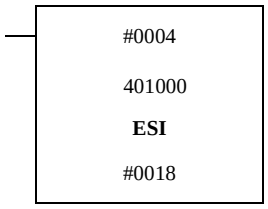
Response Structure

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:

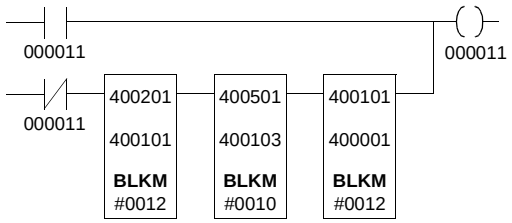
Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	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

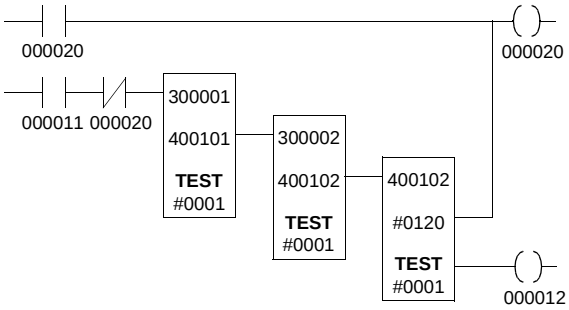


Contents of registers

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

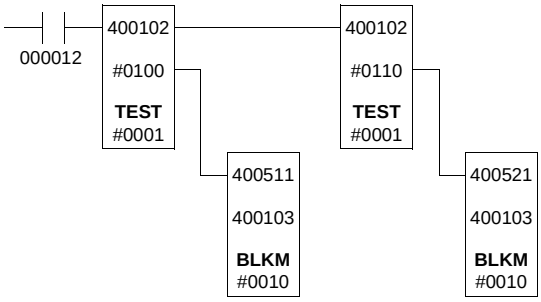
Second Network - Command Register Network



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.

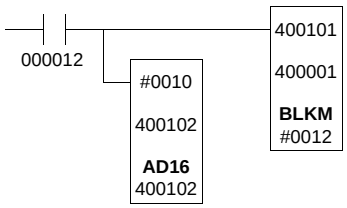
If both these tests show matches, the current module start register is tested against what would be the module start register of the last PUT DATA command for this transfer. If the test shows that the current module start register is greater than or equal to the last PUT DATA command, coil 000020 goes 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. 294*) 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 *Subfunction Parameters (Middle Node), p. 295*)
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	*Length (Bottom Node), p. 296* 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 instrcution EUCA.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	310
Representation	311
Parameter Description	312
Examples	313

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 "I*nstallation of DX Loadables, p. 53*".

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.

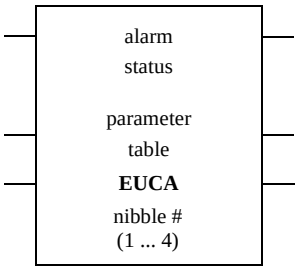
It also provides four levels of alarm checking on each of the four conversions:

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

Representation

Symbol

Representation of the instruction



Parameter Description

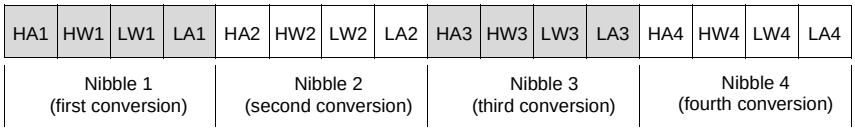
Description of the instruction's parameters

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 *Alarm Status (Top Node), p. 312*) (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:



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 ≤ 99 999
Third implied	Low engineering unit (LEU), minimum SPV required and set by the user (bottom end of the scale)	0 ≤ 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 ≤ DB < (HEU - LEU)
Fifth implied	HA alarm value in SPV units	HW < HA ≤ 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 ≤ LA < LW

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

Examples

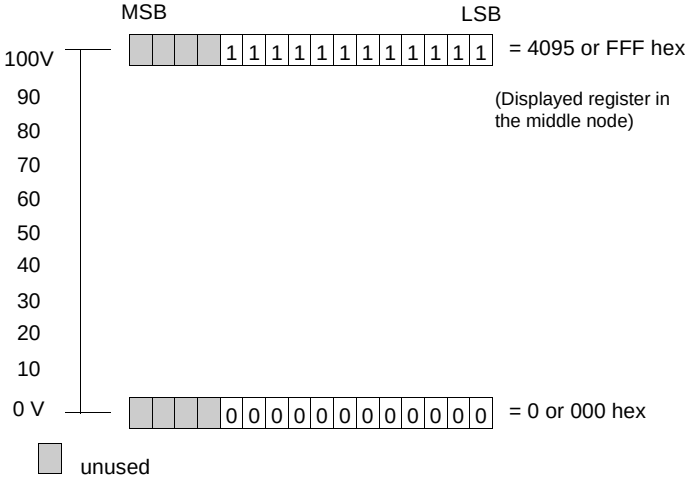
Overview

The following examples are shown:

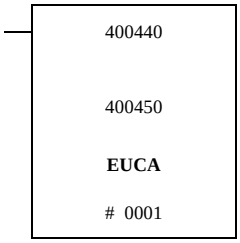
- *Example 1, p. 314*
  Principles of EUCA Operation
- *Example 2, p. 316*
  Use in a Drive System
- *Example 3, p. 318*
  Four EUCA conversions together

Example 1

This example demonstrates the principles of EUCA operation. The binary value is manually input in the displayed register in the middle node, and the result is visually available in the SPV register (the first implied register in the middle node).
The illustration below shows an input range equivalent of a 0 ... 100 V measure, corresponding to the whole binary 12-bit range:



A range of 0 ... 100 V establishes 50 V for nominal operation. EUCA provides a margin on the nominal side of both warning and alarm levels (deadband). If an alarm threshold is exceeded, the alarm bit becomes active and stays active until the signal becomes greater (or less) than the DB setting -5 V in this example.
Programming the EUCA block is accomplished by selecting the EUCA loadable and writing in the data as illustrated in the figure below:

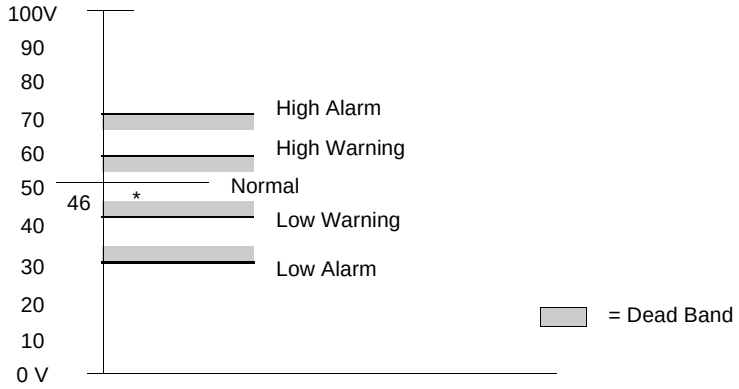


Reference Data

Register	Meaning	Content
400440	STATUS	0000000000000000
400450	INPUT	1871 DEC
400451	SPV	46 DEC

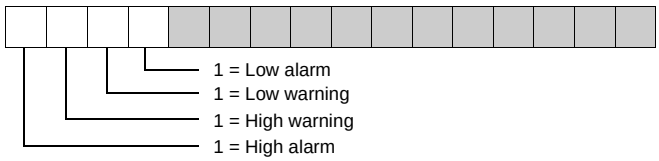
Register	Meaning	Content
400452	HIGH_unit	100 DEC
400453	LOW_unit	0 DEC
400454	Dead_band	5 DEC
400455	HIGH_ALARM	70 DEC
400456	HIGH_WARN	60 DEC
400457	LOW_ALARM	40 DEC
400458	LOW_WARN	30 DEC

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:

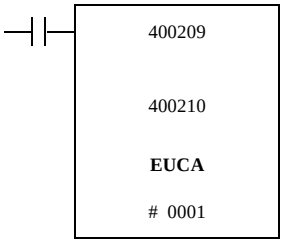


Example 2

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

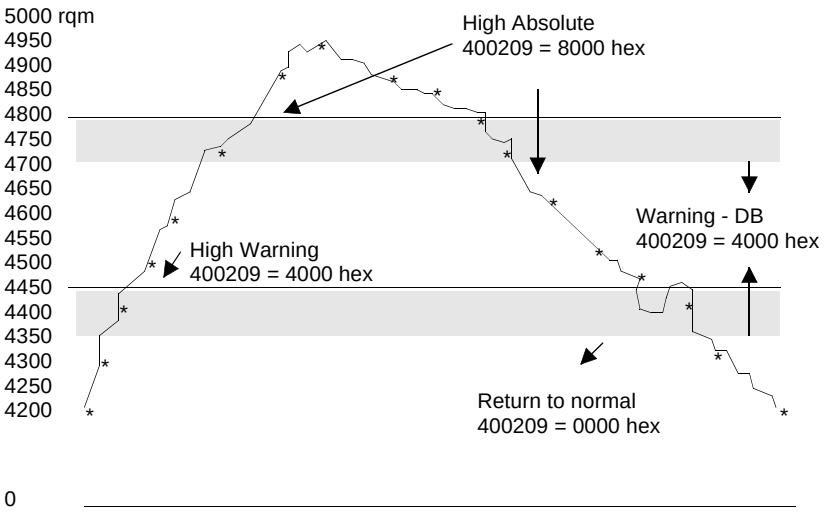
Instruction



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

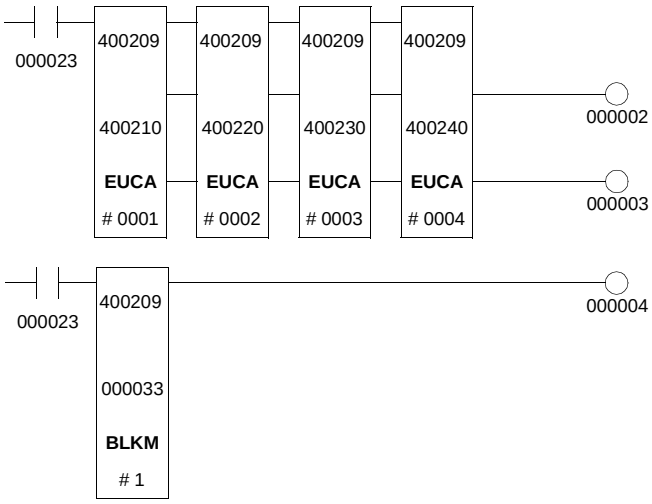
The N.O. contact is used to suppress alarm checks when the drive system is shutdown, or during initial start up allowing the system to get above the Low alarm RPM level.



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 it's 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 discretes 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

	Conversion 1	**Conversion 2**	**Conversion 3**	**Conversion 4**
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	322
Representation	322
Parameter Description	323

Short Description

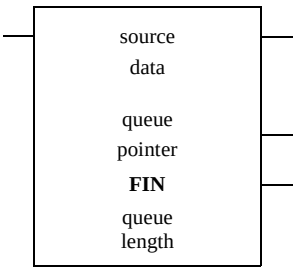
Function Description

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

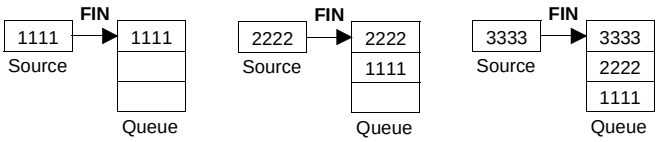
Description of the instruction's parameters

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 *Queue Pointer (Middle Node), p. 323*) (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 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.

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	326
Representation	327
Parameter Description	328

Short Description

Function Description

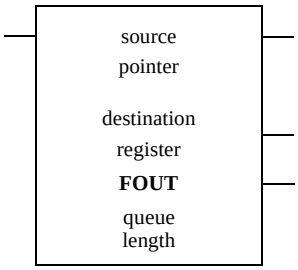
The FOUT instruction works together with the FIN instruction to produce a first in-first out (FIFO) queue. It moves the bit pattern of the holding register at the bottom of a full queue to a destination register or to word that stores 16 discrete outputs. An FOUT instruction has one control input and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

	DANGER
	Overriding any disabled coils 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. **Failure to observe this precaution will result in death or serious injury.**

Representation

Symbol

Representation of the instruction



Parameter Description

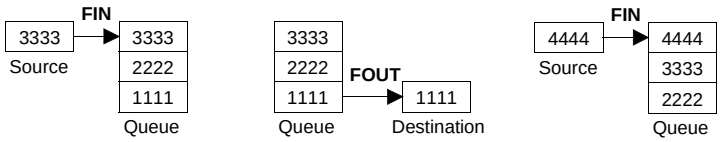
Description of the instruction's parameters

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. 321*) 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	330
Representation	330

Short Description

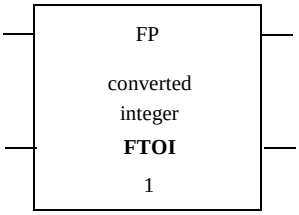
Function Description

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	332
Representation	333
Parameter Description	334
Parameter Description Top Node (History Matrix)	335
Parameter Description Middle Node (Status Matrix)	340
Parameter Description Bottom Node (Length)	344

Short Description

Function Description

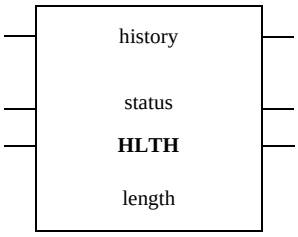
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information you will find in "I*nstallation of DX Loadables, p. 53*".

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	HLTH can be initialized to learn the configuration in which it is implemented and save the information as a point-in-time reference called *History Matrix (Top Node), p. 335* This matrix contains: ● A user-designated drop number for communications status monitoring ● User logic checksum ● Disabled I/O indicator ● S911 Health ● Choice of single or dual cable system ● I/O Map display
Monitor Mode	Monitor mode enables an operation that checks PLC system conditions. Detected changes are recorded in a *Status Matrix (Middle Node), p. 340*. The status matrix monitors the most recent system conditions and sets bit patterns to indicate detected changes. The status matrix contains: ● Communication status of the drop designated in the history matrix ● A flag to indicate when there is any disabled I/O ● Flags to indicate the "on/off" status of constant sweep and the Memory protect key switch ● Flags to indicate a battery-low condition and if Hot Standby is functional ● Failed module position data ● Changed user logic checksum flag ● RIO lost-communication flag

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 **Note:** 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:

length = (# of I/O drops x 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	346
Representation	346

Short Description

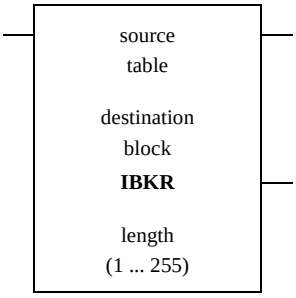
Function Description

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	348
Representation	348

Short Description

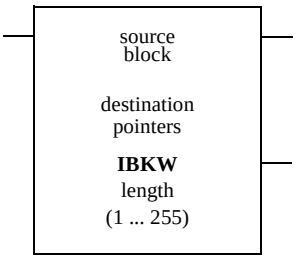
Function Description

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	350
Representation	351
Parameter Description	352
Cascaded DRUM/ICMP Blocks	353

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. 53*".

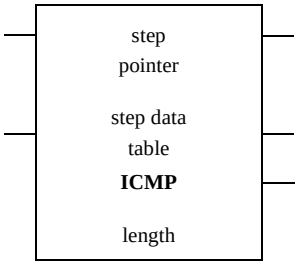
The ICMP (input compare) instruction provides logic for verifying the correct operation of each step processed by a DRUM instruction. Errors detected by ICMP may be used to trigger additional error-correction logic or to shut down the system.

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: O... 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 £ 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. The 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	356
Representation	356
Parameter Description	357

Short Description

Function Description

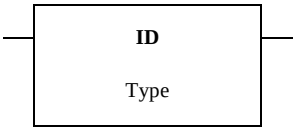
Note: This instruction is only available after configuring a CPU without extension.

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. 49*.

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

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	360
Representation	360
Parameter Description	361

Short Description

Function Description

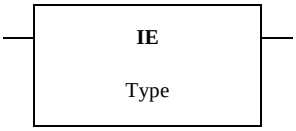
Note: This instruction is only available after configuring a CPU without extension.

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 unmasks 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. 49*.

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 unmasks 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 unmasks 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

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	364
Representation	364
Parameter Description	365
Run Time Error Handling	367

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.

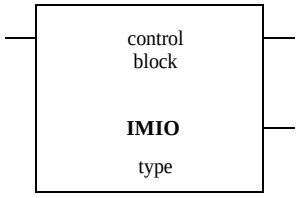
In order to use IMIO instructions, the local I/O modules to be accessed must be designated in the I/O Map in your panel software.

Further Information you will find in the chapter *Interrupt Handling, p. 49*.

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 *Runtime Errors, p. 367*) 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 *Physical Address of the I/O Module, p. 365* to be accessed.
First implied	This register logs the error status (See *Runtime Errors, p. 367*), 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

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	370
Representation	371
Parameter Description	372

Short Description

Function Description

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

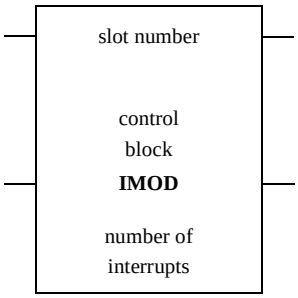
The IMOD instruction initiates a ladder logic interrupt handler subroutine when the appropriate interrupt is generated by a local interrupt module and received by the PLC. Each IMOD instruction in an application is set up to correspond to a specific slot in the local backplane where the interrupt module 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. 49*.

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. 51* 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 **CAUTION** *Lost of Interrupts, p. 374*)
9 - 15	Not used
16	0 = IMOD disabled 1 = IMOD enabled

Lost of Interrupts

CAUTION

Lost of interrupts from the working IMOD instruction

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.

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

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
Interrupt Point Status	
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
LAB number	
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	378
Representation	379
Parameter Description	380

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. 52* described in the chapter *General*.

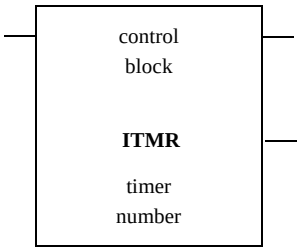
Each instance of the interval timer is delayed for a programmed interval while the PLC is running, then generates a processor interrupt when the interval has elapsed.

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. 49*.

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
Function Status	
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.
Function Control	
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	384
Representation	384

Short Description

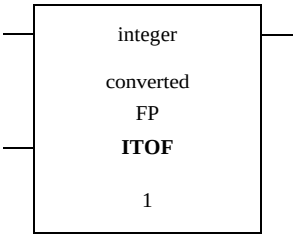
Function Description

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	386
Representation	386

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.

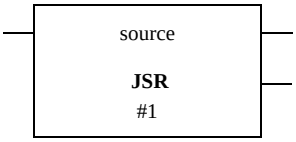
Example to Subroutine Handling

An example to subroutine handling you will find in the chapter *General*, section *Subroutine Handling, p. 51*.

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	388
Representation	388
Parameter Description	389

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.

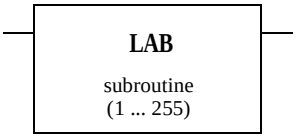
Example to Subroutine Handling

An example to subroutine handling you will find in the chapter *General*, section *Subroutine Handling, p. 51*.

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.

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	392
Representation	392
Parameter Description	393

Short Description

Function Description

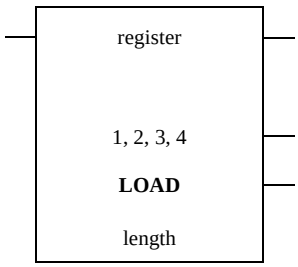
Note: This instruction is available with the PLC family TSX Compact, with Quantum CPUs 434 12/ 534 14 and Momentum CPUs CCC 960 x0/ 980 x0.

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

1, 2, 3, 4 (Middle Node)	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.
Bottom Output	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.

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	396
Representation	396
Parameter Description	397

Short Description

Function Description

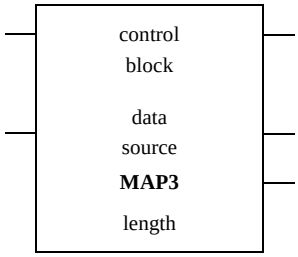
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 53*".

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	*Destination Device, p. 398*
2	*Qualifier / Function Code, p. 399*
3	*Network Mode / Network Type, p. 399*
4	*Function Status, p. 400*
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
Qualifier	
1 - 8	0 = addressed >0 = named
Function Code	
9 - 16	4 = read 5 = write

Network Mode / Network Type

Word 3 contains two bites 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
Mode	
5 - 8	1 = association
Type	
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.

Top Output	The top output passes power for one scan when a transaction completes successfully.
Middle Output	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
Bottom Output	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.

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	404
Representation	405
Parameter Description	406

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.

WARNING

Overriding of disabled coils without enabling them

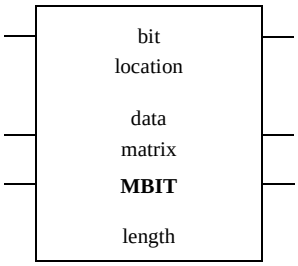
MBIT will override any disabled coils within a destination group without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the MBIT instruction.

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	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	408
Representation	409
Parameter Description	410
The MBUS Get Statistics Function	412

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. 53*".

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.

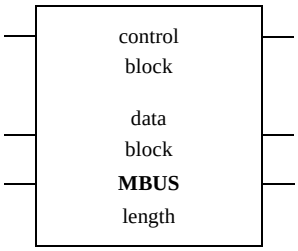
PLCs on a Modbus II network can handle up to 16 transactions simultaneously. Transactions include incoming (unsolicited) messages as well as outgoing messages. Thus, the number of message initiations a PLC can manage at any time is 16 - # of incoming messages.

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 discretes
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 discretes
Write coils	7.800 discretes

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 discretes using 24-bit CPUs (up to 16 discretes/word)
487	Writing discretes using 24-bit CPUs (up to 16 discretes/word

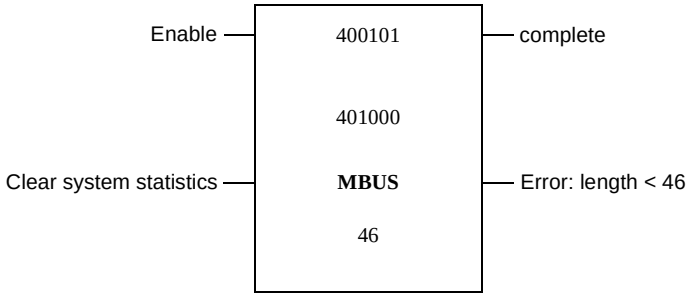
The MBUS Get Statistics Function

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. 413*
- *Software-maintained Receive Statistics, p. 413*
- *TBC-maintained Error Counters, p. 413*
- *Software-maintained Transmit Errors, p. 414*
- *Software-maintained Receive Errors, p. 414*
- *User Logic Transaction Errors, p. 414*
- *Manufacturing Message Format Standard, p. 414*
- *(MMFS) Errors, p. 414*
- *Background Statistics, p. 415*
- *Software Revision, p. 415*

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	418
Representation	419
Parameter Description	420

Short Description

Function Description

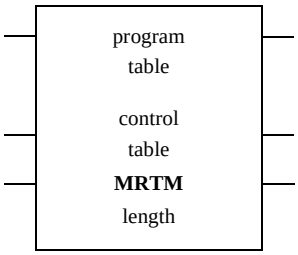
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 53*".

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 ≥ 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. **Note:** 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 form 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

Introduction This chapter describes the instruction MSTR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	425
Representation	426
Parameter Description	427
Write MSTR Operation	431
READ MSTR Operation	433
Get Local Statistics MSTR Operation	435
Clear Local Statistics MSTR Operation	437
Write Global Data MSTR Operation	439
Read Global Data MSTR Operation	440
Get Remote Statistics MSTR Operation	441
Clear Remote Statistics MSTR Operation	442
Peer Cop Health MSTR Operation	444
Reset Option Module MSTR Operation	446
Read CTE (Config Extension Table) MSTR Operation	447
Write CTE (Config Extension Table) MSTR Operation	449
Modbus Plus Network Statistics	452
TCP/IP Ethernet Statistics	457
Run Time Errors	458
Modbus Plus and SY/MAX Ethernet Error Codes	458
SY/MAX-specific Error Codes	460
TCP/IP Ethernet Error Codes	462
CTE Error Codes for SY/MAX and TCP/IP Ethernet	464

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.

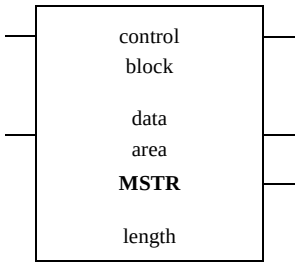
The MSTR instruction allows you to initiate one of 12 possible network communications operations over the network:

- 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	*Write MSTR Operation, p. 431*	x	x	x
2	*READ MSTR Operation, p. 433*	x	x	x
3	*Get Local Statistics MSTR Operation, p. 435*	x	x	-
4	*Clear Local Statistics MSTR Operation, p. 437*	x	x	-
5	*Write Global Data MSTR Operation, p. 439*	x	-	-
6	*Read Global Data MSTR Operation, p. 440*	x	-	-
7	*Get Remote Statistics MSTR Operation, p. 441*	x	x	-
8	*Clear Remote Statistics MSTR Operation, p. 442*	x	x	-
9	*Peer Cop Health MSTR Operation, p. 444*	x	-	-
10	*Reset Option Module MSTR Operation, p. 446*	-	x	x
11	*Read CTE (Config Extension Table) MSTR Operation, p. 447*	-	x	x
12	*Write CTE (Config Extension Table) MSTR Operation, p. 449*	-	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. 428*)
- TCP/IP Ethernet (See *Control Block for TCP/IP Ethernet, p. 429*)
- SY/MAX Ethernet (See *Control Block for SY/MAX EthernetEthernet, p. 430*)

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 *Run Time Errors , p. 458*)
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	0 0 0 0 0 1 1 1 High byte: indicating physical location (range 1 ... 16)
9 ... 16	0 x x x x x x x Destination address: binary value between 1 ... 64

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 *Run Time Errors , p. 458*)
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 *Run Time Errors , p. 458*)
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. 447*) and Write (See *Write CTE (Config Extension Table) MSTR Operation, p. 449*) 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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	Displays a hex value indicating an MSTR error: **Exception code + 3000**: Exception response, where response size is correct **4001**: Exception response, where response size is incorrect **4001**: 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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 EthernetEthernet

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status (See *Run Time Errors , p. 458*)	Displays a hex value indicating an MSTR error: **Exception code + 3000**: Exception response, where response size is correct **4001**: Exception response, where response size is incorrect **4001**: 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 *Run Time Errors , p. 458*)	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. 452*
- *TCP/IP Ethernet Statistics, p. 457*

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 *Run Time Errors , p. 458*)	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 *Modbus Plus Network Statistics, p. 452*); the length must be > 0 ≤ 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 *Run Time Errors , p. 458*)	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 *TCP/IP Ethernet Statistics, p. 457*); the length must be > 0 ≤ 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. 452*) 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. 452*
- *TCP/IP Ethernet Statistics, p. 457*

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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 ≤ 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 **Note:** 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 *Run Time Errors , p. 458*)	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 ≤ 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 **Note:** 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 *Run Time Errors , p. 458*)	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 ≤ 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 *Modbus Plus Network Statistics, p. 452*), 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 *Run Time Errors , p. 458*)	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 *TCP/IP Ethernet Statistics, p. 457*); the length must be > 0 ≤ 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. 452*) 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. 452*
- *TCP/IP Ethernet Statistics, p. 457*

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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 **Note**: 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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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. 449*) 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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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 *Run Time Errors , p. 458*)	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

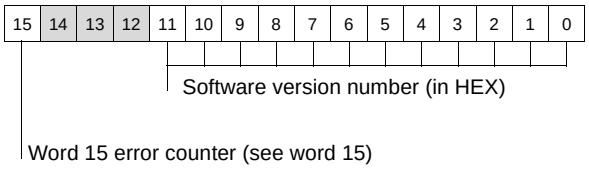
Modbus Plus Network Statistics

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: 
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		**If Word 1 bit 15 is not set**, Word 15 has the following meaning:
	LO	Receiver collision-abort error counter
	HI	Receiver alignment error counter
		If Word 1 bit 15 is set, 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. 458*
- *SY/MAX-specific Error Codes, p. 460*
- *TCP/IP Ethernet Error Codes, p. 462*
- *CTE Error Codes for SY/MAX and TCP/IP Ethernet, p. 464*

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 *ss HEX Value in Error Code 30ss, p. 459*)
4001	Inconsistent Modbus slave response

Hex Error Code	Meaning
5001	Inconsistent network response
6mss)	Routing failure (See *ss Hex Value in Error Code 6mss, p. 459*)

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: ● dest_drop, 0xFF ● 200, dest_drop, 0xFF ● 100+drop, dest_drop, 0xFF All other routing values generate an error
734B	One of the following errors has occurred: ● No CTE (configuration extension) table was configured ● No CTE table entry was created for the QSE Module slot number ● No valid drop was specified ● The QSE Module was not reset after the CTE was created **Note**: 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. ● When using an MSTR instruction, no valid slot or drop was specified

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 *ss Hex Value in Error Code 30ss, p. 462*)
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	NI_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	466
Representation	466

Short Description

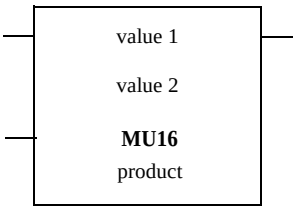
Function Description

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	468
Representation	468
Example	468

Short Description

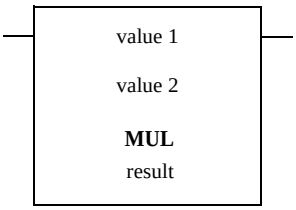
Function Description

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	470
Representation	470

Short Description

Function Description

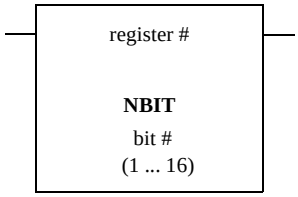
The normal bit (NBIT) instruction lets you control the state of a bit from a register by specifying its associated bit number in the bottom node. The bits being controlled are similar to coils, when a bit is turned ON, it stays ON until a control signal turns it OFF.

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	472
Representation	472

Short Description

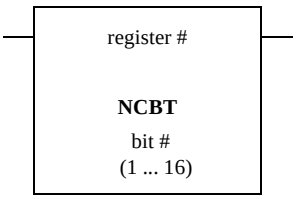
Function Description

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	474
Representation	474

Short Description

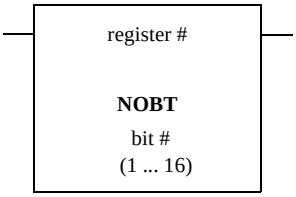
Function Description

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	476
Representation	478
Detailed Description	477

Short Description

Function Requirements

The following steps are necessary before using this instruction:

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

CAUTION

The outputs of the instruction turn on, regardless of the input states

When the NSUP loadable is not installed or is installed after the NOL loadable or is installed in a Quantum PLC with an executive < V2.0, all three outputs turn on, regardless of the input states.

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

Step	Action
2	Unpack and install the DX Loadable NOL; further information you will find in the chapter *Installation of DX Loadables, p. 53*.

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)
	10th implied	NOL version
	11th implied	Module firmware version
	12th implied	NOL DX version
	13th implied	Module DX version
	14th to 15th implied	Not used
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: ● Discrete inputs ● Register inputs ● Discrete outputs ● Register outputs 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*.

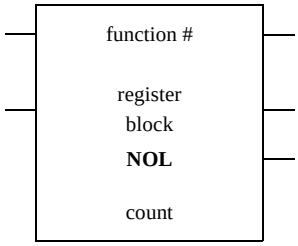
Count (Bottom Node)

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?

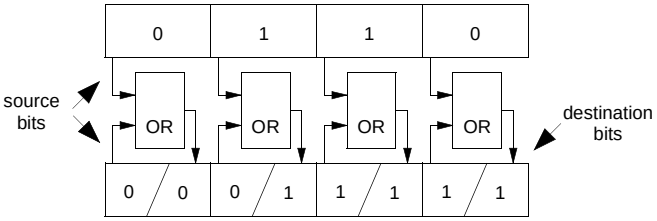
This chapter contains the following topics:

Topic	Page
Short Description	480
Representation	481
Parameter Description	481

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.

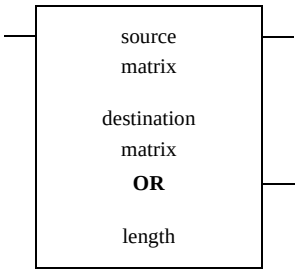


STOP	**WARNING**
	Overriding of any disabled coils within the destination matrix without enabling them OR will override any disabled coils within the destination matrix without enabling them. This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's 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.

PCFL: Process Control Function Library

96

At a Glance

Introduction

This chapter describes the instruction PCFL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	484
Representation	485
Parameter Description	486

Short Description

Function Description

The PCFL instruction gives you access to a library of process control functions utilizing analog values.

PCFL operations fall into three major categories:
- Advanced Calculations
- Signal Processing
- Regulatory Control

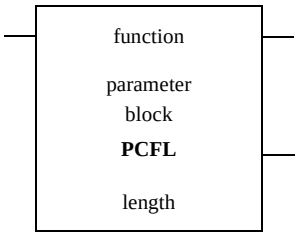
A PCFL function is selected from a list of alphabetical subfunctions in a pulldown menu in the panel software, and the subfunction is displayed in the top node of the instruction (see the table *Function (Top Node), p. 486* for a list of subfunctions and descriptions).

PCFL uses the same FP library as EMTH. If the PLC that you are using for PCFL does not have the onboard 80x87 math coprocessor chip, calculations take a comparatively long time to execute. PLCs with the math coprocessor can solve PCFL calculations ten times faster than PLCs without the chip. Speed, however, should not be an issue for most traditional process control applications where solution times are measured in seconds, not milliseconds.

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 specified process control function
function (top node)			Selection of process control function (subfunction)
parameter block (middle node)	4x	INT, UINT, WORD	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
length (bottom node)		INT, UINT	Length of parameter block (depending on selected subfunction
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Function (Top Node)

A subfunction for the selected PCFL library function is specified in the top node:

Operation	Subfunction	Description	Time-dependent Operations
Advanced Calculations	AVER	Average weighted inputs	no
	CALC	Calculate preset formula	no
	EQN	Formatted equation calculator	no
Signal Processing	ALARM	Central alarm handler for a PV input	no
	AIN	Convert inputs to scaled engineering units	no
	AOUT	Convert outputs to values in the 0 ... 4095 range	no
	DELAY	Time delay queue	yes
	LKUP	Look-up table	no
	INTEG	Integrate input at specified interval	yes
	LLAG	First-order lead/lag filter	yes
	LIMIT	Limiter for the PV (low/low, low, high, high/high)	no
	LIMV	Velocity limiter for changes in the PV (low, high)	yes
	MODE	Put input in auto or manual mode	no
	RAMP	Ramp to set point at a constant rate	yes
	RMPLN	Logarithmic ramp to set point (~2/3 closer to set point for each time constant)	yes
	RATE	Derivative rate calculation over a specified time	yes
	SEL	High/low/average input selection	no
Regulatory Control	KPID	Comprehensive ISA non-interacting proportional-integral-derivative (PID)	yes
	ONOFF	Specifies ON/OFF values for deadband	no
	PID	PID algorithms	yes
	PI	ISA non-interacting PI (with halt/manual/auto operation features)	yes
	RATIO	Four-station ratio controller	no
	TOTAL	Totalizer for metering flow	yes

Advanced Calculations

Advanced calculations are used for general mathematical purposes and are not limited to process control applications. With advanced calculations, you can create custom signal processing algorithms, derive states of the controlled process, derive statistical measures of the process, etc.

Simple math routines have already been offered in the EMTH instruction. The calculation capability included in PCFL is a textual equation calculator for writing custom equations instead of programming a series of math operations one by one.

Signal Processing

Signal processing functions are used to manipulate process and derived process signals. They can do this in a variety of ways; they linearize, filter, delay, and otherwise modify a signal. This category would include functions such as an Analog Input/Output, Limiters, Lead/Lag, and Ramp generators.

Regulatory Control

Regulatory functions perform closed loop control in a variety of applications. Typically, this is a PID (proportional integral derivative) negative feedback control loop. The PID functions in PCFL offer varying degrees of functionality. Function 75, PID, has the same general functionality as the PID2 instruction but uses floating point math and represents some options differently. PID is beneficial in cases where PID2 is not suitable because of numerical concerns such as round-off.

Further information you will find in the section Closed Loop Control (See *PCFL Subfunctions, p. 29*).

Parameter Block (Middle Node)

The 4x register entered in the middle node is the first in a block of contiguous holding register where the parameters for the specified PCFL operation are stored.

The ways that the various PCFL operations implement the parameter block are described in the description of the various subfunctions (PCFL operations).

Within the parameter block of each PCFL function are two registers used for input and output status.

Output Flags

In all PCFL functions, bits 12 ... 16 of the output status register define the following standard output flags:

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

Bit	Function
1 - 11	Not used
12	1 = Math error - invalid floating point or output
13	1 = Unknown PCFL function
14	not used
15	1 = Size of the allocated register table is too small
16	1 = Error has occurred - pass power to the bottom output

For time-dependent PCFL functions, bits 9 and 11 are also used as follows:

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

Bit	Function
1 - 8	Not used
9	1 = Initialization working
10	Not used
11	1 = Illegal solution interval
12	1 = Math error - invalid floating point or output
13	1 = Unknown PCFL function
14	not used
15	1 = Size of the allocated register table is too small
16	1 = Error has occurred - pass power to the bottom output

Input Flags

In all PCFL functions, bits 1 and 3 of the input status register define the following standard input flags:

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

Bit	Function
1	1 = Function initialization complete or in progress 0 = Initialize the function
2	not used
3	1 = Timer override
4 -16	not used

Length (Bottom Node)

The integer value entered in the bottom node specifies the length, i.e. the number of registers, of the PCFL parameter block. The maximum allowable length will vary depending on the function you specify.

PCFL-AIN: Analog Input

97

At a Glance

Introduction

This chapter describes the subfunction PCFL-AIN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	492
Representation	492
Parameter Description	493

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

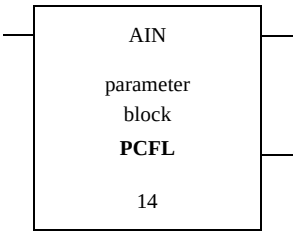
The AIN function scales the raw input produced by analog input modules to engineering values that can be used in the subsequent calculations.
Three scaling options are available:

- Auto input scaling
- Manual input scaling
- Implementing process square root on the input to linearize the signal before scaling

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 specified process control function
AIN (top node)			Selection of the subfunction AIN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction AIN (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

AIN supports the range resolutions for following device types:

Quantum Engineering Ranges

Resolution	Range: Valid	Range: Under	Range: Over
10 V	768 ... 64 768	767	64 769
V	16 768 ... 48 768	16 767	48 769
0 ... 10 V	0 ... 64 000	0	64 001
0 ... 5 V	0 ... 32 000	0	32 001
1 ... 5 V	6 400 ... 32 000	6 399	32 001

Quantum Thermocouple

Resolution	Range: Valid
TC degrees	-454 ... +3 308
TC 0.1 degrees	-4 540 ... +32 767
TC Raw Units	0 ... 65 535

Quantum Voltmeter

Resolution	Range: Valid	Range: Under	Range: Over
10 V	-10 000 ... +10 000	-10 001	+10 001
5 V	-5 000 ... +5 000	-5 001	+5 001
0 ... 10 V	0 ... 10 000	0	10 001
0 ... 5 V	0 ... 5 000	0	5 001
1 ... 5 V	1 000 ... 5 000	999	5 001

Parameter Block (Middle Node)

The length of the AIN parameter block is 14 registers:

Register	Content
Displayed	Input from a 3x register
First implied	Reserved
Second implied	*Output Status, p. 494*
Third implied	*Input Status, p. 495*
Fourth and fifth implied	Scale 100% engineering units
Sixth and seventh implied	Scale 0% engineering units
Eighth and ninth implied	Manual input
10th and 11th implied	Auto input
12th and 13th implied	Output

Output Status

Output Status

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

Bit	Function
1...5	Not used
6	1 = with TC PSQRT, invalid: in extrapolation range, PSQRT not used
7	1 = input out of range
8	1 = echo under range from input module
9	1 = echo over range from input module
10	1 = invalid output mode selected
11	1 = invalid Engineering Units
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 3	Standard input bits (flags) (See *Input Flags, p. 488*)
4 ... 8	Ranges (see following tables)
9	1 = process square root on raw input
10	1 = manual scaling mode 0 = auto scaling mode
11	1 = extrapolate over-/under-range for auto mode 0 = clamp over-/under-range for auto mode
12 ... 16	Not used

Quantum Engineering Ranges

Bit					
4	**5**	**6**	**7**	**8**	**Range**
0	1	0	0	0	+/- 10V
0	1	0	0	1	+/- 5V
0	1	0	1	0	0 ... 10 V
0	1	0	1	1	0 ... 5 V
0	1	1	0	0	1 ... 5 V

Quantum Thermocouple

Bit					
4	**5**	**6**	**7**	**8**	**Range**
0	1	1	0	1	TC degrees
0	1	1	1	0	TC 0.1 degrees
0	1	1	1	1	TC raw units

Quantum Voltmeter

Bit					
4	**5**	**6**	**7**	**8**	**Range**
1	0	0	0	0	+/- 10V
1	0	0	1	0	+/- 5V
1	0	1	0	0	0 ... 10 V
1	0	1	1	0	0 ... 5 V
1	1	0	0	0	1 ... 5 V

Note: Bit 4 in this register is nonstandard use.

PCFL-ALARM: Central Alarm Handler

98

At a Glance

Introduction

This chapter describes the subfunction PCFL-Alarm.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	498
Representation	498
Parameter Description	499

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

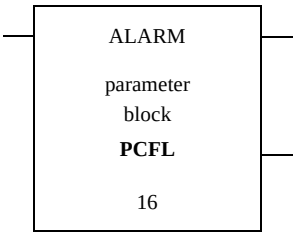
The ALARM function gives you a central block for alarm handling where you can set high (H), low (L), high high (HH), and low low (LL) limits on a process variable. ALARM lets you specify

- A choice of normal or deviation operating mode
- Whether to use H/L or both H/L and HH/LL limits
- Whether or not to use deadband (DB) around the limits

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 specified process control function
ALARM (top node)			Selection of the subfunction ALARM
parameter block (middle node)	4x	INT, UINT, WORD	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction ALARM (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

The following operating modes are available:

Mode	Meaning
Normal Operating Mode	ALARM operates directly on the input. Normal is the default condition
Deviation Operating Mode	ALARM operates on the change between the current input and the last input.
Deadband	When enabled, the DB option is incorporated into the HH/H/LL/L limits. These calculated limits are inclusive of the more extreme range, e.g. if the input has been in the high range, the output remains high and does not transition when the input hits the calculated H limit.
Operations	A flag is set when the input or deviation equals or crosses the corresponding limit. If the DB option is used, the HH, H, LL, L limits are adjusted internally for crossed-limit checking and hysteresis.

Note: ALARM automatically tracks the last input, even when you specify normal mode, to facilitate a smooth transition to deviation mode.

Parameter Block (Middle Node)

The length of the ALARM parameter block is 16 registers:

Register	Content
Displayed and first implied	Input registers
Second implied	*Output Status, p. 500*
Third implied	*Input Status, p. 500*
Fourth and fifth implied	HH limit value
Sixth and seventh implied	H limit value
Eighth and ninth implied	L limit value
10th and 11th implied	LL limit value
12th and 13th implied	Deadband (DB) around limit
14th and 15th implied	Last input

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = DB set to negative number
6	1 = deviation mode chosen with DB option
7	1 = LL crossed ($x \leq LL$
8	1 = L crossed ($x \leq L$ or $LL < x \leq L$) with HH/LL option set
9	1 = H crossed ($x \geq H$ or $H \leq x < HH$) with HH/LL option set
10	1 = HH crossed ($x \geq HH$)
11	1 = invalid limits specified
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = deviation mode 0 = normal mode
6	1 = both H/L and HH/LL limits apply
7	1 = DB enabled
8	1 = retain H/L flag when HH/LL limits crossed
9 ... 16	Not used

PCFL-AOUT: Analog Output

99

At a Glance

Introduction

This chapter describes the subfunction PCFL-AOUT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	502
Representation	503
Parameter Description	504

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The AOUT function is an interface for calculated signals for output modules. It converts the signal to a value in the range 0 ... 4 096.

Formula

Formula of the AOUT function:

$$OUT = \frac{scale \times (IN - LEU)}{(HEU - LEU)}$$

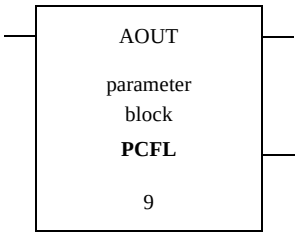
The meaning of the elements:

Element	Meaning
HEU	High Engineering Unit
IN	Input
LEU	Low Engineering Unit
OUT	Output
scale	Scale

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 specified process control function
AOUT (top node)			Selection of the subfunction AOUT
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
9 (bottom node)		INT, UINT	Length of parameter block for subfunction AOUT (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the AOUT parameter block is 9 registers:

Register	Content
Displayed and first implied	Input in engineering units
Second implied	*Output Status, p. 504*
Third implied	*Input Status, p. 504*
Fourth and fifth implied	High engineering units
Sixth and seventh implied	Low engineering units
Eighth and ninth implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 7	Not used
8	1 = clamped low
9	1 = clamped high
10	not used
11	1 = invalid H/L limits
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-AVER: Average Weighted Inputs Calculate

100

At a Glance

Introduction

This chapter describes the subfunction PCFL-AVER.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	506
Representation	507
Parameter Description	508

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 487*.

The AVER function calculates the average of up to four weighted inputs.

Formula

Formula of the AVER function:

$$RES = \frac{(k + (w_1 \times In_1) + (w_2 \times In_2) + (w_3 \times In_3) + (w_4 \times In_4))}{1 + w_1 + w_2 + w_3 + w_4}$$

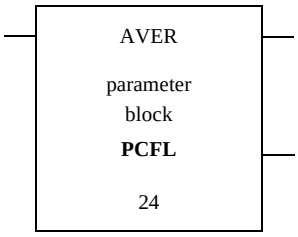
The meaning of the elements:

Element	Meaning
In_1 ... In_4	Inputs
k	Constant
RES	Result
w_1 ... w_4	Weights

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 specified process control function
AVER (top node)			Selection of the subfunction AVER
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
24 (bottom node)		INT, UINT	Length of parameter block for subfunction AVER (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the AVER parameter block is 24 registers:

Register	Content
Displayed and first implied	reserved
Second implied	*Output Status, p. 508*
Third implied	*Input Status, p. 509*
Fourth and fifth implied	Value of In1
Sixth and seventh implied	Value of Inv2
Eighth and ninth implied	Value of In3
10th and 11th implied	Value of In4
12th and 13th implied	Value of k
14th and 15th implied	Value of wv1
16th and 17th implied	Value of wv2
18th and 19th implied	Value of wv3
20th and 21st implied	Value of wv4
22nd and 23rd implied	Value of result

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = no inputs activated
11	1 = result negative 0 = result positive
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = In4 and w4 are used
6	1 = In3 and w3 are used
7	1 = In2 and w2 are used
8	1 = In1 and w1 are used
9	1 = k is active
10 ... 16	Not used

A weight can be used only when its corresponding input is enabled, e.g. the 20th and 21st implied registers (which contain the value of w4) can be used only when the 10th and 11th implied registers (which contain In4) are enabled. The I in the denominator is used only when the constant is enabled.

PCFL-CALC: Calculated preset formula

101

At a Glance

Introduction

This chapter describes the subfunction PCFL-CALC.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	512
Representation	512
Parameter Description	513

Short Description

Function Description

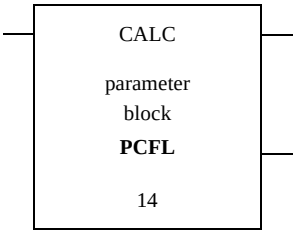
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 487*.

The CALC function calculates a preset formula with up to four inputs, each characterized in a separate register of the parameter block.

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 specified process control function
CALC (top node)			Selection of the subfunction CALC
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction CALC (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the CALC parameter block is 14 registers:

Register	Content
Displayed and first implied	Reserved
Second implied	*Output Status, p. 513*
Third implied	*Input Status, p. 514*
Fourth and fifth implied	Value of input A
Sixth and seventh implied	Value of input B
Eighth and ninth implied	Value of input C
10th and 11th implied	Value of input D
12th and 13th implied	Value of the output

Output Status

Output Status

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

Bit	Function
1...10	Not used
11	1 = bad input code chosen
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 6	not used
7 ... 10	Formula Code
11 ... 16	Not used

Formula Code

Bit				Formula Code
7	**8**	**9**	**10**	
0	0	0	1	$(A \times B) - (C \times D)$
0	0	1	1	$(A \times B)/(C \times D)$
0	1	0	0	$A/(B \times C \times D)$
0	1	0	1	$(A \times B \times C)/D$
0	1	1	0	$A \times B \times C \times D$
0	1	1	1	$A + B + C + D$
1	0	0	0	$A \times B(C-D)$
1	0	0	1	$A[(B/C)^{D}]$
1	0	1	0	$A \times LN(B/C)$
1	0	1	1	$(A-B) - (C - D)/LN[(A - B)/(C - D)]$
1	1	0	0	$(A/B)^{(-C/D)}$
1	1	0	1	$(A-B)/(C - D)$

PCFL-DELAY: Time Delay Queue

102

At a Glance

Introduction

This chapter describes the subfunction PCFL-DELAY.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	516
Representation	517
Parameter Description	518

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The DELAY function can be used to build a series of readings for time-delay compensation in the logic. Up to 10 sampling instances can be used to delay an input.

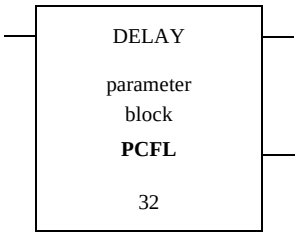
All values are carried along in registers, where register x[0] contains the current sampled input. The 10th delay period does not need to be stored. When the 10th instance in the sequence takes place, the value in register x[9] can be moved directly to the output

A DXDONE message is returned when the calculation is complete. The function can be reset by toggling the first-scan bit.

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 specified process control function
DELAY (top node)			Selection of the subfunction DELAY
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
32 (bottom node)		INT, UINT	Length of parameter block for subfunction DELAY (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the DELAY parameter block is 32 registers:

Register	Content
Displayed and first implied	Input at time n
Second implied	*Output Status, p. 518*
Third implied	*Input Status, p. 519*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	x[0] delay
12th and 13th implied	x[1] delay
14th and 15th implied	x[2] delay
...	...
28th and 29th implied	x[9] delay
30th and 31st implied	Output registers

Output Status

Output Status

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

Bit	Function
1...3	Not used
4	1 = k out of range
5 ... 8	Count of registers left to be initialized
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 8	Time Delay ≤ 10
9 ... 11	Echo number of registers left to be initialized
12 ... 16	Not used

PCFL-EQN: Formatted Equation Calculator

103

At a Glance

Introduction

This chapter describes the subfunction PCFL-EQN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	522
Representation	522
Parameter Description	523

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 487*.

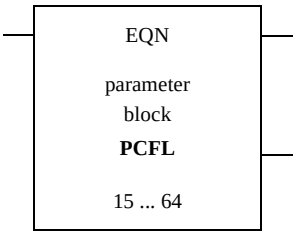
The EQN function is a formatted equation calculator. You must define the equation in the parameter block with various codes that specify operators, input selection and inputs.

EQN is used for equations that have four or fewer variables but do not fit into the CALC format. It complements the CALC function by letting you input an equation with floating point and integer inputs as well as operators.

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 specified process control function
EQN (top node)			Selection of the subfunction EQN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
15 ... 64 (bottom node)		INT, UINT	Length of parameter block for subfunction EQN
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

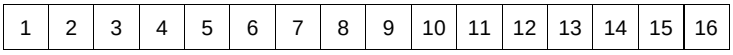
Parameter Block (Middle Node)

The length of the EQN parameter block can be as high as 64 registers:

Register	Content
Displayed and first implied	Reserved
Second implied	*Output Status, p. 523*
Third implied	*Input Status, p. 524*
Fourth and fifth implied	Variable A
Sixth and seventh implied	Variable B
Eighth and ninth implied	Variable C
10th and 11th implied	Variable D
12th and 13th implied	Output
14th implied	First *Formula Code, p. 524*
15th implied	Second possible formula code
...	...
63rd implied	Last possible formula code

Output Status

Output Status



Bit	Function
1	Stack error
2...3	Not used
4 ... 8	Code of last error logged
9	1 = bad operator selection code
10	1 = EQN not fully programmed
11	1 = bad input code chosen
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = Degree/radian option for trigonometry
6 ... 8	not used
9 ... 16	Equation size for display in Concept

Formula Code

Each formula code in the EQN function defines either an input selection code or an operator selection code.

Formula Code (Parameter Block)

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

Bit	Function
1 ... 4	Not used
5 ... 8	Definition of input selection
9 ... 11	Not used
12 ... 16	Definition of operator selection

Input Selection

Bit				Input Selection
5	**6**	**7**	**8**	
0	0	0	0	Use operator selection
0	0	0	1	Float input
0	0	1	1	16-bit integer
1	0	0	0	Variable A
1	0	0	1	Variable B
1	0	1	0	Variable C
1	0	1	1	Variable D

Operator Selection

Bit					Operator Selection
12	**13**	**14**	**15**	**16**	
0	0	0	0	0	No operation
0	0	0	0	1	Absolute value
0	0	0	1	0	Addition
0	0	0	1	1	Division
0	0	1	0	0	Exponent
0	0	1	1	1	LN (natural logarithm)
0	1	0	0	0	G (logarithm)
0	1	0	0	1	Multiplication
0	1	0	1	0	Negation
0	1	0	1	1	Power
0	1	1	0	0	Square root
0	1	1	0	1	Subtraction
0	1	1	1	0	Sine
0	1	1	1	1	Cosine
1	0	0	0	0	Tangent
1	0	0	0	1	Arcsine
1	0	0	1	0	Arccosine
1	0	0	1	1	Arctangent

PCFL-INTEG: Integrate Input at Specified Interval

104

At a Glance

Introduction

This chapter describes the subfunction PCFL-INTEG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	528
Representation	529
Parameter Description	530

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The INTEG function is used to integrate over a specified time interval. No protection against integral wind-up is provided in this function. INTEG is time-dependent, e.g. if you are integrating at an input value of 1/sec, it matters whether it operates over one second (in which case the result is 1) or over one minute (in which case the result is 60).

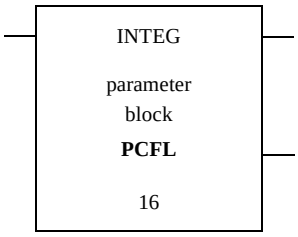
You can set flags to either initialize or restart the function after an undetermined down-time, and you can reset the integral sum if you wish. If you set the initialize flag, you must specify a reset value (zero or the last output in case of power failure), and calculations will be skipped for one sample.

The function returns a DXDONE message when the operation is complete.

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 specified process control function
INTEG (top node)			Selection of the subfunction INTEG
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction INTEG (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the INTEG parameter block is 16 registers:

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 530*
Third implied	*Input Status, p. 530*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Reset value
14th and 15th implied	Result

Output Status

Output Status

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

Bit	Function
1...8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	Reset sum
6 ... 16	Not used

PCFL-KPID: Comprehensive ISA Non Interacting PID

105

At a Glance

Introduction

This chapter describes the subfunction PCFL-KPID.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	532
Representation	533
Parameter Description	534

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

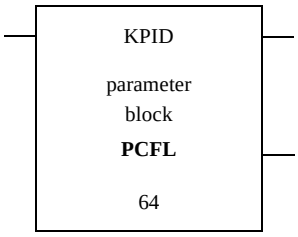
The KPID function offers a superset of the functionality of the PID function, with additional features that include:

- A gain reduction zone
- A separate register for bumpless transfer when the integral term is not used
- A reset mode
- An external set point for cascade control
- Built-in velocity limiters for set point changes and changes to a manual output
- A variable derivative filter constant
- Optional expansion of anti-reset wind-up limits

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 specified process control function
KPID (top node)			Selection of the subfunction KPID
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
64 (bottom node)		INT, UINT	Length of parameter block for subfunction KPID (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the KPID parameter block is 64 registers:

	Register	Content
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, Register 1, p. 535*
	Third implied	*Output Status, Register 2, p. 536*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 536*
Input Parameters	Sixth and seventh implied	Proportional rate, KP
	Eighth and ninth implied	Reset time, TI
	10th and 11th implied	Derivative action time, TD
	12th and 13th implied	Delay time constant, TD1
	14th and 15th implied	Gain reduction zone, GRZ
	16th and 17th implied	Gain reduction in GRZ, KGRZ
	18th and 19th implied	Limit rise of manual set point value
	20th and 21st implied	Limit rise of manual output
	22nd and 23rd implied	High limit for Y
	24th and 25th implied	Low limit for Y
	26th and 27th implied	Expansion for anti-reset wind-up limits
Inputs	28th and 29th implied	External set point for cascade
	30th and 31st implied	Manual set point
	32nd and 33rd implied	Manual Y
	34th and 35th implied	Reset for Y
	36th and 37th implied	Bias

	Register	Content
Outputs	38th and 39th implied	Bumpless transfer register, BT
	40th and 41st implied	Calculated control difference (error term), XD
	42nd implied	Previous operating mode
	43rd and 44th implied	Dt (in ms) since last solve
	45th and 46th implied	Previous system deviation, XD_1
	47th and 48th implied	Previous input, X_1
	49th and 50th implied	Integral part for Y, YI
	51st and 52nd implied	Differential part for Y, YD
	53rd and 54th implied	Set point, SP
	55th and 56th implied	Proportional part for Y, YP
	57th implied	Previous operating status
Timing Information	58th implied	10 ms clock at time n
	59th implied	Reserved
	60th and 61th implied	Solution interval (in ms)
Output	62th and 63th implied	Manipulated output variable, Y

Output Status, Register 1

Output Status Register 1

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4	1 = Cascade mode selected
5	1 = Auto mode selected
6	1 = Halt mode selected
7	1 = Manual mode selected
8	1 = Reset mode selected
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Output Status, Register 2

Output Status Register 2

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

Bit	Function
1...4	Not used
5	1 = Previous D mode selected
6	1 = Previous I mode selected
7	1 = Previous P mode selected
8	1 = Previous mode selected
9 ... 16	Not used

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = Reset mode
6	1 = Manual mode
7	1 = Halt mode
8	1 = Cascade mode
9	1 = Solve proportional algorithm
10	1 = Solve integral algorithm
11	1 = Solve derivative algorithm
12	1 = solve derivative algorithm based on x 0 = solve derivative algorithm based on xd
13	1 = anti--reset wind-up on YI only 0 = normal anti--reset wind-up
14	1 = disable bumpless transfer 0 = bumpless transfer
15	1 = Manual Y tracks Y
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-LIMIT: Limiter for the Pv

106

At a Glance

Introduction

This chapter describes the subfunction PCFL-LIMIT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	538
Representation	538
Parameter Description	539

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

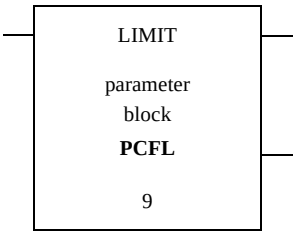
The LIMIT function limits the input to a range between a specified high and low value. If the high or low limit is reached, the function sets an H or L flag and clamps the output.

LIMIT returns a DXDONE message when the operation is complete.

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 specified process control function
LIMIT (top node)			Selection of the subfunction LIMIT
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
9 (bottom node)		INT, UINT	Length of parameter block for subfunction LIMIT (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LIMIT parameter block is 9 registers:

Register	Content
Displayed and first implied	Current input
Second implied	*Output Status, p. 539*
Third implied	*Input Status, p. 539*
Fourth and fifth implied	Low limit
Sixth and seventh implied	High Limit
Eighth implied	Output register

Output Status

Output Status

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

Bit	Function
1...8	Not used
9	1 = input < low limit
10	1 = input > high limit
11	1 = invalid high/low limits (e.g., low ≥ high
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-LIMV: Velocity Limiter for Changes in the Pv

107

At a Glance

Introduction

This chapter describes the subfunction PCFL-LIMV.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	542
Representation	542
Parameter Description	543

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

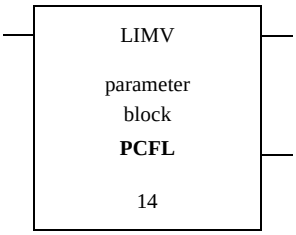
The LIMV function limits the velocity of change in the input variable between a specified high and low value. If the high or low limit is reached, the function sets an H or L flag and clamps the output.

LIMV returns a DXDONE message when the operation is complete.

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 specified process control function
LIMV (top node)			Selection of the subfunction LIMV
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction LIMV (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LIMV parameter block is 14 registers:

Register	Content
Displayed and first implied	Input register
Second implied	*Output Status, p. 543*
Third implied	*Input Status, p. 543*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Velocity limit / sec
12th and 13th implied	Result

Output Status

Output Status

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

Bit	Function
1...5	Not used
6	1 = negative velocity limit
7	1 = input < low limit
8	1 = input > high limit
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-LKUP: Look-up Table

108

At a Glance

Introduction

This chapter describes the subfunction PCFL-LKUP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	546
Representation	546
Parameter Description	547

Short Description

Function Description

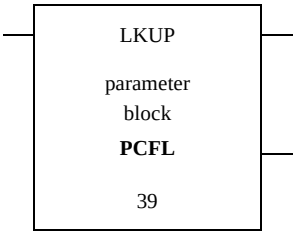
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The LKUP function establishes a look-up table using a linear algorithm to interpolate between points. LKUP can handle variable point intervals and variable numbers of points.

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 specified process control function
LKUP (top node)			Selection of the subfunction LKUP
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
39 (bottom node)		INT, UINT	Length of parameter block for subfunction LKUP (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

The LKUP function establishes a look-up table using a linear algorithm to interpolate between points. LKUP can handle variable point intervals and variable numbers of points.

If the input (x) is outside the specified range of points, the output (y) is clamped to the corresponding output y0 or yn. If the specified parameter block length is too small or if the number of points is out of range, the function does not check the xn because the information from that pointer is invalid.

Points to be interpolated are determined by a binary search algorithm starting near the center of x data. The search is valid for x1 < x < xn. The variable x may occur multiple times with the same value, the value chosen from the look-up table is the first instance found.
For example, if the table is:

x	y
10.0	1.0
20.0	2.0
30.0	3.0
30.0	3.5
40.0	4.0

then an input of 30.0 finds the first instance of 30.0 and assigns 3.0 as the output. An input of 31.0 would assign the value 3.55 as the output.
No sorting is done on the contents of the lookup table. Independent variable table values should be entered in ascending order to prevent unreachable gaps in the table.

The function returns a DXDONE message when the operation is complete.

Parameter Block (Middle Node)

The length of the LKUP parameter block is 39 registers:

Register	Content
Displayed and first implied	Input
Second implied	*Output Status, p. 548*
Third implied	*Input Status, p. 548*
Fourth implied	Number of point pairs
Fifth and sixth implied	Point x1
Seventh and eighth implied	Point y1
Ninth and tenth implied	Point x2
11th and 12th implied	Point y2
. . .	. . .
33rd and 34th implied	Point x8
35th and 36th implied	Point y8
37th and 38th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = input clamped, i.e. out of table's range
11	! = invalid number of points
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-LLAG: First-order Lead/Lag Filter

109

At a Glance

Introduction

This chapter describes the subfunction PCFL-LLAG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	550
Representation	551
Parameter Description	552

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The LLAG function provides dynamic compensation for a known disturbance. It usually appears in a feed-forward algorithm or as a dynamic filter. LLAG passes the input through a filter comprising a lead term (a numerator) and a lag term (a denominator) in the frequency domain, then multiplies it by a gain. Lead, lag, gain, and solution interval must be user-specified.

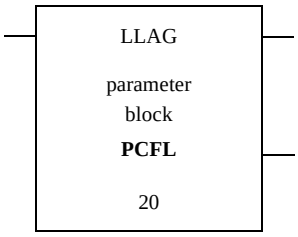
For best results, use lead and lag terms that are $\geq 4 * \Delta t$. This will ensure sufficient granularity in the output response.

LLAG returns a DXDONE message when the operation completes

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 specified process control function
LLAG (top node)			Selection of the subfunction LLAG
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
20 (bottom node)		INT, UINT	Length of parameter block for subfunction LLAG (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LLAG parameter block is 20 registers

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 552*
Third implied	*Input Status, p. 552*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Lead term
14th and 15th implied	Lag term
16th and 17th implied	Filter gain
18th and 19th implied	Result

Output Status

Output Status

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

Bit	Function
1...8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-MODE: Put Input in Auto or Manual Mode

110

At a Glance

Introduction

This chapter describes the subfunction PCFL-MODE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	554
Representation	555
Parameter Description	556

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The MODE function sets up a manual or automatic station for enabling and disabling data transfers to the next block. The function acts like a BLKM instruction, moving a value to the output register.

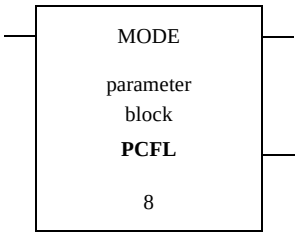
In auto mode, the input is copied to the output. In manual mode, the output is overwritten by a user entry.

MODE returns a DXDONE message when the operation completes.

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 specified process control function
MODE (top node)			Selection of the subfunction MODE
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
8 (bottom node)		INT, UINT	Length of parameter block for subfunction MODE (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the MODE parameter block is 8 registers:

Register	Content
Displayed and first implied	Input
Second implied	*Output Status, p. 556*
Third implied	*Input Status, p. 556*
Fourth and fifth implied	Manual input
Sixth and seventh implied	Output register

Output Status

Output Status

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

Bit	Function
1 ... 10	Not used
11	Echo mode: 1 = manual mode 0 = auto mode
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = manual mode 0 = auto mode
6 ... 16	Not used

PCFL-ONOFF: ON/OFF Values for Deadband

111

At a Glance

Introduction

This chapter describes the subfunction PCFL-ONOFF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	558
Representation	559
Parameter Description	560

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

The ONOFF function is used to control the output signal between fully ON and fully OFF conditions so that a user can manually force the output ON or OFF.
You can control the output via either a direct or reverse configuration:

Configuration	IF Input...	Then Output...
Direct	< (SP - DB)	ON
	> (SP + DB)	OFF
Revers	> (SP + DB)	ON
	< (SP - DB)	OFF

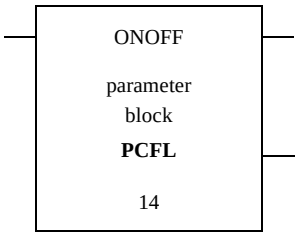
Manual Override

Two bits in the input status register (the third implied register in the parameter block) are used for manual override. When bit 6 is set to 1, manual mode is enforced. In manual mode, a 0 in bit 7 forces the output OFF, and a 1 in bit 7 forces the output ON. The state of bit 7 has meaning only in manual mode.

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 specified process control function
ONOFF (top node)			Selection of the subfunction ONOFF
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction ONOFF (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the ONOFF parameter block is 14 registers:

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 560*
Third implied	*Input Status, p. 561*
Fourth and fifth implied	Set point, SP
Sixth and seventh implied	Deadband (DB) around SP
Eighth and ninth implied	Fully ON (maximum output)
10th and 11th implied	Fully OFF (minimum output)
12th and 13th implied	Output, ON or OFF

Output Status

Output Status

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

Bit	Function
1 ... 8	Not used
9	1 = DB set to negative number
10	Echo mode: 1 = manual override 0 = auto mode
11	1 = output set to ON 0 = output set to OFF
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = reverse configuration 0 = direct configuration
6	1 = manual override 0 = auto mode
7	1 = force output ON in manual mode 0 = force output OFF in manual mode
8 ... 16	Not used

PCFL-PI: ISA Non Interacting PI

112

At a Glance

Introduction

This chapter describes the subfunction PCFL-PI.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	564
Representation	564
Parameter Description	565

Short Description

Function Description

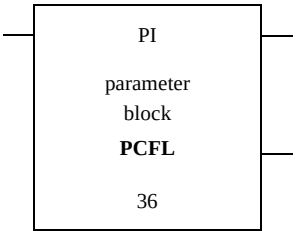
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

The PI function performs a simple proportional-integral operations using floating point math. It features halt / manual / auto operation modes. It is similar to the PID and KPID functions but does not contain as many options. It is available for higher-speed loops or inner loops in cascade strategies.

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 specified process control function
PI (top node)			Selection of the subfunction PI
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
36 (bottom node)		INT, UINT	Length of parameter block for subfunction PI (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the PI parameter block is 36 registers:

	Register	Content
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, p. 566*
	Third implied	*Error Word, p. 566*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 566*
Inputs	Sixth and seventh implied	Set point, SP
	Eighth and ninth implied	Manual output
	10th and 11th implied	Calculated control difference (error), XD
Outputs	12th implied	Previous operating mode
	13th and 14th implied	Dt (in ms) since last solve
	15th and 16th implied	Previous system deviation, XD_1
	17th and 18th implied	Integral part of output Y
	19th and 20th implied	Previous input, X_1
	21st implied	Previous operating status
Timing Information	22nd implied	10 ms clock at time n
	23rd implied	Reserved
	24th and 25th implied	Solution interval (in ms)
Input Parameters	26th and 27th implied	Proportional rate, KP
	28th and 29th implied	Reset time, TI
	30th and 31st implied	High limit on output Y
	32nd and 33rd implied	Low limit on output Y
Output	34th and 35th implied	Manipulated variable output, Y

Output Status

Output Status

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Error Word

Error Word

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

Bit	Function
1...11	Not used
12 ... 16	Error Description

Error Description

Bit					Meaning
12	**13**	**14**	**15**	**16**	
1	0	1	1	0	Negative integral time constant
1	0	1	0	1	High/low limit error (low ≥ high)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	Not used
6	1 = Manual mode
7	1 = Halt mode
8 ... 15	Not used
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-PID: PID Algorithms

113

At a Glance

Introduction

This chapter describes the subfunction PCFL-PID.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	568
Representation	568
Parameter Description	569

Short Description

Function Description

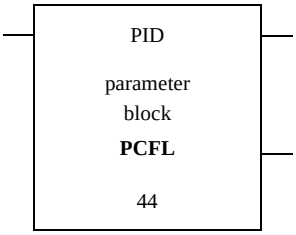
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

The PID function performs ISA non-interacting proportional-integral-derivative (PID) operations using floating point math. Because it uses FP math (unlike PID2), round-off errors are negligible.
In the part "General Information" you will find *A PID Example, p. 33*.

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 specified process control function
PID (top node)			Selection of the subfunction PID
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
44 (bottom node)		INT, UINT	Length of parameter block for subfunction PID (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the KPID parameter block is 44 registers:

	Register	Content
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, p. 570*
	Third implied	*Error Word, p. 570*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 571*
Inputs	Sixth and seventh implied	Set point, SP
	Eighth and ninth implied	Manual output
	10th and 11th implied	Summing junction, Bias
Outputs	12th and 13th implied	Error, XD
	14th implied	Previous operating mode
	15th and 16th implied	Elapsed time (in ms) since last solve
	17th and 18th implied	Previous system deviation, XD_1
	19th and 20th implied	Previous input, X_1
	21st and 22nd implied	Integral part of output Y, YI
	23rd and 24th implied	Differential part of output Y, YD
	25th and 26th implied	Proportional part of output Y, YP
	27th implied	Previous operating status
Timing Information	28th implied	Current time
	29th implied	Reserved
Inputs	30th and 31st implied	Solution interval (in ms)
	34th and 35th implied	Reset time, TI
	36th and 37th implied	Derivative action time, TD
	38th and 39th implied	High limit on output Y
	40th and 41st implied	Low limit on output Y
	42nd and 43rd implied	Manipulated control output, Y

Output Status

Output Status

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Error Word

Error Word

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

Bit	Function
1...11	Not used
12 ... 16	Error Description

Error Description

Bit					Meaning
12	**13**	**14**	**15**	**16**	
1	0	1	1	1	Negative derivative time constant
1	0	1	1	0	Negative integral time constant
1	0	1	0	1	High/low limit error (low ≥ high)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	Not used
6	1 = Manual mode
7	1 = Halt mode
8	Not used
9	1 = Solve proportional algorithm
10	1 = Solve integral algorithm
11	1 = Solve derivative algorithm
12	1 = solve derivative algorithm based on x 0 = solve derivative algorithm based on xd
13... 15	Not used
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-RAMP: Ramp to Set Point at a Constant Rate

114

At a Glance

Introduction

This chapter describes the subfunction PCFL-RAMP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	574
Representation	575
Parameter Description	576

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The RAMP function allows you to ramp up linearly to a target set point at a specified approach rate.
You need to specify:

- The target set point, in the same units as the contents of the input register are specified
- The sampling rate
- A positive rate toward the target set point, negative rates are illegal

The direction of the ramp depends on the relationship between the target set point and the input, i.e. if x < SP, the ramp is up; if x > SP, the ramp is down.

You may use a flag to initialize after an undetermined down-time. The function will store a new sample, then wait for one cycle to collect the second sample. Calculations will be skipped for one cycle and the output will be left as is, after which the ramp will resume.

RAMP terminates when the entire ramping operation is complete (over multiple scans) and returns a DXDONE message.

Starting the Ramp

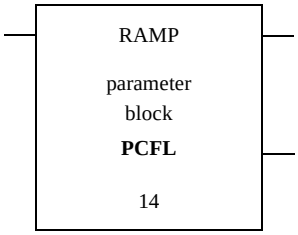
The following steps need to be done when starting the ramp (up/down) and each and every time you need to start or restart the ramp.

Step	Action
1	Set bit 1 of the standard input bits (See *Input Flags, p. 488*) to "1" (third implied register of the parameter block).
2	Retoggle the top input (enable input) to the instruction. Ramp will now start to ramp up/down from the initial value previously configured up/down to the previously configured setpoint. Monitor the 12th implied register of the parameter block for floating point value of the ramp value in progress.

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 specified process control function
RAMP (top node)			Selection of the subfunction RAMP
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction RAMP (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RAMP parameter block is 14 registers:

Register	Content
Displayed and first implied	Set point (Input)
Second implied	*Output Status, p. 576*
Third implied	*Input Status, p. 576*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Rate of change (per second) toward set point
12th and 13th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = ramp rate is negative
6	1 = ramp complete 0 = ramp in progress
7	1 = ramping down
8	1 = ramping up
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

Top Output (Operation Succesfull)

The top output of the PCFL subfunction RAMP goes active at each successive discrete ramp step up/down. It happens so fast that it appears to be solidly on. This top output should **NOT** be used as "Ramp done bit".

Bit 6 of the output status (second impied register of the parameter block) should be monitored as "Ramp done bit".

PCFL-RATE: Derivative Rate Calculation over a Specified Time

115

At a Glance

Introduction

This chapter describes the subfunction PCFL-RATE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	580
Representation	581
Parameter Description	582

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The RATE function calculates the rate of change over the last two input values. If you set an initialization flag, the function records a sample and sets the appropriate flags.

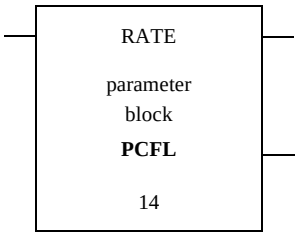
If a divide-by-zero operation is attempted, the function returns a DXERROR message.

It returns a DXDONE message when the operation completes successfully.

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 specified process control function
RATE (top node)			Selection of the subfunction RATE
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction RATE (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RATE parameter block is 14 registers:

Register	Content
Displayed and first implied	Current input
Second implied	*Output Status, p. 582*
Third implied	*Input Status, p. 582*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Result

Output Status

Output Status

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

Bit	Function
1 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-RATIO: Four Station Ratio Controller

116

At a Glance

Introduction

This chapter describes the subfunction PCFL-RATIO.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	584
Representation	585
Parameter Description	586

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

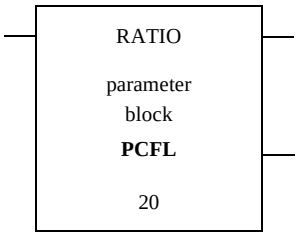
The RATIO function provides a four-station ratio controller. Ratio control can be used in applications where one or more raw ingredients are dependent on a primary ingredient. The primary ingredient is measured, and the measurement is converted to engineering units via an AIN function. The converted value is used to set the target for the other ratioed inputs.

Outputs from the ratio controller can provide set points for other controllers. They can also be used in an open loop structure for applications where feedback is not required.

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 specified process control function
RATIO (top node)			Selection of the subfunction RATIO
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
20 (bottom node)		INT, UINT	Length of parameter block for subfunction RATIO (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RATIO parameter block is 20 registers:

Register	Content
Displayed and first implied	Live input
Second implied	*Output Status, p. 586*
Third implied	*Input Status, p. 587*
Fourth and fifth implied	Ratio for input 1
Sixth and seventh implied	Ratio for input 2
Eighth and ninth implied	Ratio for input 3
10th and 11th implied	Ratio for input 4
12th and 13th implied	Output for input 1
14th and 15th implied	Output for input 2
16th and 17th implied	Output for input 3
18th and 19th implied	Output for input 4

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = parameter(s) out of range
11	1 = no inputs activated
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1= input 4 active
6	1= input 3 active
7	1= input 2 active
8	1= input 1 active
9 ... 16	Not used

PCFL-RMPLN: Logarithmic Ramp to Set Point

117

At a Glance

Introduction

This chapter describes the subfunction PCFL-RMPLN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	590
Representation	591
Parameter Description	592

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

The RMPLN function allows you to ramp up logarithmically to a target set point at a specified approach rate. At each successive call, it calculates the output until it is within a specified deadband (DB). DB is necessary because the incremental distance the ramp crosses decreases with each solve.

You need to specify:
- The target set point, in the same units as the contents of the input register are specified
- The sampling rate
- The time constant used for the logarithmic ramp, which is the time it takes to reach 63.2% of the new set point

For best results, use a t that is $\geq 4 * \Delta t$. This will ensure sufficient granularity in the output response.

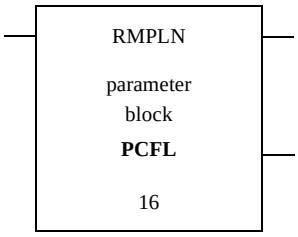
You may use a flag to initialize after an undetermined down-time. The function will store a new sample, then wait for one cycle to collect the second sample. Calculations will be skipped for one cycle and the output will be left as is, after which the ramp will resume.

RMPLN terminates when the input reaches the target set point + the specified DB and returns a DXDONE message.

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 specified process control function
RMPLN (top node)			Selection of the subfunction RMPLN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction RMPLN (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RMPLN parameter block is 16 registers:

Register	Content
Displayed and first implied	Set point (Input)
Second implied	*Output Status, p. 592*
Third implied	*Input Status, p. 592*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Time constant, τ, (per second) of exponential ramp toward the target set point
12th and 13th implied	DB (in engineering units)
14th and 15th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = DB or τ set to negative units
6	1 = ramp complete 0 = ramp in progress
7	1 = ramping down
8	1 = ramping up
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5 ... 16	Not used

PCFL-SEL: Input Selection

118

At a Glance

Introduction

This chapter describes the subfunction PCFL-SEL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	594
Representation	594
Parameter Description	595

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 487*.

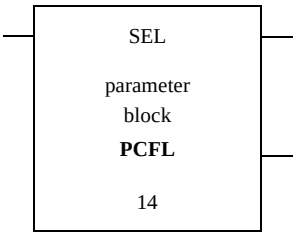
The SEL function compares up to four inputs and makes a selection based upon either the highest, lowest, or average value. You choose the inputs to be compared and the comparison criterion. The output is a copy of the selected input.

SEL returns a DXDONE message when the operation is complete.

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 specified process control function
SEL (top node)			Selection of the subfunction SEL
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction SEL (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the SEL parameter block is 14 registers:

Register	Content
Displayed and first implied	Reserved
Second implied	*Output Status, p. 595*
Third implied	*Input Status, p. 596*
Fourth and fifth implied	Input 1
Sixth and seventh implied	Input 2
Eighth and ninth implied	Input 3
10th and 11th implied	Input 4
12th and 13th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	Invalid selection modes
11	No inputs selected
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = enable input 1 0 = disable input 1
6	1 = enable input 2 0 = dyeable input 2
7	1 = enable input 3 0 = dyeable input 3
8	1 = enable input 4 0 = dyeable input 4
9 ... 10	Selection mode
11 ... 16	Not used

Selection mode

Bit		Meaning
9	**10**	
0	0	Select average
0	1	Select high
1	0	Select low
1	1	reserved / invalid

PCFL-TOTAL: Totalizer for Metering Flow

119

At a Glance

Introduction

This chapter describes the subfunction PCFL-TOTAL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	598
Representation	599
Parameter Description	600

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 487*.

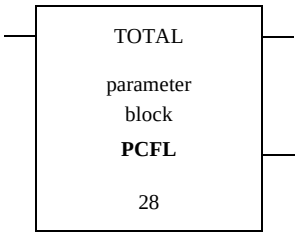
The TOTAL function provides a material totalizer for batch processing reagents. The input signal contains the units of weight or volume per unit of time. The totalizer integrates the input over time.

The algorithm reports three outputs:
- The integration sum
- The remainder left to meter in
- The valve output (in engineering units).

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 specified process control function
TOTAL (top node)			Selection of the subfunction TOTAL
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
28 (bottom node)		INT, UINT	Length of parameter block for subfunction TOTAL (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

The function uses up to three different set points:

- A trickle flow set point
- A target set point
- An auxiliary trickle flow set point

The target set point is for the full amount to be metered in. Here the output will be turned OFF.

The trickle flow set point is the cut-off point when the output should be decreased from full flow to a percentage of full flow so that the target set point is reached with better granularity.
The auxiliary trickle flow set point is optional. It is used to gain another level of granularity. If this set point is enabled, the output is reduced further to 10% of the trickle output.

The totalizer works from zero as a base point. The set point must be a positive value

In normal operation, the valve output is set to 100% flow when the integrated value is below the trickle flow set point. When the sum crosses the trickle flow set point, the valve flow becomes a programmable percentage of full flow. When the sum reaches the desired target set point, the valve output is set to 0% flow.

Set points can be relative or absolute. With a relative set point, the deviation between the last summation and the set point is used. Otherwise, the summation is used in absolute comparison to the set point.

There is a halt option to stop the system from integrating.

When the operation has finished, the output summation is retained for future use. You have the option of clearing this sum. In some applications, it is important to save the sum, e.g. if the meters or load cells cannot handle the full batch in one charge and measurements are split up, if there are several tanks to fill for a batch and you want to keep track of batch and production sums.

Parameter Block (Middle Node)

The length of the TOTAL parameter block is 28 registers:

Register	Content
Displayed and first implied	Live input
Second implied	*Output Status, p. 601*
Third implied	*Input Status, p. 602*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input, X_1
12th and 13th implied	Reset value
14th and 15th implied	Set point, target
16th and 17th implied	Set point, trickle flow
18th and 19th implied	% of full flow for trickle flow set point
20th and 21st implied	Full flow
22nd and 23rd implied	Remaining amount to SP
24th and 25th implied	Resulting sum
26th and 27th implied	Output for final control element

Output Status

Output Status

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

Bit	Function
1 ... 2	Not used
3 ... 4	0 0 = OFF 0 1 = trickle flow 1 0 = full flow
5	1 = operation done
6	1 = totalizer running
7	1 = overshoot past set point by more than 5%
8	1 = parameter(s) out of range
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 488*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 488*)
5	1 = reset sum
6	1 = halt integration
7	1 = deviation set point 0 = absolute set point
8	1 = use auxiliary trickle flow set point
9 ... 16	Not used

PEER: PEER Transaction

120

At a Glance

Introduction

This chapter describes the instruction PEER.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	604
Representation	605
Parameter Description	606

Short Description

Function Description

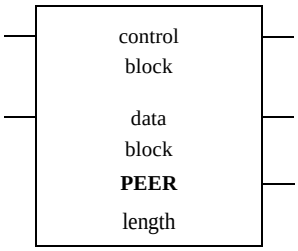
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 53*".

The S975 Modbus II Interface option modules use two loadable function blocks: MBUS and PEER. The PEER instruction can initiate identical message transactions with as many as 16 devices on Modbus II at one time. In a PEER transaction, you may only write register data.

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
control block (top node)	4x	INT, UINT, WORD	First of 19 contiguous registers in the PEER control block
data block (middle node)	4x	INT, UINT	First register in a data block to be transmitted by the PEER function
length (bottom node)		INT, UINT	Length, i.e. the number of holding registers, of the data block; range: 1 ... 249.
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 19 contiguous registers in the PEER control block:

Register	Function
Displayed	Indicates the status of the transactions at each device, the leftmost bit being the status of device #1 and the rightmost bit the status of device #16: 0 = OK, 1 = transaction error
First implied	Defines the reference to the first 4x register to be written to in the receiving device; a 0 in this field is an invalid value and will produce an error (the bottom output will go ON)
Second 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
Third implied	The Modbus port 3 address of the first of the receiving devices; address range: 1 ... 255 (0 = no transaction requested)
Fourth implied	The Modbus port 3 address of the second of the receiving devices; address range: 1 ... 255 (0 = no transaction requested)
. . .	. . .
18th implied	The Modbus port 3 address of the 16th of the receiving devices (address range: 1 ... 255)

PID2: Proportional Integral Derivative

121

At a Glance

Introduction

This chapter describes the instruction PID2.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	608
Representation	609
Detailed Description	610
Parameter Description	613
Run Time Errors	618

Short Description

Function Description

The PID2 instruction implements an algorithm that performs proportional-integral-derivative operations. The algorithm tunes the closed loop operation in a manner similar to traditional pneumatic and analog electronic loop controllers. It uses a rate gain limiting (RGL) filter on the PV as it is used for the derivative term only, thereby filtering out higher-frequency PV noise sources (random and process generated).

Formula

Proportional Control

$$M_V = K_1 E + bias$$

Proportional-Integral Control

$$M_V = K_1 \left(E + K_2 \int_0^t E \Delta t \right)$$

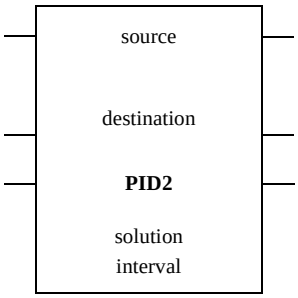
Proportional-Integral-Derivative Control

$$M_V = K_1 \left(E + K_2 \int_0^t E \Delta t + K_3 \frac{\Delta PV}{\Delta t} \right)$$

Representation

Symbol

Representation of the instruction



Parameter Description

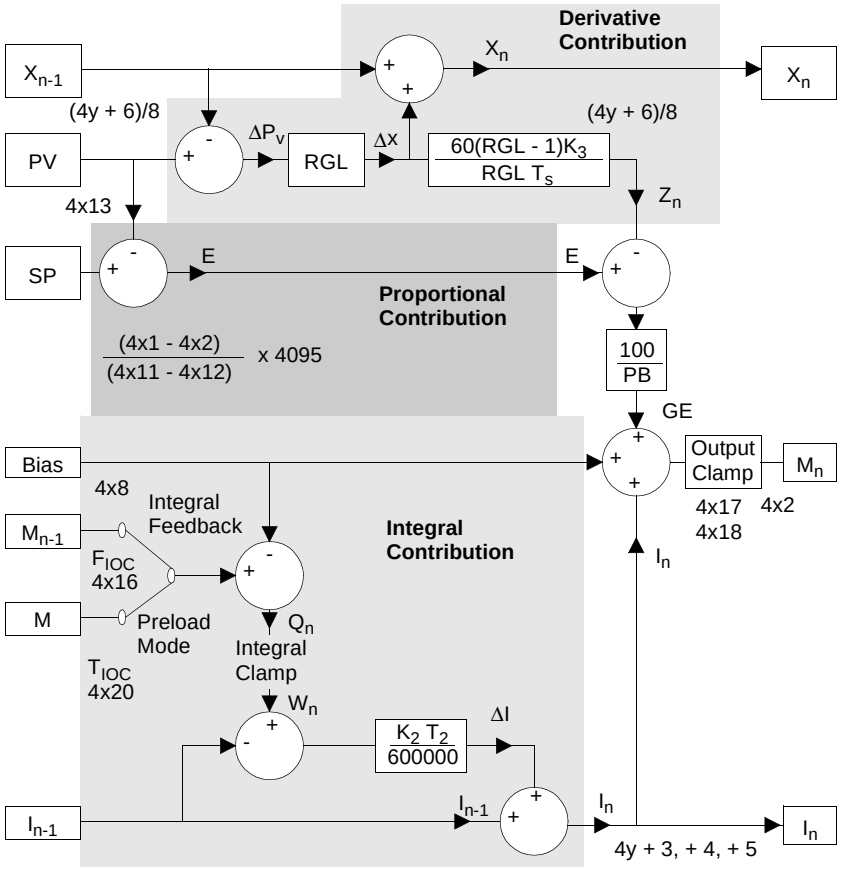
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	0 = Manual mode 1 = Auto mode
Middle input	0x, 1x	None	0 = Integral preload OFF 1 = Integral preload ON
Bottom input	0x, 1x	None	0 = Output increases as E increases 1 = Output decreases as E decreases
source (top node)	4x	INT, UINT	First of 21 contiguous holding registers in a source block
destination (middle node)	4x	INT, UINT	First of nine contiguous holding registers used for PID2 calculation. Do not load anything in these registers!
solution interval (bottom node)		INT, UINT	Contains a number ranging from 1 ... 255, indicating how often the function should be performed.
Top output	0x	None	1 = Invalid user parameter or Loop ACTIVE but not being solved
Middle output	0x	None	1 = PV ≥ high alarm limit
Bottom output	0x	None	1 = PV ≤ low alarm limit

Detailed Description

Block Diagram

Block Diagram



The elements in the block diagram have the following meaning:

Element	Meaning
E	Error, expressed in raw analog units
SP	Set point, in the range 0 ... 4095
PV	Process variable, in the range 0 ... 4095
x	Filtered PV
K2	Integral mode gain constant, expressed in 0.01 min^{-1}

Element	Meaning
K3	Derivative mode gain constant, expressed in hundredths of a minute
RGL	Rate gain limiting filter constant, in the range 2 ... 30
Ts	Solution time, expressed in hundredths of a second
PB	Proportional band, in the range 5 ... 500%
bias	Loop output bias factor, in the range 0 ... 4095
M	Loop output
GE	Gross error, the proportional-derivative contribution to the loop output
Z	Derivative mode contribution to GE
Qn	Unbiased loop output
F	Feedback value, in the range 0 ... 4095
I	Integral mode contribution to the loop output
I_{low}	Anti-reset-windup low SP, in the range 0 ... 4095
I_{high}	Anti-reset-windup high SP, in the range 0 ... 4095
K1	100/PB

Note: The integral mode contribution calculation actually integrates the difference of the output and the integral sum, this is effectively the same as integrating the error.

Proportional Control

With proportional-only control (P), you can calculate the manipulated variable by multiplying error by a proportional constant, K1, then adding a bias, see *Formula, p. 608*.

However, process conditions in most applications are changed by other system variables so that the bias does not remain constant; the result is offset error, where PV is constantly offset from the SP. This condition limits the capability of proportional-only control.

Note: The value in the integral term (in registers 4y + 3, 4y + 4, and 4y + 5) is always used, even when the integral mode is not enabled. Using this value is necessary to preserve bumpless transfer between modes. If you wish to disable bumpless transfer, these three registers must be cleared.

Proportional-Integral Control

To eliminate this offset error without forcing you to manually change the bias, an integral function can be added to the control equation, see *Formula, p. 608*. Proportional-integral control (PI) eliminates offset by integrating E as a function of time. K1 is the integral constant expressed as rep/min. As long as $E \neq 0$, the integrator increases (or decreases) its value, adjusting Mv. This continues until the offset error is eliminated.

Proportional-Integral-Derivative Control

You may want to add derivative functionality to the control equation to minimize the effects of frequent load changes or to override the integral function in order to get to the SP condition more quickly, see *Formula, p. 608*.

Proportional-integral-derivative (PID) control can be used to save energy in the process or as a safety valve in the event of a sudden, unexpected change in process flow. K3 is the derivative time constant expressed as min. DPV is the change in the process variable over a time period of Δt.

Example

An example to PID2 level control you will find in *PID2 Level Control Example, p. 37*.

Parameter Description

Source Block (Top Node)

The 4x register entered in the top node is the first of 21 contiguous holding registers in a source block. The contents of the fifth ... eighth implied registers determine whether the operation will be P, PI, or PID:

Operation	Fifth Implied	Sixth Implied	Seventh Implied	Eighth Implied
P	ON			ON
PI	ON	ON		
PID	ON	ON	ON	

The source block comprises the following register assignments:

Register	Name	Content
Displayed	Scaled PV	Loaded by the block each time it is scanned; a linear scaling is done on register 4x + 13 using the high and low ranges from registers 4x + 11 and 4x + 12: Scaled PV = (4x13 / 4095) * (4x11 - 4x12) + 4x12
First implied	SP	You must specify the set point in engineering units; the value must be < value in the 11th implied register and > value in the 12th implied register
Second implied	Mv	Loaded by the block every time the loop is solved; it is clamped to a range of 0 ... 4095, making the output compatible with an analog output module; the manipulated variable register may be used for further CPU calculations such as cascaded loops
Third implied	High Alarm Limit	Load a value in this register to specify a high alarm for PV (at or above SP); enter the value in engineering units within the range specified in the 11th and 12th implied registers
Fourth implied	Low Alarm Limit	Load a value in this register to specify a low alarm for PV (at or below SP); enter the value in engineering units within the range specified in the 11th and 12th implied registers
Fifth implied	Proportional Band	Load this register with the desired proportional constant in the range 5 ... 500; the smaller the number, the larger the proportional contribution; a valid number is required in this register for PID2 to operate

Register	Name	Content
Sixth implied	Reset Time Constant	Load this register to add integral action to the calculation; enter a value between 0000 ... 9999 to represent a range of 00.00 ... 99.99 repeats/min; the larger the number, the larger the integral contribution; a value > 9999 stops the PID2 calculation
Seventh implied	Rate Time Constant	Load this register to add derivative action to the calculation; enter a value between 0000 ... 9999 to represent a range of 00.00 ... 99.99 min; the larger the number, the larger the derivative contribution; a value > 9999 stops the PID2 calculation
Eighth implied	Bias	Load this register to add a bias to the output; the value must be between 000 .... 4095, and added directly to Mv, whether the integral term is enabled or not
Ninth implied	High Integral Windup Limit	Load this register with the upper limit of the output value (between 0 ... 4095) where the anti-reset windup takes effect; the updating of the integral sum is stopped if it goes above this value (this is normally 4095)
10th implied	Low Integral Windup Limit	Load this register with the lower limit of the output value (between 0 ... 4095) where the anti-reset windup takes effect (this is normally 0)
11th implied	High Engineering Range	Load this register with the highest value for which the measurement device is spanned, e.g. if a resistance temperature device ranges from 0 ... 500 degrees C, the high engineering range value is 500; the range must be given as a positive integer between 0001 ... 9999, corresponding to the raw analog input 4095
12th implied	Low Engineering Range	Load this register with the lowest value for which the measurement device is spanned; the range must be given as a positive integer between 0 ... 9998, and it must be less than the value in the 11th implied register; it corresponds to the raw analog input 0
13th implied	Raw Analog Measurement	The logic program loads this register with PV; the measurement must be scaled and linear in the range 0 ... 4095

Register	Name	Content
14th implied	Pointer to Loop Counter Register	The value you load in this register points to the register that counts the number of loops solved in each scan; the entry is determined by discarding the most significant digit in the register where the controller will count the loops solved/scan, e.g., if the PLC does the count in register 41236, load 1236 into the 14th implied register; the same value must be loaded into the 14th implied register in every PID2 block in the logic program
15th implied	Maximum Number of Loops	Solved In a Scan: If the 14th implied register contains a non-zero value, you may load a value in this register to limit the number of loops to be solved in one scan
16th implied	Pointer To Reset Feedback Input:	The value you load in this register points to the holding register that contains the value of feedback (F); drop the 4 from the feedback register and enter the remaining four digits in this register; integration calculations depend on the F value being should F vary from 0 ... 4095
17th implied	Output Clamp - High	The value entered in this register determines the upper limit of Mv (this is normally 4095)
18th implied	Output Clamp - Low	The value entered in this register determines the lower limit of Mv (this is normally 0)
19th implied	Rate Gain Limit (RGL) Constant	The value entered in this register determines the effective degree of derivative filtering; the range is from 2 ... 30; the smaller the value, the more filtering takes place
20th implied	Pointer to Integral Preload	The value entered in this register points to the holding register containing the track input (T) value; drop the 4 from the tracking register and enter the remaining four digits in this register; the value in the T register is connected to the input of the integral lag whenever the auto bit and integral preload bit are both true

Destination (MIddle Node)

The 4y register entered in the middle node is the first of nine contiguous holding register used for PID2 calculations. You do not need to load anything into these registers:

Register	Name	Content
Displayed	Loop Status Register	Twelve of the 16 bits in this register are used to define loop status.
First implied	Error (E) Status Bits	This register displays PID2 error codes.
Second implied	Loop Timer Register	This register stores the real-time clock reading on the system clock each time the loop is solved: the difference between the current clock value and the value stored in the register is the elapsed time; if elapsed time ≥ solution interval (10 times the value given in the bottom node of the PID2 block), then the loop should be solved in this scan
Third implied	For Internal Use	Integral (integer portion)
Fourth implied	For Internal Use	Integral-fraction 1 (1/3 000)
Fifth implied	For Internal Use	Integral-fraction 2 (1/600 000)
Sixth implied	Pv x 8 (Filtered)	This register stores the result of the filtered analog input (from register 4x14) multiplied by 8; this value is useful in derivative control operations
Seventh implied	Absolute Value of E	This register, which is updated after each loop solution, contains the absolute value of (SP - PV); bit 8 in register 4y + 1 indicates the sign of E
Eighth implied	For Internal Use	Current solution interval

Loop Status Register

LOOP Status

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

Bit	Function
1	Top output status (Node lockout or parameter error
2	Middle output status (High alarm)
3	Bottom output status (Low alarm)
4	Loop in AUTO mode and time since last solution ≥ solution interval
5	Wind-down mod (for REV B or higher)
6	Loop in AUTO mode but not being solved
7	4x14 register referenced by 4x15 is valid
8	Sign of E in 4y + 7: ● 0 = + (plus) ● 1 = - (minus)
9	Rev B or higher
10	Integral windup limit never set
11	Integral windup saturated
12	Negative values in the equation
13	Bottom input status (direct / reverse acting)
14	Middle input status (tracking mode) ● 1 = tracking ● 0 = no tracking
15	Top input status (MAN / AUTO)
16	Bit 16 is set after initial startup or installation of the loop. If you clear the bit, the following actions take place in one scan: ● The loop status register 4y is reset ● The current value in the real-time clock is stored in the first implied register (4y+1) ● Values in the third ... fifth registers (4y+2,3) are cleared ● The value in the13th implied register (4x+13) x 8 is stored in the sixth implied register (4y+6) ● The seventh and eighth implied registers (4y+7,8) are cleared

Solution Interval (Bottom Node)

The bottom node indicates that this is a PID2 function and contains a number ranging from 1 ... 255, indicating how often the function should be performed. The number represents a time value in tenths of a second, or example, the number 17 indicates that the PID function should be performed every 1.7 s.

Run Time Errors

Error Status Bit

The first implied register of the destination contains the error status bits:

Code	Explanation	Check these Registers in the Source Block (Top Node)
0000	No errors, all validations OK	None
0001	Scaled SP above 9999	First implied
0002	High alarm above 9999	Third implied
0003	Low alarm above 9999	Fourth implied
0004	Proportional band below 5	Fifth implied
0005	Proportional band above 500	Fifth implied
0006	Reset above 99.99 r/min	Sixth implied
0007	Rate above 99.99 min	Seventh implied
0008	Bias above 4095	Eighth implied
0009	High integral limit above 4095	Ninth implied
0010	Low integral limit above 4095	10th implied
0011	High engineering unit (E.U.) scale above 9999	11th implied
0012	Low E.U. scale above 9999	12th implied
0013	High E.U. below low E.U.	11th and 12th implied
0014	Scaled SP above high E.U.	First and 11th implied
0015	.Scaled SP below low E.U.	First and 12th implied
0016	Maximum loops/scan > 9999 **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero.	15th implied
0017	Reset feedback pointer out of range	16th implied
0018	High output clamp above 4095	17th implied
0019	Low output clamp above 4095	18th implied
0020	Low output clamp above high output clamp	17th and 18th implied
0021	RGL below 2	19th implied
0022	RGL above 30	19th implied
0023	Track F pointer out of range **Note**: Activated only if the track feature is ON, i.e. the middle input of the PID2 block is receiving power while in AUTO mode.	20th implied with middle input ON

Code	Explanation	Check these Registers in the Source Block (Top Node)
0024	Track F pointer is zero **Note**: Activated only if the track feature is ON, i.e. the middle input of the PID2 block is receiving power while in AUTO mode.	20th implied with middle input ON
0025	Node locked out (short of scan time) **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero. **Note:** If lockout occurs often and the parameters are all valid, increase the maximum number of loops/scan. Lockout may also occur if the counting registers in use are not cleared as required.	None
0026	Loop counter pointer is zero **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero.	14th and 15th implied
0027	Loop counter pointer out of range	14th and 15th implied

R —–> T: Register to Table

122

At a Glance

Introduction

This chapter describes the instruction R → T.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	622
Representation	622
Parameter Description	623

Short Description

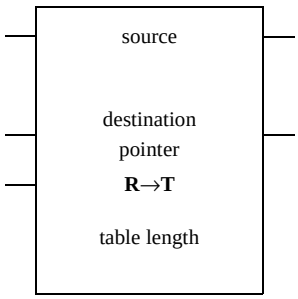
Function Description

The R→T instruction copies the bit pattern of a register or of a string of contiguous discretes stored in a word into a specific register located in a table. It can accommodate the transfer of one register/word per scan.

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 data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	Source data to be copied in the current scan
destination pointer (middle node)	4x	INT, UINT	Destination table where source data will be copied in the scan
table length (bottom node)		INT, UINT	Number of registers in the destination table, range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Top Input

The input to the top node initiates the DX move operation.

Middle Input

When the middle input goes ON, the current value stored in the destination pointer register is frozen while the DX operation continues. This causes new data being copied to the destination to overwrite the data copied on the previous scan.

Bottom Input

When the bottom input goes ON, the value in the destination pointer register is reset to zero. This causes the next DX move operation to copy source data into the first register in the destination table.

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

Destination Pointer (Middle Node)

The 4x register entered in the middle node is a pointer to the destination table where source data will be copied in the scan. The first register in the destination table is the next contiguous 4x register following the pointer, i.e. if the pointer register is 400027, then the destination table begins at register 400028.

The value posted in the pointer register indicates the register in the destination table where the source data will be copied. A value of zero indicates that the source data will be copied to the first register in the destination table; a value of 1 indicates that the source data be copied to the second register in the destination table; etc.

Note: The value posted in the destination pointer register cannot be larger than the table length integer specified in this node.

Outputs

R→T can produce two possible outputs, from the top and middle nodes. The state of the output from the top node echoes the state of the top input. The output from the middle node goes ON when the value in the destination pointer register equals the specified table length. At this point, the instruction cannot increment any further.

RBIT: Reset Bit

123

At a Glance

Introduction

This chapter describes the instruction RBIT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	626
Representation	626

Short Description

Function Description

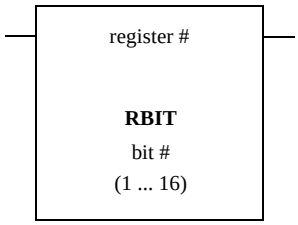
The reset bit (RBIT) instruction lets you clear a latched-ON bit by powering the top input. The bit remains cleared after power is removed from the input. This instruction is designed to clear a bit set by the SBIT instruction.

Note: The RBIT instruction does not follow the same rules of network placement as 0x-referenced coils do. An RBIT 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 = clears the specified bit to 0. The bit remains cleared after power is removed from the input
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 beeing cleared
Top output	0x	None	ON = the specified bit has been cleared to 0

READ: Read

124

At a Glance

Introduction

This chapter describes the instruction READ.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	628
Representation	629
Parameter Description	630

Short Description

Function Description

The READ instruction provides the ability to read data from an ASCII input device (keyboard, bar code reader, etc.) into the PLC's memory via its RIO network. The connection to the ASCII device is made at an RIO interface.

In the process of handling the messaging operation, READ performs the following functions:

- Verifies the lengths of variable data fields
- Verifies the correctness of the ASCII communication parameters, e.g. the port number, the message number
- Performs error detection and recording
- Reports RIO interface status

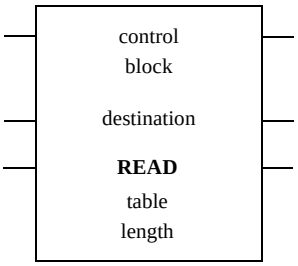
READ requires two tables of registers: a destination table where retrieved variable data (the message) is stored, and a control block where comm port and message parameters are identified.

Further information about formatting messages you will find in *Formatting Messages for ASCII READ/WRIT Operations, p. 41*.

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 READ
Middle input	0x, 1x	None	ON = pauses READ operation
Bottom input	0x, 1x	None	ON = abort READ operation
control block (top node)	4x	INT, UINT, WORD	Control block (first of seven contiguous holding registers)
destination (middle node)	4x	INT, UINT, WORD	Destination table
table length (bottom node)		INT, UINT	Length of destination table (number of registers where the message data will be stored), range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = error in communication or operation has timed out (for one scan)
Bottom output	0x	None	ON = READ complete (for one scan)

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of seven contiguous holding register in the control block.

Register	Definition
Displayed	*Port Number and Error Code, p. 630*
First implied	Message number
Second implied	Number of registers required to satisfy format
Third implied	Count of the number of registers transmitted thus far
Fourth implied	Status of the solve
Fifth implied	Reserved
Sixth implied	Checksum of registers 0 ... 5

Port Number and Error Code

Port Number and Error Code

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

Bit	Function
1 ... 4	PLC error code
5	Not used
6	Input from the ASCII device not compatible with format
7	Input buffer overrun, data received too quickly at RIOP
8	USART error, bad byte received at RIOP
9	Illegal format, not received properly by RIOP
10	ASCII device off-line, check cabling
11	ASCII message terminated early (in keyboard mode
12 ... 16	Comm port # (1 ... 32)

PLC Error Code

Bit				Meaning
1	**2**	**3**	**4**	
0	0	0	1	Error in the input to RIOP from ASCII device
0	0	1	0	Exception response from RIOP, bad data
0	0	1	1	Sequenced number from RIOP differs from expected value
0	1	0	0	User register checksum error, often caused by altering READ registers while the block is active
0	1	0	1	Invalid port or message number detected
0	1	1	0	User-initiated abort, bottom input energized
0	1	1	1	No response from drop, communication error
1	0	0	0	Node aborted because of SKP instruction
1	0	0	1	Message area scrambled, reload memory
1	0	1	0	Port not configured in the I/O map
1	0	1	2	Illegal ASCII request
1	1	0	0	Unknown response from ASCII port
1	1	0	1	Illegal ASCII element detected in user logic
1	1	1	1	RIOP in the PLC is down

Destination (Middle Node)

The middle node contains the first 4x register in a destination table. Variable data in a READ message are written into this table. The length of the table is defined in the bottom node.

Consider this READ message:

please enter password**:** **AAAAAAAAAA**

(Embedded Text) (Variable Data)

Note: An ASCII READ message may contain the embedded text, placed inside quotation marks, as well as the variable data in the format statement, i.e., the ASCII message.
The 10-character ASCII field **AAAAAAAAAA** is the variable data field; variable data must be entered via an ASCII input device.

RET: Return from a Subroutine

125

At a Glance

Introduction

This chapter describes the instruction RET.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	634
Representation	634

Short Description

Function Description

The RET instruction may be used to conditionally return the logic scan to the node immediately following the most recently executed JSR block. This instruction can be implemented only from within the subroutine segment, the (unscheduled) last segment in the user logic program.

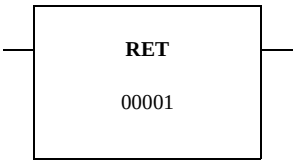
Note: If a subroutine does not contain a RET block, either a LAB block or the end-of-logic (whichever comes first) serves as the default return from the subroutine.

An example to the subroutine handling you will find in *Subroutine Handling, p. 51*.

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 = return to previous logic
00001		INT, UINT	Constant value, can not be changed
Top output	0x	None	ON = error in the specified subroutine

SAVE: Save Flash

126

At a Glance

Introduction

This chapter describes the instruction SAVE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	636
Representation	636
Parameter Description	637

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 SAVE instruction saves a block of 4x registers to 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 SAVE 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 saved to state RAM
1, 2, 3, 4 (middle node)		INT	Integer value, which defines the specific buffer where the block of data is to be saved
length (bottom node)		INT	Number of words to be saved, range: 1 ... 512
Top output	0x	None	ON = SAVE is active
Middle output	0x	None	ON = SAVE is not allowed

Parameter Description

1, 2, 3, 4 (Middle Node)	The middle node defines the specific buffer, within state RAM, where the block of data is to be saved. 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 save data to the same buffer without first loading it with the instruction LOAD. 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.
Middle Output	The output from the middle node goes ON when previously saved data has not been accessed using the LOAD instruction. This prevents inadvertent overwriting of data in the SAVE buffer.

SBIT: Set Bit

127

At a Glance

Introduction

This chapter describes the instruction SBIT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	640
Representation	640

Short Description

Function Description

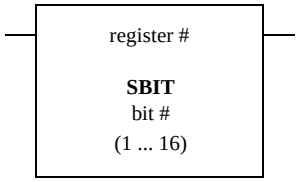
The set bit (SBIT) instruction lets you set the state of the specified bit to ON (1) by powering the top input.

Note: The SBIT instruction does not follow the same rules of network placement as 0x-referenced coils do. An SBIT 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. The bit remains set after power is removed from the input
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 set
Top output	0x	None	Goes ON, when the specified bit is set and remains ON until it is cleared (via the RBIT instruction)

SCIF: Sequential Control Interfaces

128

At a Glance

Introduction

This chapter describes the instruction SCIF.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	642
Representation	643
Parameter Description	644

Short Description

Function Description

The SCIF instruction performs either a drum sequencing operation or an input comparison (ICMP) using the data defined in the step data table.

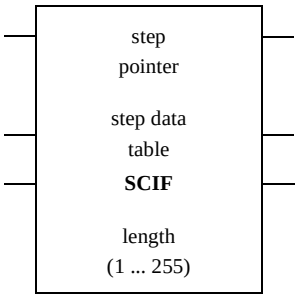
The choice of operation is made by defining the value in the first register of the step data table:

- **0 = drum mode:**
  The instruction controls outputs in the drum sequencing application.
- **1 = ICMP mode:**
  The instruction reads inputs to ensure that limit switches, proximity switches, pushbuttons, etc. are properly positioned to allow drum outputs to be fired.

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 specified sequence control operation
Middle input	0x, 1x	None	**Drum mode:** step pointer increments to the next step **ICMP mode:** compare status is shown at the middle output
Bottom input	0x, 1x	None	**Drum mode:** ON = reset step pointer to 0 **ICMP mode:** not used
step pointer (top node)	4x	INT, UINT	Number of the current step in the step data table
step data table (middle node)	4x	INT, UINT	First register in the step data table
length (bottom node)		INT, UINT	Number of application-specific registers used in the step data table
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	**Drum mode**: step pointer = length **ICMP mode**: indicates a valid input comparison
Bottom output	0x	None	ON = error is detected

Parameter Description

Step Data Table (Middle Node)

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

Register	Register Name	Description
Displayed	subfunction type	0 = drum mode; 1 = ICMP mode (entry of any other value in this register will result in all outputs OFF)
First implied	masked output data (in drum mode)	Loaded by SCIF each time the block is solved; the register contains the contents of the current step data register masked with the output mask register
	raw input data (in ICMP mode)	Loaded by the user from a group of sequential inputs to be used by the block in the current step
Second implied	current step data	Loaded by SCIF each time the block is solved; the register contains data from the current step (pointed to by the step pointer)
Third implied	output mask (in drum mode)	Loaded by the user before using the block, the contents will not be altered during logic solving; contains a mask to be applied to the data for each sequencer step
	input mask (in ICMP mode)	Loaded by the user before using the block, it contains a mask to be ANDed with raw input data for each step, masked bits will not be compared; the masked data are put in the masked input data register
Fourth implied	masked input data (in ICMP mode)	Loaded by SCIF each time the block is solved, it contains the result of the ANDed input mask and raw input data
	not used in drum mode	
Fifth implied	compare status (in ICMP mode)	Loaded by SCIF each time the block is solved, it 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 turn ON the middle output from the SCIF block
	not used in drum mode	
Sixth implied	start of data table	First of K registers in the table containing the user-specified control data **Note:** This and the rest of the registers represent application-specific step data in the process being controlled.

Length of Step Data Table (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 ... 255.

The total number of registers required in the step data table is the length + 7. The length must be ≥ the value placed in the steps used register in the middle node.

SENS: Sense

129

At a Glance

Introduction

This chapter describes the instruction SENS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	648
Representation	648
Parameter Description	649

Short Description

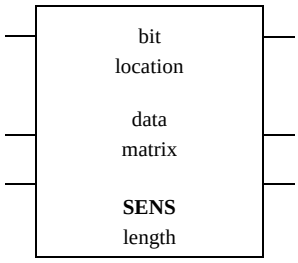
Function Description

The SENS instruction examines and reports the sense (1 or 0) of a specific bit location in a data matrix. One bit location is sensed per scan.

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 = senses the bit location
Middle input	0x, 1x	None	Increment bit location by one on next scan
Bottom input	0x, 1x	None	Reset bit location to 1
bit location (top node)	3x, 4x	WORD	Specific bit location to be sensed in the data matrix, entered explicitly as an integer or stored in a register; range: 1 ... 9 600
data matrix (middle node)	0x, 4x	BOOL, 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	ON = bit sense is 1 OFF = bit sense is 0
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 middle and bottom inputs.

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.

SKPC: Skip (Constants)

130

At a Glance

Introduction

This chapter describes the instruction SKPC.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	652
Representation	653
Parameter Description	653
Example	654

Short Description

Function Description

When a SKPC instruction is implemented, skipped networks in the ladder logic program are not solved. SKPC instructions can be used to reduce scan time and, in effect, establish subroutines within the scheduled logic.
A SKPC operation cannot pass the boundary of a segment. No matter how many extra networks you specify to be skipped, the instruction will stop if it reaches the end of a segment.

Note: A SKPC instruction can be activated only if you specify in the configurator editor that skips are allowed.

WARNING

Inputs and outputs could be unintentionally skipped or not skipped.

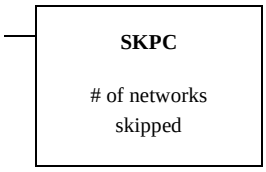
SKPC is a dangerous instruction that should be used carefully. If inputs and outputs that normally effect control are unintentionally skipped (or not skipped), the result can create hazardous conditions for personnel and application equipment.

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	ON = skip operation is performed on every scan
# of networks skipped (bottom node)	3x, 4x	INT	Number of networks to be skipped, specified explicitly as an integer constant (range: 1 ... 999) or stored in a register

Parameter Description

Number of Networks skipped (Bottom Node)

The value entered in the node specifies the number of networks to be skipped.

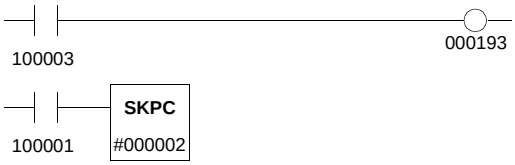
The node value includes the network that contains the SKPC instruction. The nodal regions in the network where the SKPC resides that have not already been scanned will be skipped; this counts as one of the networks specified to be skipped. The CPU continues to skip networks until the total number of networks skipped equals the value specified.

Example

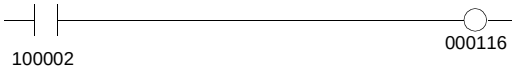
A simple SKPC Example

The illustration is showing two contiguous networks of ladder logic. The first network contains a SKPC instruction that specifies that two networks will be skipped when contact 100001 passes power.

Network 1



Network 2



When N.O. contact 100001 is closed, the remainder of the top network and all of the bottom network are skipped. The power flow display for these two networks becomes invalid, and your system displays an information message to that effect.

Coil 000193 is still controlled by contact 100003 because the solution of coil 000193 occurs before the SKPC instruction. Coil 000116 will remain in whatever state it was in when the bottom network was skipped.

SKPR: Skip (Registers)

131

At a Glance

Introduction

This chapter describes the instruction SKPR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	656
Representation	656
Parameter Description	657
Example	657

Short Description

Function Description

When a SKPR instruction is implemented, skipped networks in the ladder logic program are not solved. SKPR instructions can be used to reduce scan time and, in effect, establish subroutines within the scheduled logic.
A SKPR operation cannot pass the boundary of a segment. No matter how many extra networks you specify to be skipped, the instruction will stop if it reaches the end of a segment.

WARNING

Inputs and outputs could be unintentionally skipped or not skipped.

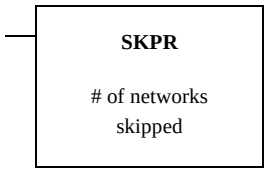
SKPR is a dangerous instruction that should be used carefully. If inputs and outputs that normally effect control are unintentionally skipped (or not skipped), the result can create hazardous conditions for personnel and application equipment.

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	ON = skip operation is performed on every scan
# of networks skipped (bottom node)	3x, 4x	INT	Number of networks to be skipped, specified explicitly as an integer constant (range: 1 ... 999) or stored in a register

Parameter Description

Number of Networks skipped (Bottom Node)

The value entered in the node specifies the number of networks to be skipped.

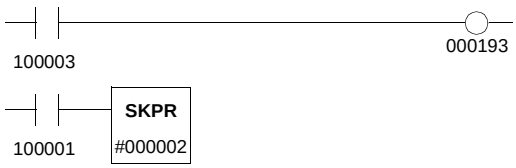
The node value includes the network that contains the SKPR instruction. The nodal regions in the network where the SKPR resides that have not already been scanned will be skipped; this counts as one of the networks specified to be skipped. The CPU continues to skip networks until the total number of networks skipped equals the value specified.

Example

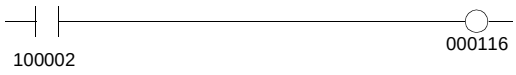
A simple SKPR Example

The illustration is showing two contiguous networks of ladder logic. The first network contains a SKPR instruction that specifies that two networks will be skipped when contact 100001 passes power.

Network 1:



Network 1:



When N.O. contact 100001 is closed, the remainder of the top network and all of the bottom network are skipped. The power flow display for these two networks becomes invalid, and your system displays an information message to that effect.

Coil 000193 is still controlled by contact 100003 because the solution of coil 000193 occurs before the SKPR instruction. Coil 000116 will remain in whatever state it was in when the bottom network was skipped.

SRCH: Search

132

At a Glance

Introduction

This chapter describes the instruction SRCH.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	660
Representation	660
Parameter Description	661

Short Description

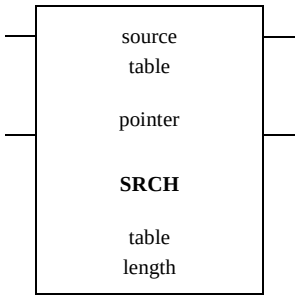
Function Description

The SRCH instruction searches the registers in a source table for a specific bit pattern.

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 search
Middle input	0x, 1x	None	OFF = search from beginning ON = search from last match
source table (top node)	3x, 4x	INT, UINT, WORD	Source table to be searched
pointer (middle node)	4x	INT, UINT	Pointer into the source table
table length (bottom node)		INT, UINT	Number of registers in the source table; range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = match found

Parameter Description

Pointer (Middle Node)

The 4x register entered in the middle node is the pointer into the source table. It points to the source register that contains the same value as the value stored in the next contiguous register after the pointer, e.g. if the pointer register is 400015, then register 400016 contains a value that the SRCH instruction will attempt to match in source table.

STAT: Status

133

At a Glance

Introduction

This chapter describes the instruction STAT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	664
Representation	664
Parameter Description	665
Description of the Status Table	665
Controller Status Words 1 - 11 for Quantum and Momentum	670
I/O Module Health Status Words 12 - 20 for Momentum	674
I/O Module Health Status Words 12 - 171 for Quantum	676
Communication Status Words 172 - 277 for Quantum	677
Controller Status Words 1 - 11 for TSX Compact and Atrium	682
I/O Module Health Status Words 12 - 15 for TSX Compact	685
Global Health and Communications Retry Status Words 182 ... 184 for TSX Compact	686

Short Description

Function Description

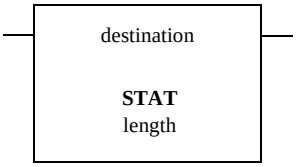
The STAT instruction accesses a specified number of words in a status table in the PLC's system memory. Here vital diagnostic information regarding the health of the PLC and its remote I/O drops is posted.

This information includes:
- PLC status
- Possible error conditions in the I/O modules
- Input-to-PLC-to-output communication status

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 specified number of words from the status table
destination (top node)	0x, 4x	INT, UINT, BOOL, WORD	First position in the destination block
length (bottom node)		INT, UINT	number of registers or 16-bit words in the destination block
Top output	0x	None	ON = operation successful

Parameter Description

Mode of Functioning

With the STAT instruction, you can copy some or all of the status words into a block of registers or a block of contiguous discrete references.

The copy to the STAT block always begins with the first word in the table up to the last word of interest to you. For example, if the status table is 277 words long and you are interested only in the statistics provided in word 11, you need to copy only words 1 ... 11 by specifying a length of 11 in the STAT instruction.

Destination Block (Top Node)

The reference number entered in the top node is the first position in the destination block, i.e. the block where the current words of interest from the status table will be copied.

The number of holding registers or 16-bit words in the destination block is specified in the bottom node (length).

Note: We recommend that you do not use discretes in the STAT destination node because of the excessive number required to contain status information.

Length (Bottom Node)

The integer value entered in the bottom node specifies the number of registers or 16-bit words in the destination block where the current status information will be written. The maximum allowable length the Quantum PLCs with S908 RIO protocol is 1 ... 277.

Description of the Status Table

General

The STAT instruction is used to display the Status of Controller and I/O system for Quantum, Atrium, TSX Compact and Momentum.

The first 11 status words are used by Quantum and Momentum in the same way and by TSX Compact and Atrium in the same way. The following have a different meaning for Quantum, TSX Compact and Momentum.

Quantum Overview

The 277 words in the status table are organized in three sections:

- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 171)
- I/O Communications Health (words 172 ... 277)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	Controller Status	01
2	Hot Standby Status	02
3	Controller Status	03
4	RIO Status	04
5	Controller Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	RIO Redundancy and Timeout	08
9	ASCII Message Status	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	Drop 1, Rack 1	0C
13	Drop 1, Rack 2	0D
. . .	. . . . .	. . .
16	Drop 1, Rack 5	0F
17	Drop 2, Rack 1	10
18	Drop 2, Rack 2	11
. . .	. . . . .	. . .
171	Drop 32, Rack 5	AB
172	S908 Startup Error Code	AC
173	Cable A Errors	AD
174	Cable A Errors	AE
175	Cable A Errors	AF
176	Cable B Errors	B0
178	Cable B Errors	B1
178	Cable B Errors	B2
179	Global Communication Errors	B3
180	Global Communication Errors	B4
181	Global Communication Errors	B5

Decimal Word	Word Content	Hex Word
182	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	B6
183	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Second word)	B7
184	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Third word)	B8
185	Drop 2 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	B9
. . .	. . . . .	. . .
275	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	113
276	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Second word)	114
277	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Third word)	115

Momentum Overview

The 20 words in the status table are organized in two sections:

- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 20)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	Controller Status	01
2	Hot Standby Status	02
3	Controller Status	03
4	RIO Status	04
5	Controller Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	RIO Redundancy and Timeout	08
9	ASCII Message Status	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	Local Momentum I/O Module Health	0C
13	I/O Bus Module Health	0D
14	I/O Bus Module Health	0E
15	I/O Bus Module Health	0F
16	I/O Bus Module Health	10
17	I/O Bus Module Health	11
18	I/O Bus Module Health	12
19	I/O Bus Module Health	13
20	I/O Bus Module Health	14

TSX Compact and Atrium Overview

The 184 words in the status table are organized in three sections:

- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 15)
- Not used (16 ... 181)
- Global Health and Communications retry status (words 182 ... 184)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	CPU Status	01
2	not used	02
3	Controller Status	03
4	not used	04
5	CPU Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	not used	08
9	not used	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	I/O Health Status Rack 1	0C
13	I/O Health Status Rack 2	0D
14	I/O Health Status Rack 3	0E
15	I/O Health Status Rack 4	0F
16 ... 181	not used	10 ... B5
182	Health Status	B6
183	I/O Error Counter	B7
184	PAB Bus Retry Counter	B8

Controller Status Words 1 - 11 for Quantum and Momentum

Controller Status (Word 1)

Word 1 displays the following aspects of the PLC status:

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

Bit	Function
1 - 5	Not used
6	1 = enable constant sweep
7	1 = enable single sweep delay
8	1 = 16 bit user logic 0 = 24 bit user logic
9	1 = AC power on
10	1 = RUN light OFF
11	1 = memory protect OFF
12	1 = battery failed
13 - 16	Not used

Hot Standby Status (Word 2)

Word 2 displays the Hot Standby status for 984 PLCs that use S911/R911 Hot Standby Modules:

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

Bit	Function
1	1 = S911/R911 present and healthy
2 - 10	Not used
11	0 = controller toggle set to A 1 = controller toggle set to B
12	0 = controllers have matching logic 1 = controllers do not have matching logic
13, 14	Remote system state: 0 1 = Off line (1 dec) 1 0 = primary (2 dec) 1 1 = standby (3 dec)
15, 16	Local system state: 0 1 = Off line (1 dec) 1 0 = primary (2 dec) 1 1 = standby (3 dec)

Controller Status (Word 3)

Word 3 displays more aspects of the controller status:

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

Bit	Function
1	1 = first scan
2	1 = start command pending
3	1 = constant sweep time exceeded
4	1 = Existing DIM AWARENESS
5 - 12	Not used
13 - 16	Single sweeps

RIO Status (Word 4)

Word 4 is used for IOP information:

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

Bit	Function
1	1 = IOP bad
2	1 = IOP time out
3	1 = IOP loop back
4	1 = IOP memory failure
5 - 12	Not used
13 - 16	00 = IO did not respond 01 = no response 02 = failed loopback

Controller Stop State (Word 5)

Word 5 displays the PLC's stop state conditions:

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

Bit	Function
1	1 = peripheral port stop
2	Extended memory parity error (for chassis mount controllers) or traffic cop/S908 error (for other controllers) If the bit = 1 in a **984B controller**, an error has been detected in extended memory; the controller will run, but the error output will be ON for XMRD/XMWT functions If the bit = 1 for **any other controller than a chassis mount**, then either a traffic cop error has been detected or the S908 is missing from a multi-drop configuration.
3	1 = controller in DIM AWARENESS
4	1 = illegal peripheral intervention
5	1 = segment scheduler invalid
6	1 = start of node did not start segment
7	1 = state RAM test failed
8	1 = invalid traffic cop
9	1 = watchdog timer expired
10	1 = real time clock error
11	CPU logic solver failed (for chassis mount controllers) or Coil Use TABLE (for other controllers) If the bit = 1 in a chassis mount controller, the internal diagnostics have detected CPU failure. If the bit = 1 in any controller other than a chassis mount, then the Coil Use Table does not match the coils in user logic.
12	1 = IOP failure
13	1 = invalid node
14	1 = logic checksum
15	1 = coil disabled in RUN mode (see **Caution** below)
16	1 = bad config

	CAUTION **Using a Quantum or 984-684E/785E PLC** If you are using a Quantum or 984-684E/785E PLC, bit 15 in word 5 is never set. These PLCs can be started and run with coils disabled in RUN (optimized) mode. Also all the bits in word 5 must be set to 0 when one of these PLCs is running. **Failure to observe this precaution can result in injury or equipment damage.**

Controller Stop State (Word 6)

Word 6 displays the number of segments in ladder logic; a binary number is shown:

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

Bit	Function
1 - 16	Number of segments (expressed as a decimal number)

Controller Stop State (Word 7)

Word 7 displays the address of the end-of-logic (EOL) pointer:

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

Bit	Function
1 - 16	EOL pointer address

RIO Redundancy and Timeout (Word 8)

Word 8 uses its most significant bit to display whether or not redundant coaxial cables are run to the remote I/O drops, and it uses its four least significant bits to display the remote I/O timeout constant:

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

Bit	Function
1	RIO redundant cables? 0 = NO 1 = YES
2 - 12	Not used
13 - 16	RIO timeout constant

ASCII Message Status (Word 9)

Word 9 uses its four least significant bits to display ASCII message status:

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

Bit	Function
1 ... 12	Not used
13	1 = Mismatch between numbers of messages and pointers
14	1 = Invalid message pointer
15	1 = Invalid message
16	1 = Message checksum error

RUN/LOAD/ DEBUG Status (Word 10)

Word 10 uses its two least significant bits to display RUN/LOAD/DEBUG status:

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

Bit	Function
1 ... 14	Not used
15, 15	0 0 = Debug (0 dec) 0 1 = Run (1 dec) 1 0 = Load (2 dec)

Word 11

This word is not used.

I/O Module Health Status Words 12 - 20 for Momentum

I/O Module Health Status

Status words 12 ... 20 display I/O module health status.

1 word is reserved for each of up to 1 Local drop, 8 words are used to represent the health of up to 128 I/O Bus Modules

Local Momentum I/O Module Health

Word 12 displays the Local Momentum I/O Module health:

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

Bit	Function
1	1 = Local Module
2 - 16	Not used

Momentum I/O Bus Module Health

Word 13 through 20 display the health status for Momentum I/O Bus Modules as follows:

Word	I/O Bus Modules
13	1 ... 16
14	17 ... 32
15	33 ... 48
16	49 ... 64
17	65 ... 80
18	81 ... 96
19	97 ... 112
20	113 ... 128

Each Word display the Momentum I/O Bus Module health as follows:

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

Bit	Function
1	1 = Module 1
2	1 = Module 2
3	1 = Module 3
4	1 = Module 4
5	1 = Module 5
6	1 = Module 6
7	1 = Module 7
8	1 = Module 8
9	1 = Module 9
10	1 = Module 10
11	1 = Module 11
12	1 = Module 12
13	1 = Module 13
14	1 = Module 14
15	1 = Module 15
16	1 = Module 16

I/O Module Health Status Words 12 - 171 for Quantum

RIO Status Words

Status words 12 ... 20 display I/O module health status.

Five words are reserved for each of up to 32 drops, one word for each of up to five possible racks (I/O housings) in each drop. Each rack may contain up to 11 I/O modules; bits 1 ... 11 in each word represent the health of the associated I/O module in each rack.

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

Bit	Function
1	1 = Slot 1
2	1 = Slot 2
3	1 = Slot 3
4	1 = Slot 4
5	1 = Slot 5
6	1 = Slot 6
7	1 = Slot 7
8	1 = Slot 8
9	1 = Slot 9
10	1 = Slot 10
11	1 = Slot 11
12	1 = Slot 12
13	1 = Slot 13
14	1 = Slot 14
15	1 = Slot 15
16	1 = Slot 16

Four conditions must be met before an I/O module can indicate good health:

- The slot must be traffic copped
- The slot must contain a module with the correct personality
- Valid communications must exist between the module and the RIO interface at remote drops
- Valid communications must exist between the RIO interface at each remote drop and the I/O processor in the controller

Status Words for the MMI Operator Panels

The status of the 32 Element Pushbutton Panels and PanelMate units on an RIO network can also be monitored with an I/O health status word. The Pushbutton Panels occupy slot 4 in an I/O rack and can be monitored at bit 4 of the appropriate status word. A PanelMate on RIO occupies slot 1 in rack 1 of the drop and can be monitored at bit 1 of the first status word for the drop.

Note: The ASCII Keypad's communication status can be monitored with the error codes in the ASCII READ/WRIT blocks.

Communication Status Words 172 - 277 for Quantum

DIO Status

Status words 172 ... 277 contain the I/O system communication status. Words 172 ... 181 are global status words. Among the remaining 96 words, three words are dedicated to each of up to 32 drops, depending on the type of PLC.

Word 172 stores the Quantum Startup Error Code. This word is always 0 when the system is running. If an error occurs, the controller does not start-it generates a stop state code of 10 (word 5 (See *Controller Stop State (Word 5), p. 672*)).

Quantum Start-up Error Codes

Code	Error	Meaning (Where the error has occurred)
01	BADTCLEN	Traffic Cop length
02	BADLNKNUM	Remote I/O link number
03	BADNUMDPS	Number of drops in Traffic Cop
04	BADTCSUM	Traffic Cop checksum
10	BADDDLEN	Drop descriptor length
11	BADDRPNUM	I/O drop number
12	BADHUPTIM	Drop holdup time
13	BADASCNUM	ASCII port number
14	BADNUMODS	Number of modules in drop
15	PRECONDRP	Drop already configured
16	PRECONPRT	Port already configured
17	TOOMNYOUT	More than 1024 output points
18	TOOMNYINS	More than 1024 input points
20	BADSLTNUM	Module slot address
21	BADRCKNUM	Module rack address
22	BADOUTBC	Number of output bytes

Code	Error	Meaning (Where the error has occurred)
23	BADINBC	Number of input bytes
25	BADRF1MAP	First reference number
26	BADRF2MAP	Second reference number
27	NOBYTES	No input or output bytes
28	BADDISMAP	Discrete not on 16-bit boundary
30	BADODDOUT	Unpaired odd output module
31	BADODDIN	Unpaired odd input module
32	BADODDREF	Unmatched odd module reference
33	BAD3X1XRF	1x reference after 3x register
34	BADDMYMOD	Dummy module reference already used
35	NOT3XDMY	3x module not a dummy
36	NOT4XDMY	4x module not a dummy
40	DMYREAL1X	Dummy, then real 1x module
41	REALDMY1X	Real, then dummy 1x module
42	DMYREAL3X	Dummy, then real 3x module
43	REALDMY3X	Real, then dummy 3x module

Status of Cable A

Words 173 ... 175 are Cable A error words:

Word 173

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

Bit	Function
1 ... 8	Counts framing errors
9 ... 16	Counts DMA receiver overruns

Word 174

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

Bit	Function
1 ... 8	Counts receiver errors
9 ... 16	Counts bad drop receptions

Word 175

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

Bit	Function
1	1 = Short frame
2	1 = No end-of- frame
3 ... 12	Not used
13	1 = CRC error
14	1 = Alignment error
15	1 =Overrun error
16	Not used

Status of Cable B

Words 176 ... 178 are Cable A error words:

Word 176

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

Bit	Function
1 ... 8	Counts framing errors
9 ... 16	Counts DMA receiver overruns

Word 177

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

Bit	Function
1 ... 8	Counts receiver errors
9 -...16	Counts bad drop receptions

Word 178

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

Bit	Function
1	1 = Short frame
2	1 = No end-of- frame
3 ... 12	Not used
13	1 = CRC error
14	1 = Alignment error
15	1 =Overrun error
16	Not used

Status of Global Communication (Words 179 ... 181)

Word 179 displays global communication status:

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

Bit	Function
1	1 = Comm health
2	1 = Cable A status
3	1 = Cable B status
4	Not used
5 ... 8	Lost communication counter
9 ... 16	Cumulative retry counter

Word 180 is the global cumulative error counter for Cable A:

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

Bit	Function
1 ... 8	Counts detected errors
9 ... 162	Counts No responses

Word 181 is the global cumulative error counter for Cable B:

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

Bit	Function
1 ... 8	Counts detected errors
9 ... 162	Counts No responses

Status of Remote I/O (Words 182 ... 277)

Words 182 ... 277 are used to describe remote I/O drop status; three status words are used for each drop.

The first word in each group of three displays communication status for the appropriate drop:

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

Bit	Function
1	1 = Communication health
2	1 = Cable A status
3	1 = Cable B status
4	Not used
5 ... 8	Lost communication counter
9 ... 16	Cumulative retry counter

The second word in each group of three is the drop cumulative error counter on Cable A for the appropriate drop:

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

Bit	Function
1 ... 8	At least one error in words 173 ...175
9 ... 162	Counts No responses

The third word in each group of three is the drop cumulative error counter on Cable B for the appropriate drop:

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

Bit	Function
1 ... 8	At least one error in words 176 ...178
9 ... 162	Counts No responses

Note: For PLCs where drop 1 is reserved for local I/O, status words 182 ... 184 are used as follows:

Word 182 displays local drop status:

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

Bit	Function
1	1 = All modules healthy
2 ... 8	Always 0
9 ... 162	Number of times a module has been seen as unhealthy; counter rolls over at 255

Word 183 is used as a 16-bit I/O bus error counter.

Word 184 is used as a 16-bit I/O bus retry counter.

Controller Status Words 1 - 11 for TSX Compact and Atrium

CPU Status (Word 1)

Word 1 displays the following aspects of the CPU status:

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

Bit	Function
1 - 5	Not used
6	1 = enable constant sweep
7	1 = enable single sweep delay
8	1 = 16 bit user logic 0 = 24 bit user logic
9	1 = AC power on
10	1 = RUN light OFF
11	1 = memory protect OFF
12	1 = battery failed
13 - 16	Not used

Word 2

This word is not used.

Controller Status (Word 3)

Word 3 displays aspects of the controller status:

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

Bit	Function
1	1 = first scan
2	1 = start command pending
3	1 = scan time has exceed constant scan target
4	1 = existing DIM AWARENESS
5 - 12	Not used
13 - 16	Single sweeps

Word 4

This word is not used.

CPU Stop State (Word 5)

Word 5 displays the CPU's stop state conditions:

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

Bit	Function
1	1 = peripheral port stop
2	1 = XMEM parity error
3	1 = DIM AWARENESS
4	1 = illegal peripheral intervention
5	1 = invalid segment scheduler
6	1 = no start-of-network (SON) at the start of a segment
7	1 = state RAM test failed
8	1 = no end of logic (EOL), (bad Tcop)
9	1 = watch dog timer has expired
10	1 = real time clock error
11	1 = CPU failure
12	Not used
13	1 = invalid node in ladder logic
14	1 = logic checksum error
1	1 = coil disabled in RUN mode
16	1 = bad PLC setup

Number of Segments in program (Word 6)

Word 6 displays the number of segments in ladder logic; a binary number is shown. This word is confirmed during power up to be the number of EOS (DOIO) nodes plus 1 (for the end of logic nodes), if untrue, a stop code is set, causing the run light to be off:

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

Bit	Function
1 - 16	Number of segments in the current ladder logic program (expressed as a decimal number)

Address of the End of Logic Pointer (Word 7)

Word 7 displays the address of the end-of-logic (EOL) pointer:

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

Bit	Function
1 - 16	EOL pointer address

Word 8, Word 9

These words are not used.

RUN/LOAD/ DEBUG Status (Word 10)

Word 10 uses its two least significant bits to display RUN/LOAD/DEBUG status:

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

Bit	Function
1 ... 14	Not used
15, 16	0 0 = Debug (0 dec) 0 1 = Run (1 dec) 1 0 = Load (2 dec)

Word 11

This word is not used.

I/O Module Health Status Words 12 - 15 for TSX Compact

TSX Compact I/O Module Health

Words 12 ... 15 are used to display the health of the A120 I/O modules in the four racks:

Word	Rack No.
12	1
13	2
14	3
15	4

Each word contains the health status of up to five A120 I/O modules. The most significant (left-most) bit represents the health of the module in Slot 1 of the rack:

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

Bit	Function
1	1 = Slot 1
2	1 = Slot 2
3	1 = Slot 3
4	1 = Slot 4
5	1 = Slot 5
6 ... 16	Not used

If a module is I/O Mapped and ACTIVE, the bit will have a value of "1". If a module is inactive or not I/O Mapped, the bit will have a value of "0".

Note: Slots 1 and 2 in Rack 1 (Word 12) are not used because the controller itself uses those two slots.

Global Health and Communications Retry Status Words 182 ... 184 for TSX Compact

Overview

There are three words that contain health and communication information on the installed I/O modules. If monitored with the Stat block, they are found in Words 182 through 184. This requires that the length of the Stat block is a minimum of 184 (Words 16 through 181 are not used).

Words 16 ... 181

These words are not used.

Health Status (Word 182)

Word 182 increments each time a module becomes bad. After a module becomes bad, this counter does not increment again until that module becomes good and then bad again.

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

Bit	Function
1	1 = All modules healthy
2 ... 9	Not used
10 ... 16	"Module went unhealthy" counter

I/O Error Counter (Word 183)

This counter is similar to the above counter, except this word increments every scan that a module remains in the bad state.

PAB Bus Retry Counter (Word 184)

Diagnostics are performed on the communications through the bus. This word should normally be all zeroes. If after 5 retries, a bus error is still detected, the controller will stop and error code 10 will be displayed. An error could occur if there is a short in the backplane or from noise. The counter rolls over while running. If the retries are less than 5, no bus error is detected.

SU16: Subtract 16 Bit

134

At a Glance

Introduction

This chapter describes the instruction SU16.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	688
Representation	688

Short Description

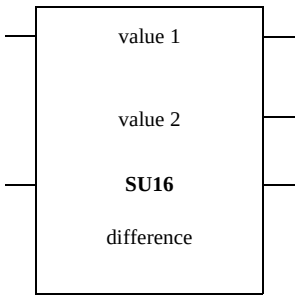
Function Description

The SU16 instruction performs a signed or unsigned 16-bit subtraction (value 1 - value 2) on the top and middle node values, then posts the signed or unsigned difference in a 4x holding register 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 - value 2
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Minuend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Subtrahend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
difference (bottom node)	4x	INT, UINT	Difference
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

SUB: Subtraction

135

At a Glance

Introduction

This chapter describes the instruction SUB.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	690
Representation	690

Short Description

Function Description

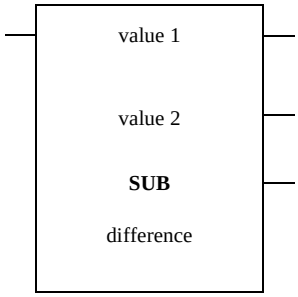
The SUB instruction performs a signed or unsigned 16-bit subtraction (value 1 - value 2) on the top and middle node values, then posts the signed or unsigned difference in a 4x holding register in the bottom node.

Note: SUB is often used as a comparator where the state of the outputs identifies whether value 1 is greater than, equal to, or less than value 2.

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 - value 2
value 1 (top node)	3x, 4x	INT, UINT	Minuend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Subtrahend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
difference (bottom node)	4x	INT, UINT	Difference
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

T—–>R: Table to Register

136

At a Glance

Introduction

This chapter describes the instruction T→R.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	692
Representation	692
Parameter Description	693

Short Description

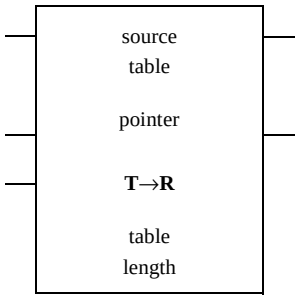
Function Description

The T→R instruction copies the bit pattern of a register or 16 contiguous discretes in a table to a specific holding register. It can accommodate the transfer of one register per scan. It has three control inputs and produces two 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 data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register or discrete reference in the source table. A register or string of contiguous discretes from this table will be copied in a scan.
pointer (middle node)	4x	INT, UINT	Pointer to the destination where the source data will be copied
table length (bottom node)		INT, UINT	Length of the source table: number of registers that may be copied; range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Middle Input	When the middle input goes ON, the current value stored in the pointer register is frozen while the DX operation continues. This causes the same table data to be written to the destination register on each scan.
Bottom Input	When the bottom input goes ON, the value in the pointer is reset to zero. This causes the next DX move operation to copy the first destination register in the table.
Pointer (Middle Node)	The 4x register entered in the middle node is a pointer to the destination where the source data will be copied. The destination register is the next contiguous 4x register after the pointer. For example, if the middle node displays a pointer of 400100, then the destination register for the T→R copy is 400101. The value stored in the pointer register indicates which register in the source table will be copied to the destination register in the current scan. A value of 0 in the pointer indicates that the bit pattern in the first register of the source table will be copied to the destination; a value of 1 in the pointer register indicates that the bit pattern in the second register of the source table will be copied to the destination register; etc.

T—>T: Table to Table

137

At a Glance

Introduction

This chapter describes the instruction T→T.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	696
Representation	696
Parameter Description	697

Short Description

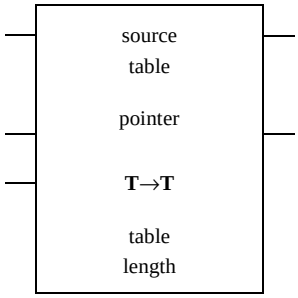
Function Description

The T→T instruction copies the bit pattern of a register or of 16 discretes from a position within one table to an equivalent position in another table of registers. It can accommodate the transfer of one register per scan. It has three control inputs and produces two 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 data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register or discrete reference in the source table. A register or string of contiguous discretes from this table will be copied in a scan.
pointer (middle node)	4x	INT, UINT	Pointer into both the source and destination table
table length (bottom node)		INT, UINT	Length of the source and the destination table (must be equal in length); range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Middle Input

When the input to the middle node goes ON, the current value stored in the pointer register is frozen while the DX operation continues. This causes new data being copied to the destination to overwrite the data copied on the previous scan.

Bottom Input

When the input to the bottom node goes ON, the value in the pointer register is reset to zero. This causes the next DX move operation to copy source data into the first register in the destination table.

Pointer (Middle Node)

The 4x register entered in the middle node is a pointer into both the source and destination tables, indicating where the data will be copied from and to in the current scan. The first register in the destination table is the next contiguous 4x register following the pointer. For example, if the middle node displays a a pointer reference of 400100, then the first register in the destination table is 400101.

The value stored in the pointer register indicates which register in the source table will be copied to which register in the destination table. Since the length of the two tables is equal and T→T copy is to the equivalent register in the destination table, the current value in the pointer register also indicates which register in the destination table the source data will be copied to.

A value of 0 in the pointer register indicates that the bit pattern in the first register of the source table will be copied to the first register of the destination table; a value of 1 in the pointer register indicates that the bit pattern in the second register of the source table will be copied to the second register of the destination register; etc.

T.01 Timer: One Hundredth Second Timer

138

At a Glance

Introduction

This chapter describes the instruction T.01 Timer.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	700
Representation	700

Short Description

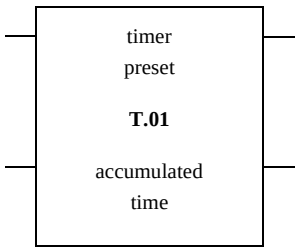
Function Description

The T.01 instruction measures time in hundredth of a second intervals. It can be used for timing an event or creating a delay. T.01 has two control inputs and can produce one of two 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	OFF → ON = initiates the timer operation: time accumulates in hundredths-of-a-second when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of hundredth-of-a-second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in hundredth-of-a-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T0.1 Timer: One Tenth Second Timer

139

At a Glance

Introduction

This chapter describes the instruction T0.1 Timer.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	702
Representation	702

Short Description

Function Description

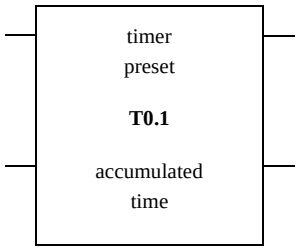
The T0.1 instruction measures time in tenth-of-a-second increments. It can be used for timing an event or creating a delay. T0.1 has two control inputs and can produce one of two possible outputs.

Note: If you cascade T0.1 timers with presets of 1, the timers will time-out together; to avoid this problem, change the presets to 10 and substitute a T.01 timer.

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	OFF → ON = initiates the timer operation: time accumulates in tenth-of-a-second when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of tenth-of-a-second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in tenth-of-a-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T1.0 Timer: One Second Timer

140

At a Glance

Introduction

This chapter describes the instruction T1.0 Timer.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	704
Representation	704

Short Description

Function Description

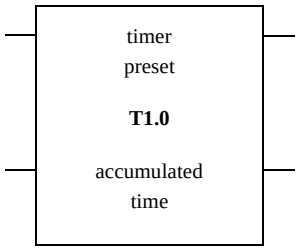
The T1.0 timer instruction measures time in one-second increments. It can be used for timing an event or creating a delay. T1.0 has two control inputs and can produce one of two possible outputs.

Note: If you cascade T1.0 timers with presets of 1, the timers will time-out together; to avoid this problem, change the presets to 10 and substitute a T0.1 timer.

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	OFF → ON = initiates the timer operation: time accumulates in seconds when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of one second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in one-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T1MS Timer: One Millisecond Timer

141

At a Glance

Introduction

This chapter describes the instruction T1MS Timer.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	706
Representation	707
Example	708

Short Description

Function Description

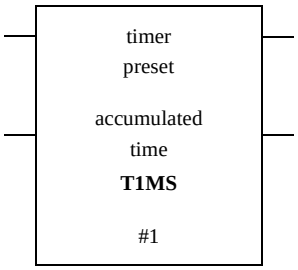
Note: This instruction is available in Micro PLC models and the Quantum CPU 424 02 PLC.

The T1MS timer instruction measures time in one-millisecond increments. It can be used for timing an event or creating a delay.

Representation

Symbol

Representation of the instruction



Parameter Description

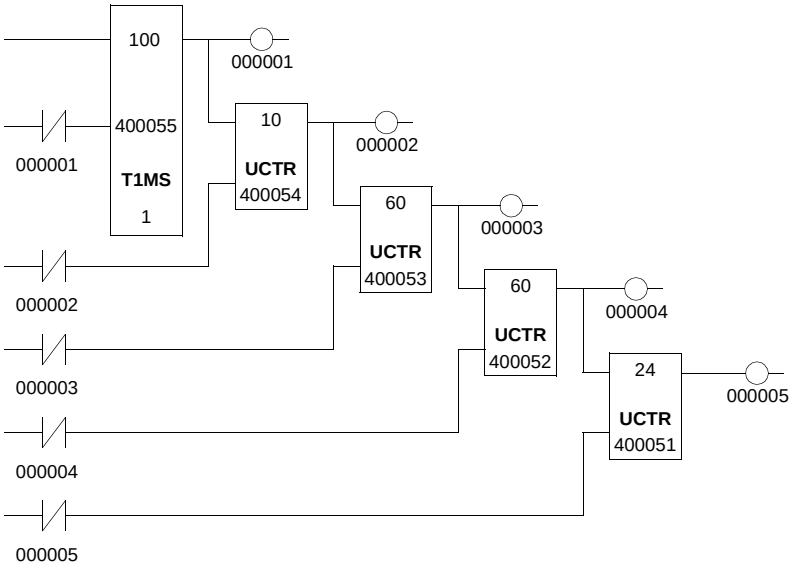
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates the timer operation: time accumulates in milliseconds when top and middle input are ON
Middle input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of millisecond increments the timer can accumulate), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (middle node)	4x	INT, UINT	Accumulated time count in millisecond increments.
#1 (bottom node)		INT, UINT	Constant value of #1
Top output	0x	None	ON = accumulated time = timer preset
Middle output	0x	None	ON = accumulated time < timer preset

Example

A Millisecond Timer Example

Here is the ladder logic for a real-time clock with millisecond accuracy:



The T1MS instruction is programmed to pass power at 100 ms intervals; it is followed by a cascade of four up-counters (See *UCTR: Up Counter, p. 717*) that store the time respectively in hundredth-of-a-second units, tenth-of-a-second units, one- second units, one-minute units, and one-hour units.

When logic solving begins, the accumulated time value begins incrementing in register 40055 of the T1MS block. After 100 one-ms increments, the top output passes power and energizes coil 00001. At this point, the value in register 40055 in the timer is reset to 0. The accumulated count value in register 40054 in the first UCTR block increments by 1, indicating that 100 ms have passed. Because the accumulated time count in T1MS no longer equals the timer preset, the timer begins to re-accumulate time in ms.

When the accumulated count in register 40054 of the first UCTR instruction increments to 10, the top output from that instruction block passes power and energizes coil 00002. The value in register 40054 then resets to 0, and the accumulated count in register 40053 of the second UCTR block increments by 1.

As the times accumulate in each counter, the time of day can be read in five holding registers as follows:

Register	Unit of Time	Valid Range
40055	Thousandths-of-a-second	0 ... 100
40054	Tenths-of-a-second	0 ... 10
40053	Seconds	0 ... 60
40052	Minutes	0 ... 60
40051	Hours	0 ... 24

TBLK: Table to Block

142

At a Glance

Introduction

This chapter describes the instruction TBLK.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	712
Representation	712
Parameter Description	713

Short Description

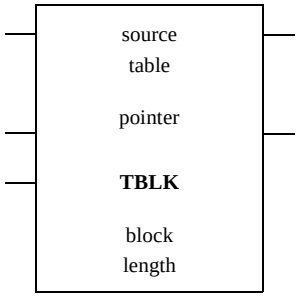
Function Description

The TBLK (table-to-block) instruction combines the functions of T→R and the BLKM in a single instruction. In one scan, it can copy up to 100 contiguous 4x registers from a table to a destination block. The destination block is of a fixed length. The block of registers being copied from the source table is of the same length, but the overall length of the source table is limited only by the number of registers in your system configuration.

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 move operation
Middle input	0x, 1x	None	ON = hold pointer
Bottom input	0x, 1x	None	ON = reset pointer to zero
source table (top node)	4x	INT, UINT, WORD	First holding register in the source table
pointer (middle node)	4x	INT, UINT	Pointer to the source block, destination block
block length (bottom node)		INT, UINT	Number of registers of the destination block and of the blocks within the source table; range: 1 ... 100
Top output	0x	None	ON = move successful
Middle output	0x	None	ON = error / move not possible

Parameter Description

Middle Input

When the middle input is ON, the value in the pointer register is frozen while the TBLK operation continues. This causes the same source data block to be copied to the destination table on each scan.

Bottom Input

When the bottom input is ON, the pointer value is reset to zero. This causes the TBLK operation to copy data from the first block of registers in the source table.

CAUTION

Confine the value in the destination pointer to a safe range.

You should use external logic in conjunction with the middle and the bottom inputs to confine the value in the destination pointer to a safe range.

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

Source Table (Top Node)

The 4x register entered in the top node is the first holding register in the source table.

Note: The source table is segmented into a series of register blocks, each of which is the same length as the destination block. Therefore, the size of the source table is a multiple of the length of the destination block, but its overall size is not specifically defined in the instruction. If left uncontrolled, the source table could consume all the 4x registers available in the PLC configuration.

Pointer (Middle Node)

The 4x register entered in the middle node is the pointer to the source block. The first register in the destination block is the next contiguous register after the pointer. For example, if the pointer is register 400107, then the first register in the destination block is 400108.

The value stored in the pointer indicates which block of data from the source table will be copied to the destination block. This value specifies a block number within the source table.

TEST: Test of 2 Values

143

At a Glance

Introduction

This chapter describes the instruction TEST.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	716
Representation	716

Short Description

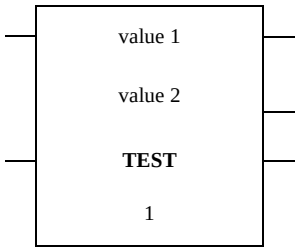
Function Description

The TEST instruction compares the signed or unsigned size of the 16-bit values in the top and middle nodes and describes the relationship via the block 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 = compares value 1 and value 2
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Value 1, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Value 2, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
1 (bottom node)		INT, UINT	Constant value, cannot be changed
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

UCTR: Up Counter

144

At a Glance

Introduction

This chapter describes the instruction UCTR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	718
Representation	718

Short Description

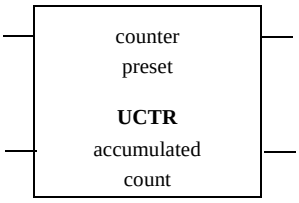
Function Description

The UCTR instruction counts control input transitions from OFF to ON up from zero to a counter preset value.

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	OFF → ON = initiates the counter operation
Bottom input	0x, 1x	None	OFF = reset accumulator to 0 ON = counter accumulating
counter preset (top node)	3x, 4x	INT, UINT	Preset value, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated count (bottom node)	4x	INT, UINT	Count value (actual value); which increments by one on each transition from OFF to ON of the top input until it reaches the specified counter preset value.
Top output	0x	None	ON = accumulated count = counter preset
Bottom output	0x	None	ON = accumulated count < counter preset

WRIT: Write

145

At a Glance

Introduction

This chapter describes the instruction WRIT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	720
Representation	721
Parameter Description	722

Short Description

Function Description

The WRIT instruction sends a message from the PLC over the RIO communications link to an ASCII display (screen, printer, etc.).

In the process of sending the messaging operation, WRIT performs the following functions:

- Verifies the correctness of the ASCII communication parameters, e.g. the port number, the message number
- Verifies the lengths of variable data fields
- Performs error detection and recording
- Reports RIO interface status

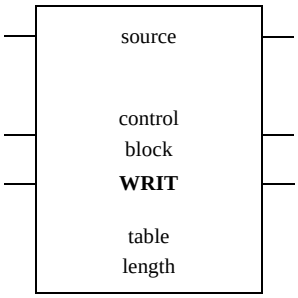
WRIT requires two tables of registers: a source table where variable data (the message) is copied, and a control block where comm port and message parameters are identified.

Further information about formatting messages you will find in *Formatting Messages for ASCII READ/WRIT Operations, p. 41*.

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 WRIT
Middle input	0x, 1x	None	ON = pauses WRIT operation
Bottom input	0x, 1x	None	ON = abort WRIT operation
source (top node)	3x, 4x	INT, UINT, WORD	Source table
control block (middle node)	4x	INT, UINT, WORD	ASCII Control block (first of seven contiguous holding registers)
table length (bottom node)		INT, UINT	Length of source table (number of registers where the message data will be stored), range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = error in communication or operation has timed out (for one scan)
Bottom output	0x	None	ON = WRIT complete (for one scan)

Parameter Description

Source Table (Top Node)

The top node contains the first 3x or 4x register in a source table whose length is specified in the bottom node. This table contains the data required to fill the variable field in a message.

Consider the following WRIT message

vessel #1 temperature is**:** **III**

The 3-character ASCII field **`III`** is the variable data field; variable data are loaded, typically via DX moves, into a table of variable field data.

Control Block (Middle Node)

The 4x register entered in the middle node is the first of seven contiguous holding register in the control block.

Register	Definition
Displayed	*Port Number and Error Code, p. 723*
First implied	Message number
Second implied	Number of registers required to satisfy format
Third implied	Count of the number of registers transmitted thus far
Fourth implied	Status of the solve
Fifth implied	Reserved
Sixth implied	Checksum of registers 0 ... 5

Port Number and Error Code

Port Number and Error Code

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

Bit	Function
1 ... 4	PLC error code (see table below)
5	Not used
6	Input from the ASCII device not compatible with format
7	Input buffer overrun, data received too quickly at RIOP
8	USART error, bad byte received at RIOP
9	Illegal format, not received properly by RIOP
10	ASCII device off-line, check cabling
11	ASCII message terminated early (in keyboard mode
12 ... 16	Comm port # (1 ... 32)

PLC Error Code

Bit				Meaning
1	**2**	**3**	**4**	
0	0	0	1	Error in the input to RIOP from ASCII device
0	0	1	0	Exception response from RIOP, bad data
0	0	1	1	Sequenced number from RIOP differs from expected value
0	1	0	0	User register checksum error, often caused by altering READ registers while the block is active
0	1	0	1	Invalid port or message number detected
0	1	1	0	User-initiated abort, bottom input energized
0	1	1	1	No response from drop, communication error
1	0	0	0	Node aborted because of SKP instruction
1	0	0	1	Message area scrambled, reload memory
1	0	1	0	Port not configured in the I/O map
1	0	1	1	Illegal ASCII request
1	1	0	0	Unknown response from ASCII port
1	1	0	1	Illegal ASCII element detected in user logic
1	1	1	1	RIOP in the PLC is down

XMIT: XMIT Communication Block

146

At a Glance

Introduction

This chapter describes the instruction XMIT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	726
Representation	727
Detailed Description	728

Short Description

Function Requirements

The following steps are necessary before using this instruction:

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

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

Step	Action
2	Unpack and install the DX Loadable XMIT; further information you will find in the chapter *Installation of DX Loadables, p. 53*.

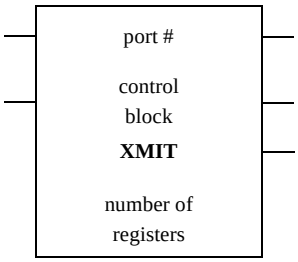
Function Description

The XMIT instruction is provided to receive and transmit ASCII messages and Modbus master messages using the PLC ports.

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 = Start XMIT operation
Middle input	0x, 1x	None	Abort XMIT operation
port # (top node)	4x	INT	PLC port selection
control block (middle node)	4x	INT, UINT	Control block (first of sixteen contiguous registers)
length (bottom node)		INT	Number of registers used by the XMIT instruction (must be a constant equal to 16)
Top output	0x, 1x	None	Operation is active
Middle output	0x, 1x	None	Operation terminated unsuccessfully (error detected or aborted)
Bottom output	0x, 1x	None	Operation has been successfully completed

Detailed Description

Mode of Functioning

The XMIT (Transmit) instruction sends Modbus messages from a master PLC to multiple slave PLCs or sends ASCII character strings from the PLC's Modbus slave port#1 or port#2 to ASCII printers and terminals. XMIT sends these messages over telephone dialup modems, radio modems, or simply direct connection.

XMIT comes with three modes:
- communication mode
- port status mode
- conversion mode.

XMIT performs general ASCII input functions in the communication mode including simple ASCII and terminated ASCII. You may use an additional XMIT block for reporting port status information into registers while another XMIT block performs the ASCII communication function. You may import and export ASCII or binary data into your PLC and convert it into various binary data or ASCII to send to DCE (Data Communication Equipment) devices based upon the needs of your application. The block has built-in diagnostics that checks to make sure no other XMIT blocks are active in the PLC. Within the XMIT block a control table allows you to control the communications link between the PLC and DCE (Data Communication Equipment) devices attached to Modbus port #1 or port#2 of the PLC. The XMIT block does NOT activate the port LED when it is transmitting data.
Further information you will find in the *Modicon XMIT Function Block User Guide*.

	CAUTION
	Contention and Collision when using the XMIT instruction in a network with multiple masters Remember, the Modbus protocol is a master/ slave protocol. Modbus is designed to have only one master polling multiple slaves. Therefore, when using the XMIT instruction in a network with multiple masters, contention resolution and collision avoidance is your responsibility and may easily be addressed through ladder logic programming. **Failure to observe this precaution can result in injury or equipment damage.**

Top Input

The top input begins an XMIT operation and it should remain ON until the operation has completed successfully or an error has occurred.

Middle Input

The middle input aborts any active XMIT operation and forces the port to slave mode. An abort code (121) is placed into the fault status register. The port remains closed as long as this input is ON.

Note: To reset an XMIT fault and clear the fault register, the top input must go OFF for at least one PLC scan.

Port # (Top Node)

In the top node you select the PLC port number, from where the messages are sent or received.
The top node must contain one of the following constants:

- #0001 = PLC port #1
- #0002 = PLC port #2

Control Block (Middle Node)

The 4x register entered in the middle node is the first of sixteen contiguous 4x registers that comprise the control block:

Register	Content
Displayed	Current revision number of XMIT block.
First implied	*Fault Status, p. 730*
Second implied	Available to user May be used as pointers for instructions like TBLK.
Third implied	Data Rate: 50, 75, 110, 134, 150, 300, 600, 1200, 2400, 9600 and 19200
Fourth implied	Data Bits: 7 for ASCII mode 8 for RTU mode
Fifth implied	Parity: 0 = no parity 1 = odd parity 2 = even parity
Sixth implied	Stop Bits 1 = one stop bit 2 = two stop bits
Seventh implied	Available to user May be used as pointers for instructions like TBLK.
Eighth implied	*Command Word, p. 732* 0000-0001-0000-0000 (256Dec)
Ninth implied	Pointer to message table (See *Message Pointer, p. 734*) Limited by the range of 4x registers configured
10th implied	Length of message Range: 0...512

Register	Content
11th implied	Response Time Out (ms) Range: 0 ... 65535
12th implied	Retry limit Range: 0 ... 65535
13th implied	Start of transmission delay (ms) Range: 0 ... 65535
14th implied	End of transmission delay (ms) Range: 0 ... 65535
15th implied	Current number of retry attempts made by the instruction

WARNING

No modification of the control block address

Do **not** modify the address in the middle node of the XMIT block or delete it from the program while it is active. This locks up the port preventing communications.

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

Fault Status

The following fault code is generated by the XMIT instruction:

Fault Code	Fault Description
1	Modbus exception Illegal function
2	Modbus exception Illegal data address
3	Modbus exception Illegal data value
4	Modbus exception Slave device failure
5	Modbus exception Acknowledge
6	Modbus exception Slave device busy
7	Modbus exception Negative acknowledge
8	Modbus exception Memory parity error
9 ... 99	Reserved
100	Slave PLC data area cannot equal zero
101	Master PLC data area cannot equal zero
102	Coil (0x) not configured
103	Holding register (4x) not configured
104	Data length cannot equal zero

Fault Code	Fault Description
105	Pointer to message table cannot equal zero
106	Pointer to message table is outside the range of configured holding registers (4x)
107	Transmit message timeout This error is generated when the UART cannot complete a transmission in 10 seconds or less. This error bypasses the retry counter and will activate the error output on the first error.
108	Undefined error
109	Modem returned ERROR
110	Modem returned NO CARRIER
111	Modem returned NO DIALTONE
112	Modem returned BUSY
113	Invalid LRC checksum from the slave PLC
114	Invalid CRC checksum from the slave PLC
115	Invalid Modbus function code
116	Modbus response message timeout
117	Modem reply timeout
118	XMIT could not gain access to PLC communications port #1 or port #2
119	XMIT could not enable PLC port receiver
120	XMIT could not set PLC UART
121	User issued an abort command
122	Top node of XMIT not equal to zero, one or two
123	Bottom node of XMIT is not equal to seven, eight or sixteen
124	Undefined internal state
125	Broadcast mode not allowed with this Modbus function code
126	DCE did not assert CTS
127	Illegal configuration (data rate, data bits, parity, or stop bits)
128	Unexpected response received from Modbus slave
129	Illegal command word setting
130	Command word changed while active
131	Invalid character count
132	Invalid register block
133	ASCII input FIFO overflow error
134	Invalid number of start characters or termination characters

Command Word Command Word

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

BIT	Function	Definition
Bit 1 (msb)		Reserved
Bit 2	Enable RTS/ CTS modem control	Set to 1 when a DCE connected to the PLC requires hardware handshaking using RTS/CTS control. This bit may be used in conjunction with values contained in (4x + 13) and (4x + 14). Start of transmission delay (4x + 13) keeps RTS asserted for (X ms) before XMIT sends message out of PLC port #1. Likewise, end of transmission delay (4x + 14) keeps RTS asserted for (X ms) after XMIT has finished sending a message out of the PLC port #1. Once the end of transmission delay expires XMIT de-assert RTS.
Bit 3	Enable RS485 mode	Set to 1 when the selected port should operate in RS485 mode. Otherwise it defaults to 0, which is RS232 mode.
Bit 4		Reserved
Bit 5	Terminated ASCII input	Set to 1 to remove and discard all characters from FIFO until the starting string is matched, then these starting characters and subsequent characters are written into a contiguous 4x register destination block until the terminator sequence is matched. The terminator string is also written into the 4x register destination block.
Bit 6	Simple ASCII input	Set to 1 to remove the ASCII characters from FIFO for writing into a contiguous 4x register block. The Message pointer (4x + 9) specifics the 4x register block.
Bit 7	Enable ASCII string messaging	Set to 1 when you want to send ASCII messages out of the PLC. XMIT sends ASCII strings up to 1024 characters in length. You program each ASCII message into contiguous 4x registers of the PLC. Two characters allowed per register. Only use Bit 7 OR Bit 8, do not try to use both.
Bit 8	Enable Modbus messaging	Set to 1 when you want to send Modbus messages out of the PLC. Modbus messages may be in either RTU or ASCII formats. When data bits=8, XMIT uses Modbus RTU format. When data bits=7, XMIT uses Modbus ASCII format. Only use Bit 7 OR Bit 8, do not try to use both.
Bit 9	Enable ASCII receive FIFO	Set to 1 to allow the XMIT block to take control over the selected port (1 or 2) from the PLC. The block begins to receive ASCII characters into an empty 512 byte circular FIFO.
Bit 10	Enable back space	Set to 1 to allow special handling of ASCII back space character (BS, 8Hex). When using either Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 5) each back space character is removed from FIFO and may or may NOT be stored into a 4x register destination block.

BIT	Function	Definition
Bit 11	Enable RTS/ CTS flow control	Set to 1 to allow full duplex hardware flow control using the RTS and CTS handshaking signals for ASCII massaging. The RTS/CTS operates in both the input and output modes.
Bit 12	Enable Xon/ Xoff flow control	Set to 1 to allow full duplex software flow control using the ASCII Xon character (DC1, 11 Hex) and the ASCII Xoff character (DC3, 13 Hex). The Xon/Xoff operates in both the input and output modes.
Bit 13	Pulse dial modem	Set to 1 when using a Hayes compatible dial-up modem and you wish to pulse dial a telephone number. You program the phone number into contiguous 4x registers of the PLC. A pointer to these registers must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Pulse dialed numbers are sent to the modem automatically preceded by ATDP and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 14	Disconnect modem	Set to 1 when using a Hayes compatible dial-up modem and you want to disconnect the modem. You must use ladder logic to turn this bit ON. Since the disconnect message is an ASCII string, bit 7 must be ON prior to sending the message. disconnect messages are sent to the modem automatically preceded by +++AT and with carriage return <CR> and line feed <LF> appended. XMIT looks for a correct disconnect response from the modem before it turns ON the bottom output, noting a successful completion.
Bit 15	Tone dial modem	Set to 1 when using a Hayes compatible dial-up modem and you wish to tone dial a telephone number. You program the dial message into contiguous 4x registers of the PLC. A pointer to the dial message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Tone dial numbers are sent to the modem automatically preceded by ATDT and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 16	Initialize modem	Set to 1 when using a Hayes compatible dial-up modem and you want to initialize the modem. You program the initialization message into contiguous 4x registers of the PLC. A pointer to the initialization message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). All messages are sent to the modem automatically preceded by AT and with a carriage return <CR> and line feed <LF> appended. Since the initialization message is an ASCII string, bit 7 must be ON prior to sending the message.

Detailed information about the bits of the command word you will find in the *Modicon XMIT Function Block User Guide*.

Message Pointer

You enter a pointer that points to the beginning of the message table. There are two different handlings of the pointer depending on using ASCII character strings or Modbus messages.

For **ASCII character strings**, the pointer is the register offset to the first register of the ASCII character string. Each register holds up to two ASCII characters. Each ASCII string may be up to 1024 characters in length. For example, when you want to send 10 ASCII messages out of the PLC, you must program 10 ASCII character strings into 4x registers of the PLC and then through ladder logic set the pointer to the start of each message after each successful operation of XMIT.
For **Modbus messages**, the pointer is the register offset to the first register of the Modbus definition table. The Modbus definition table has different length depending on the used Modbus function code and you must program it for successful XMIT operation.
Detailed information about the bits of the command word you will find in the *Modicon XMIT Function Block User Guide*.

Outputs

	CAUTION
	All three outputs of the instruction turn on, regardless of the input states When the NSUP loadable is not installed or is installed after the XMIT loadable or is installed in a Quantum PLC with an executive < V2.0, all three outputs turn on, regardless of the input states. **Failure to observe this precaution can result in injury or equipment damage.**

XMRD: Extended Memory Read

147

At a Glance

Introduction

This chapter describes the instruction XMRD.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	736
Representation	736
Parameter Description	737

Short Description

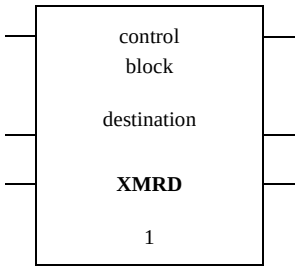
Function Description

The XMRD instruction is used to copy a table of 6x extended memory registers to a table of 4x holding registers in state RAM.

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 = activates read operation
Middle input	0x, 1x	None	OFF = clears offset to 0 ON = does not clear offset
Bottom input	0x, 1x	None	OFF = abort on error ON = do not abort on error
control block (top node)	4x	INT, UINT, WORD	First of six contiguous holding registers in the extended memory
destination (middle node)	4x	INT, UINT, WORD	The first 4x holding register in a table of registers that receive the transferred data from the 6x extended memory storage registers
1 (bottom node)		INT, UINT	Contains the constant value 1, which cannot be changed
Top output	0x	None	Read transfer active
Middle output	0x	None	Error condition detected
Bottom output	0x	None	ON = operation complete

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of six contiguous holding registers in the extended memory control block.

Reference	Register Name	Description
Displayed	status word	Contains the diagnostic information about extended memory (see *Status Word of the Control Block, p. 738*)
First implied	file number	Specifies which of the extended memory files is currently in use (range: 1 ... 10)
Second implied	start address	Specifies which 6x storage register in the current file is the starting address; 0 = 60000, 9999 = 69999
Third implied	count	Specifies the number of registers to be read or written in a scan when the appropriate function block is powered; range: 0 ... 9999, not to exceed number specified in max registers (fifth implied)
Fourth implied	offset	Keeps a running total of the number of registers transferred thus far
Fifth implied	max registers	Specifies the maximum number of registers that may be transferred when the function block is powered (range: 0 ... 9999)

If you are in multi-scan mode, these six registers should be unique to this function block.

Status Word of the Control Block

Status Word of the Control Block

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

Bit	Function
1	1 = power-up diagnostic error
2	1 = parity error in extended memory
3	1 = extended memory does not exist
4	0 = transfer not running 1 = busy
5	0 = transfer in progress 1 = transfer complete
6	1 = file boundary crossed
7	1 = offset parameter too large
8 - 9	Not used
10	1 = nonexistent state RAM
11	Not used
12	1 = maximum registers parameter error
13	1 = offset parameter error
14	1 = count parameter error
15	1 = starting address parameter error
16	1 = file number parameter error

XMWT: Extended Memory Write

148

At a Glance

Introduction

This chapter describes the instruction XMWT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Short Description	740
Representation	740
Parameter Description	741

Short Description

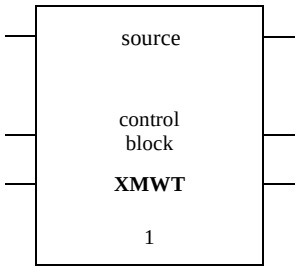
Function Description

The XMWT instruction is used to write data from a block of input registers or holding registers in state RAM to a block of 6x registers in an extended memory file.

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 = activates write operation
Middle input	0x, 1x	None	OFF = clears offset to 0 ON = does not clear offset
Bottom input	0x, 1x	None	OFF = abort on error ON = do not abort on error
source (top node)	3x, 4x	INT, UINT, WORD	The first 3x or 4x register in a block of contiguous source registers, i.e. input or holding registers, whose contents will be written to 6x extended memory registers
control block (middle node)	4x	INT, UINT, WORD	First of six contiguous holding registers in the extended memory
1 (bottom node)		INT, UINT	Contains the constant value 1, which cannot be changed
Top output	0x	None	Write transfer active
Middle output	0x	None	Error condition detected
Bottom output	0x	None	ON = operation complete

Parameter Description

Control Block (Top Node)

The 4x register entered in the middle node is the first of six contiguous holding registers in the extended memory control block.

Reference	Register Name	Description
Displayed	status word	Contains the diagnostic information about extended memory (see *Status Word of the Control Block, p. 742*)
First implied	file number	Specifies which of the extended memory files is currently in use (range: 1 ... 10)
Second implied	start address	Specifies which 6x storage register in the current file is the starting address; 0 = 60000, 9999 = 69999
Third implied	count	Specifies the number of registers to be read or written in a scan when the appropriate function block is powered; range: 0 ... 9999, not to exceed number specified in max registers (fifth implied)
Fourth implied	offset	Keeps a running total of the number of registers transferred thus far
Fifth implied	max registers	Specifies the maximum number of registers that may be transferred when the function block is powered (range: 0 ... 9999)

If you are in multi-scan mode, these six registers should be unique to this function block.

Status Word of the Control Block

Status Word of the Control Block

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

Bit	Function
1	1 = power-up diagnostic error
2	1 = parity error in extended memory
3	1 = extended memory does not exist
4	0 = transfer not running 1 = busy
5	0 = transfer in progress 1 = transfer complete
6	1 = file boundary crossed
7	1 = offset parameter too large
8 - 9	Not used
10	1 = nonexistent state RAM
11	Not used
12	1 = maximum registers parameter error
13	1 = offset parameter error
14	1 = count parameter error
15	1 = starting address parameter error
16	1 = file number parameter error

XOR: Exclusive OR

149

At a Glance

Introduction

This chapter describes the instruction XOR.

What's in this chapter?

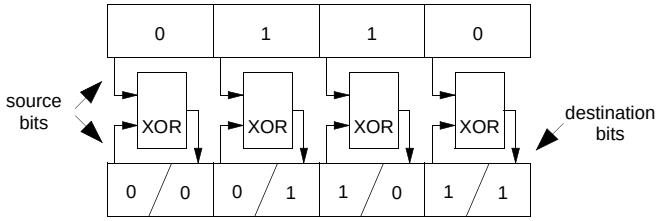
This chapter contains the following topics:

Topic	Page
Short Description	744
Representation	745
Parameter Description	745

Short Description

Function Description

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

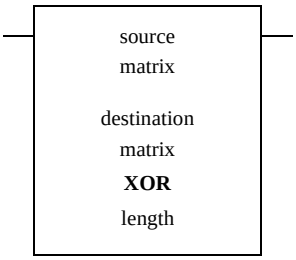


STOP	**WARNING**
	XOR will override any disabled coils within the destination matrix without enabling them. This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's state can be changed by the XOR operation. **Failure to observe this precaution can result in severe injury or equipment damage.**

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

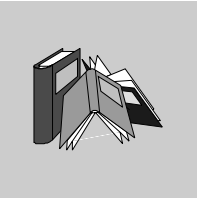
Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Initiates XOR
source matrix (top node)	0x, 1x, 3x, 4x	BOOL, WORD	First reference in the source matrix
destination matrix (middle node)	0x, 4x	BOOL, WORD	First reference in the destination matrix
length (bottom node)		INT, UINT	Matrix 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 XORed.

Glossary



A

active window The window, which is currently selected. Only one window can be active at any one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.

Actual parameter Currently connected Input/Output parameters.

Addresses (Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules.
The display/input of direct addresses is possible in the following formats:
- Standard format (400001)
- Separator format (4:00001)
- Compact format (4:1)
- IEC format (QW1)

ANL_IN ANL_IN stands for the data type "Analog Input" and is used for processing analog values. The 3x References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANL_OUT ANL_OUT stands for the data type "Analog Output" and is used for processing analog values. The 4x-References of the configured analog output module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANY In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

ANY_BIT In the existing version, "ANY_BIT" covers the data types BOOL, BYTE and WORD.

ANY_ELEM In the existing version "ANY_ELEM" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD.

ANY_INT In the existing version, "ANY_INT" covers the data types DINT, INT, UDINT and UINT.

ANY_NUM In the existing version, "ANY_NUM" covers the data types DINT, INT, REAL, UDINT and UINT.

ANY_REAL In the existing version "ANY_REAL" covers the data type REAL.

Application window The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII mode American Standard Code for Information Interchange. The ASCII mode is used for communication with various host devices. ASCII works with 7 data bits.

Atrium The PC based controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB) The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (**Objects** → **Source**). If a back up file can be assigned, the name of the source file can be given.

Base 16 literals Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 8 literal Base 8 literals function as the input of whole number values in the octal system. The base must be denoted by the prefix 3.63kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
8#3_1111 or 8#377 (decimal 255)
8#34_1111 or 8#340 (decimal 224)

Basis 2 literals Base 2 literals function as the input of whole number values in the dual system. The base must be denoted by the prefix 0.91kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_1111 or 2#11100000 (decimal 224)

Binary connections Connections between outputs and inputs of FFBs of data type BOOL.

Bit sequence A data element, which is made up from one or more bits.

BOOL BOOL stands for the data type "Boolean". The length of the data elements is 1 bit (in the memory contained in 1 byte). The range of values for variables of this type is 0 (FALSE) and 1 (TRUE).

Bridge A bridge serves to connect networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not deployed via bridges.

BYTE BYTE stands for the data type "Bit sequence 8". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 8 bit. A numerical range of values cannot be assigned to this data type.

C

Cache The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.

Call up The operation, by which the execution of an operation is initiated.

Coil A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.

Compact format (4:1) The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.

Connection A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.

Constants Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).

Contact A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

Data transfer settings

Settings, which determine how information from the programming device is transferred to the PLC.

Data types

The overview shows the hierarchy of data types, as they are used with inputs and outputs of Functions and Function blocks. Generic data types are denoted by the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System data types (IEC extensions)
- Derived (from "ANY" data types)

DCP I/O station

With a Distributed Control Processor (D908) a remote network can be set up with a parent PLC. When using a D908 with remote PLC, the parent PLC views the remote PLC as a remote I/O station. The D908 and the remote PLC communicate via the system bus, which results in high performance, with minimum effect on the cycle time. The data exchange between the D908 and the parent PLC takes place at 1.5 Megabits per second via the remote I/O bus. A parent PLC can support up to 31 (Address 2-32) D908 processors.

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs under Windows. The DDE interface can be used in the extended monitor to call up its own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data onto the PLC via the server. Data can therefore be altered directly in the PLC, while it monitors and analyzes the results. When using this interface, the user is able to make their own "Graphic-Tool", "Face Plate" or "Tuning Tool", and integrate this into the system. The tools can be written in any DDE supporting language, e.g. Visual Basic and Visual-C++. The tools are called up, when the one of the buttons in the dialog box extended monitor uses Concept Graphic Tool: Signals of a projection can be displayed as timing diagrams via the DDE connection between Concept and Concept Graphic Tool.

Decentral Network (DIO)	A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.
Declaration	Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.
Definition data file (Concept EFB)	The definition file contains general descriptive information about the selected FFB and its formal parameters.
Derived data type	Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept. Distinctions are made between global data types and local data types.
Derived Function Block (DFB)	A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol. The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version. Distinctions are made between local and global DFBs.
DINT	DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from –2 exp (31) to 2 exp (31) –1.
Direct display	A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.
Document window	A window within an Application window. Several document windows can be opened at the same time in an application window. However, only one document window can be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.
Dummy	An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.

DX Zoom This property enables connection to a programming object to observe and, if necessary, change its data value.

E

Elementary functions/ function blocks (EFB) Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.

EN / ENO (Enable / Error display) If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands **Objects** → **Properties...** or via a double click on the FFB.

Error When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output is set to "0".

Evaluation The process, by which a value for a Function or for the outputs of a Function block during the Program execution is transmitted.

Expression Expressions consist of operators and operands.

F

FFB (functions/ function blocks) Collective term for EFB (elementary functions/function blocks) and DFB (derived function blocks)

Field variables Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.

FIR filter Finite Impulse Response Filter

Formal parameters Input/Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC) A Program organization unit, which exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values. Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.

Function block (item) (FB) A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s).
Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places.
The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described in the corresponding definitions.

Function block dialog (FBD) One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type A language element, consisting of: 1. the definition of a data structure, subdivided into input, output and internal variables, 2. A set of operations, which is used with the elements of the data structure, when a function block type instance is called up. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (called up) several times.

Function counter The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m

n = Section number (number running)
m = Number of the FFB object in the section (number running)

G

Generic data type A Data type, which stands in for several other data types.

Generic literal If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.

Global derived data types Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global DFBs Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global macros Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Groups (EFBs) Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBs, especially in extensive libraries.

I

I/O component list
The I/O and expert assemblies of the various CPUs are configured in the I/O component list.

IEC 61131-3
International norm: Programmable controllers – part 3: Programming languages.

IEC format (QW1)
In the place of the address stands an IEC identifier, followed by a five figure address:
- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier)
An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö,ü, é, õ) can be used, taken from project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively.
Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers are not permitted to be Key words.

IIR filter
Infinite Impulse Response Filter

Initial step (starting step)
The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.

Initial value
The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.

Input bits (1x references)
The 1/0 status of input bits is controlled via the process data, which reaches the CPU from an entry device.

> **Note:** The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.

Input parameters (Input)
When calling up a FFB the associated Argument is transferred.

Input words (3x references)
An input word contains information, which come from an external source and are represented by a 16 bit figure. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.

Instantiation
The generation of an Item.

Instruction (IL)
Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.

Instruction (LL984)
When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.

Instruction list (IL)
IL is a text language according to IEC 1131, in which operations, e.g. conditional/ unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.

INT
INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from –2 exp (15) to 2 exp (15) –1.

Integer literals
Integer literals function as the input of whole number values in the decimal system. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP)
To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.

Item name

An Identifier, which belongs to a certain Function block item. The item name serves as a unique identifier for the function block in a program organization unit. The item name is automatically generated, but can be edited. The item name must be unique throughout the Program organization unit, and no distinction is made between upper/lower case. If the given name already exists, a warning is given and another name must be selected. The item name must conform to the IEC name conventions, otherwise an error message appears. The automatically generated instance name always has the structure: FBI_n_m

FBI = Function block item
n = Section number (number running)
m = Number of the FFB object in the section (number running)

J

Jump

Element of the SFC language. Jumps are used to jump over areas of the chain.

K

Key words

Key words are unique combinations of figures, which are used as special syntactic elements, as is defined in appendix B of the IEC 1131-3. All key words, which are used in the IEC 1131-3 and in Concept, are listed in appendix C of the IEC 1131-3. These listed keywords cannot be used for any other purpose, i.e. not as variable names, section names, item names etc.

L

Ladder Diagram (LD)

Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.

Ladder Logic 984 (LL)

In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols),which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant.
The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance.
In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.

Landscape format

Landscape format means that the page is wider than it is long when looking at the printed text.

Language element

Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.

Library

Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries.
EFB libraries can be subdivided into Groups.

Literals

Literals serve to directly supply values to inputs of FFBs, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated.
Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears as Base 2 literal, Base 8 literal, Base 16 literal, Integer literal, Real literal or Real literal with exponent.

Local derived data types

Local derived data types are only available in a single Concept project and its local DFBs and are contained in the DFB directory under the project directory.

Local DFBs Local DFBs are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local link The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.

Local macros Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local network nodes The local node is the one, which is projected evenly.

Located variable Located variables are assigned a state RAM address (reference addresses 0x,1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses.

Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro

Macros are created with help from the software Concept DFB.
Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration).
Distinctions are made between local and global macros.

Macros have the following properties:
- Macros can only be created in the programming languages FBD and LD.
- Macros only contain one single section.
- Macros can contain any complex section.
- From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro.
- Calling up DFBs in a macro
- Variable declaration
- Use of macro-own data structures
- Automatic acceptance of the variables declared in the macro
- Initial value for variables
- Multiple instancing of a macro in the whole program with different variables
- The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).

MMI

Man Machine Interface

Multi element variables

Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY.
Distinctions are made between Field variables and structured variables.

N

Network

A network is the connection of devices to a common data path, which communicate with each other via a common protocol.

Network node

A node is a device with an address (164) on the Modbus Plus network.

Node address The node address serves a unique identifier for the network in the routing path. The address is set directly on the node, e.g. with a rotary switch on the back of the module.

O

Operand An operand is a Literal, a Variable, a Function call up or an Expression.

Operator An operator is a symbol for an arithmetic or Boolean operation to be executed.

Output parameters (Output) A parameter, with which the result(s) of the Evaluation of a FFB are returned.

Output/discretes (0x references) An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of the State RAM.

Output/marker words (4x references) An output/marker word can be used to save numerical data (binary or decimal) in the State RAM, or also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.

PLC Programmable controller

Program The uppermost Program organization unit. A program is closed and loaded onto a single PLC.

Program cycle A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit	A Function, a Function block, or a Program. This term can refer to either a Type or an Item.
Programming device	Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.
Programming redundancy system (Hot Standby)	A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.
Project	General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation. General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.
Project data bank	The data bank in the Programming device, which contains the projection information for a Project.
Prototype data file (Concept EFB)	The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal

R

REAL

REAL stands for the data type "real". The input appears as Real literal or as Real literal with exponent. The length of the data element is 32 bit. The value range for variables of this data type reaches from 8.43E-37 to 3.36E+38.

> **Note:** Depending on the mathematic processor type of the CPU, various areas within this valid value range cannot be represented. This is valid for values nearing ZERO and for values nearing INFINITY. In these cases, a number value is not shown in animation, instead NAN (**N**ot **A** **N**umber) oder INF (**INF**inite).

Real literal

Real literals function as the input of real values in the decimal system. Real literals are denoted by the input of the decimal point. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent

Real literals with exponent function as the input of real values in the decimal system. Real literals with exponent are denoted by the input of the decimal point. The exponent sets the key potency, by which the preceding number is multiplied to get to the value to be displayed. The basis may be preceded by a negative sign (-). The exponent may be preceded by a positive or negative sign (+/-). Single underline signs (_) between figures are not significant. (Only between numbers, not before or after the decimal poiont and not before or after "E", "E+" or "E-")

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference

Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM.
0x area = Discrete outputs
1x area = Input bits
3x area = Input words
4x area = Output bits/Marker words
6x area = Register in the extended memory

Note: The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

Register in the extended memory (6x reference)

6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.

RIO (Remote I/O)

Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.

RP (PROFIBUS)

RP = Remote Peripheral

RTU mode

Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Rum-time error

Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.

S

SA85 module The SA85 module is a Modbus Plus adapter for an IBM-AT or compatible computer.

Section A section can be used, for example, to describe the functioning method of a technological unit, such as a motor.
A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section.
Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.

Separator format (4:00001) The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).

Sequence language (SFC) The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.

Serial ports With serial ports (COM) the information is transferred bit by bit.

Source code data file (Concept EFB) The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.

Standard format (400001) The five figure address is located directly after the first figure (the reference).

Standardized literals If the data type for the literal is to be automatically determined, use the following construction: 'Data type name'#'Literal value'.

Example
INT#15 (Data type: Integer, value: 15),
BYTE#00001111 (data type: Byte, value: 00001111)
REAL#23.0 (Data type: Real, value: 23.0)

For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretes, input words, and discrete words are located in the state RAM.

Statement (ST) Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.

Status bits There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.

Step SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.

Step name The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears.
The automatically generated step name always has the structure: S_n_m

S = Step
n = Section number (number running)
m = Number of steps in the section (number running)

Structured text (ST) ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.

Structured variables Variables, one of which is assigned a Derived data type defined with STRUCT (structure).
A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are performed when highlighting and including in the I/O population of the Concept configuration.

Symbol (Icon) Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB)

The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.

TIME

TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time span literals

Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted.

Example
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

The network "Token" controls the temporary property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.

Traffic Cop

The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.

Transition

The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB
User defined elementary functions/function blocks
Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.

UDINT
UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1.

UINT
UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to (2exp16)-1.

Unlocated variable
Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.

V

Variables
Variables function as a data exchange within sections between several sections and between the Program and the PLC.
Variables consist of at least a variable name and a Data type.
Should a variable be assigned a direct Address (Reference), it is referred to as a Located variable. Should a variable not be assigned a direct address, it is referred to as an unlocated variable. If the variable is assigned a Derived data type, it is referred to as a Multi-element variable.
Otherwise there are Constants and Literals.

Vertical format
Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output remains at "1".

WORD WORD stands for the data type "Bit sequence 16". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 16 bit. A numerical range of values cannot be assigned to this data type.

Index

A

AD16, 67
ADD, 69
Add 16 Bit, 67
Addition, 69
    AD16, 67
    ADD, 69
Advanced Calculations, 484
Analog Input, 491
Analog Output, 501
Analog Values, 27
AND, 71
ASCII Functions
    READ, 627
    WRIT, 719
Average Weighted Inputs Calculate, 505

B

Base 10 Antilogarithm, 151
Base 10 Logarithm, 227
BCD, 75
Binary to Binary Code, 75
Bit Control, 469
Bit pattern comparison
    CMPR, 101
Bit Rotate, 87
BLKM, 77
BLKT, 81
Block Move, 77
Block Move with Interrupts Disabled, 85
Block to Table, 81
BMDI, 85
BROT, 87

C

Calculated preset formula, 511
Central Alarm Handler, 497
Changing the Sign of a Floating Point Number, 167
Check Sum, 97
CHS, 91
CKSM, 97
Closed Loop Control, 27
CMPR, 101
Coils, 55
Communications
    MSTR, 423
COMP, 105
Compare Register, 101
Complement a Matrix, 105
Comprehensive ISA Non Interacting PID, 531
Configure Hot Standby, 91
Contacts, 55
Convertion
    BCD to binary, 75
    binary to BCD, 75

Counters / Timers
T.01 Timer, 699
T0.1 Timer, 701
T1.0 Timer, 703
T1MS Timer, 705
UCTR, 717
Counters/Timers
DCTR, 109

D

Data Logging for PCMCIA Read/Write Support, 119
DCTR, 109
Derivative Rate Calculation over a Specified Time, 579
DIOH, 111
Distributed I/O Health, 111
DIV, 115
Divide, 115
Divide 16 Bit, 129
DLOG, 119
Double Precision Addition, 139
Double Precision Division, 199
Double Precision Multiplication, 235
Double Precision Subtraction, 271
Down Counter, 109
DRUM, 125
DRUM Sequencer, 125
DV16, 129

E

EMTH, 133
EMTH Subfunction
EMTH-ADDDP, 139
EMTH-ADDFP, 143, 147
EMTH-ANLOG, 151
EMTH-ARCOS, 155
EMTH-ARSIN, 159
EMTH-ARTAN, 163
EMTH-CHSIN, 167
EMTH-CMPFP, 171
EMTH-CMPIF, 175
EMTH-CNVDR, 179
EMTH-CNVFI, 183
EMTH-CNVIF, 187
EMTH-CNVRD, 191
EMTH-COS, 195
EMTH-DIVDP, 199
EMTH-DIVFI, 203
EMTH-DIVFP, 207
EMTH-DIVIF, 211
EMTH-ERLOG, 215
EMTH-EXP, 219
EMTH-LNFP, 223
EMTH-LOG, 227
EMTH-LOGFP, 231
EMTH-MULDP, 235
EMTH-MULFP, 239
EMTH-MULIF, 243
EMTH-PI, 247
EMTH-POW, 251
EMTH-SINE, 255
EMTH-SQRFP, 259
EMTH-SQRT, 263
EMTH-SQRTP, 267
EMTH-SUBDP, 271
EMTH-SUBFI, 275
EMTH-SUBFP, 279
EMTH-SUBIF, 283
EMTH-TAN, 287
EMTH-ADDDP, 139
EMTH-ADDFP, 143
EMTH-ADDIF, 147
EMTH-ANLOG, 151
EMTH-ARCOS, 155
EMTH-ARSIN, 159
EMTH-ARTAN, 163
EMTH-CHSIN, 167
EMTH-CMPFP, 171
EMTH-CMPIF, 175
EMTH-CNVDR, 179
EMTH-CNVFI, 183
EMTH-CNVIF, 187
EMTH-CNVRD, 191
EMTH-COS, 195
EMTH-DIVDP, 199
EMTH-DIVFI, 203
EMTH-DIVFP, 207
EMTH-DIVIF, 211

EMTH-ERLOG, 215
EMTH-EXP, 219
EMTH-LNFP, 223
EMTH-LOG, 227
EMTH-LOGFP, 231
EMTHMULDP, 235
EMTH-MULFP, 239
EMTH-MULIF, 243
EMTH-PI, 247
EMTH-POW, 251
EMTH-SINE, 255
EMTH-SQRFP, 259
EMTH-SQRT, 263
EMTH-SQRTP, 267
EMTH-SUBDP, 271
EMTH-SUBFI, 275
EMTH-SUBFP, 279
EMTH-SUBIF, 283
EMTH-TAN, 287
Engineering Unit Conversion and Alarms, 309
ESI, 291
EUCA, 309
Exclusive OR, 743
Extended Math, 133
Extended Memory Read, 735
Extended Memory Write, 739

F

Fast I/O Instructions
  BMDI, 85
  ID, 355
  IE, 359
  IMIO, 363
  IMOD, 369
  ITMR, 377
FIN, 321
First In, 321
First Out, 325
First-order Lead/Lag Filter, 549
Floating Point - Integer Subtraction, 275
Floating Point Addition, 143
Floating Point Arc Cosine of an Angle (in Radians), 155
Floating Point Arc Tangent of an Angle (in Radians), 163
Floating Point Arcsine of an Angle (in Radians), 159
Floating Point Common Logarithm, 231
Floating Point Comparison, 171
Floating Point Conversion of Degrees to Radians, 179
Floating Point Conversion of Radians to Degrees, 191
Floating Point Cosine of an Angle (in Radians), 195
Floating Point Divided by Integer, 203
Floating Point Division, 207
Floating Point Error Report Log, 215
Floating Point Exponential Function, 219
Floating Point Multiplication, 239
Floating Point Natural Logarithm, 223
Floating Point Sine of an Angle (in Radians), 255
Floating Point Square Root, 259, 263
Floating Point Subtraction, 279
Floating Point Tangent of an Angle (in Radians), 287
Floating Point to Integer, 329
Floating Point to Integer Conversion, 183
Formatted Equation Calculator, 521
Formatting Messages, 41
Four Station Ratio Controller, 583
FOUT, 325
FTOI, 329

H

History and Status Matrices, 331
HLTH, 331
Hot standby
  CHS, 91

I

IBKR, 345
IBKW, 347
ICMP, 349
ID, 355
IE, 359
IMIO, 363

Immediate I/O, 363
IMOD, 369
Indirect Block Read, 345
Indirect Block Write, 347
Input Compare, 349
Input Selection, 593
Installation of DX Loadables, 53
Instruction
  Coils, Contacts and Interconnects, 55
Instruction Groups, 17
  ASCII Communication Instructions, 19
  Coils, Contacts and Interconnects, 26
  Counters and Timers Instructions, 19
  Fast I/O Instructions, 20
  Loadable DX, 21
  Math Instructions, 21
  Matrix Instructions, 23
  Miscellaneous, 24
  Move Instructions, 25
  Overview, 18
  Skips/Specials, 25
  Special Instructions, 26
Integer - Floating Point Subtraction, 283
Integer + Floating Point Addition, 147
Integer Divided by Floating Point, 211
Integer to Floating Point, 383
Integer x Floating Point Multiplication, 243
Integer-Floating Point Comparison, 175
Integer-to-Floating Point Conversion, 187
Integrate Input at Specified Interval, 527
Interconnects, 55
Interrupt Disable, 355
Interrupt Enable, 359
Interrupt Handling, 49
Interrupt Module Instruction, 369
Interrupt Timer, 377
ISA Non Interacting PI, 563
ITMR, 377
ITOF, 383

J

JSR, 385
Jump to Subroutine, 385

L

LAB, 387
Label for a Subroutine, 387
Limiter for the Pv, 537
LL984
  AD16, 67
  ADD, 69
  AND, 71
  BCD, 75
  BLKM, 77
  BLKT, 81
  BMDI, 85
  BROT, 87
  CHS, 91
  CKSM, 97
  Closed Loop Control / Analog Values, 27
  CMPR, 101
  Coils, Contacts and Interconnects, 55
  COMP, 105
  DCTR, 109
  DIOH, 111
  DIV, 115
  DLOG, 119
  DRUM, 125
  DV16, 129
  EMTH, 133
  EMTH-ADDDP, 139
  EMTH-ADDFP, 143
  EMTH-ADDIF, 147
  EMTH-ANLOG, 151
  EMTH-ARCOS, 155
  EMTH-ARSIN, 159
  EMTH-ARTAN, 163
  EMTH-CHSIN, 167
  EMTH-CMPFP, 171
  EMTH-CMPIF, 175
  EMTH-CNVDR, 179
  EMTH-CNVFI, 183
  EMTH-CNVIF, 187
  EMTH-CNVRD, 191
  EMTH-COS, 195
  EMTH-DIVDP, 199
  EMTH-DIVFI, 203
  EMTH-DIVFP, 207
  EMTH-DIVIF, 211

EMTH-ERLOG, 215
EMTH-EXP, 219
EMTH-LNFP, 223
EMTH-LOG, 227
EMTH-LOGFP, 231
EMTH-MULDP, 235
EMTH-MULFP, 239
EMTH-MULIF, 243
EMTH-PI, 247
EMTH-POW, 251
EMTH-SINE, 255
EMTH-SQRFP, 259
EMTH-SQRT, 263
EMTH-SQRTP, 267
EMTH-SUBDP, 271
EMTH-SUBFI, 275
EMTH-SUBFP, 279
EMTH-SUBIF, 283
EMTH-TAN, 287
ESI, 291
EUCA, 309
FIN, 321
Formatting Messages for ASCII READ/ WRIT Operations, 41
FOUT, 325
FTOI, 329
HLTH, 331
IBKR, 345
IBKW, 347
ICMP, 349
ID, 355
IE, 359
IMIO, 363
IMOD, 369
Interrupt Handling, 49
ITMR, 377
ITOF, 383
JSR, 385
LAB, 387
LOAD, 391
MAP 3, 395
MBIT, 403
MBUS, 407
MRTM, 417
MSTR, 423
MU16, 465
MUL, 467
NBIT, 469
NCBT, 471
NOBT, 473
NOL, 475
OR, 479
PCFL, 483
PCFL-AIN, 491
PCFL-ALARM, 497
PCFL-AOUT, 501
PCFL-AVER, 505
PCFL-CALC, 511
PCFL-DELAY, 515
PCFL-EQN, 521
PCFL-INTEG, 527
PCFL-KPID, 531
PCFL-LIMIT, 537
PCFL-LIMV, 541
PCFL-LKUP, 545
PCFL-LLAG, 549
PCFL-MODE, 553
PCFL-ONOFF, 557
PCFL-PI, 563
PCFL-PID, 567
PCFL-RAMP, 573
PCFL-RATE, 579
PCFL-RATIO, 583
PCFL-RMPLN, 589
PCFL-SEL, 593
PCFL-TOTAL, 597
PEER, 603
PID2, 607
R --> T, 621
RBIT, 625
READ, 627
RET, 633
SAVE, 635
SBIT, 639
SCIF, 641
SENS, 647
SKPC, 651
SKPR, 655
SRCH, 659
STAT, 663
SU16, 687
SUB, 689

Subroutine Handling, 51
T.01 Timer, 699
T-->R, 691
T-->T, 695
T0.1 Timer, 701
T1.0 Timer, 703
T1MS Timer, 705
TBLK, 711
TEST, 715
UCTR, 717
WRIT, 719
XMIT, 725
XMRD, 735
XMWT, 739
XOR, 743
LOAD, 391
Load Flash, 391
Load the Floating Point Value of "Pi", 247
Loadable DX
CHS, 91
DRUM, 125
ESI, 291
EUCA, 309
HLTH, 331
ICMP, 349
Installation, 53
MAP 3, 395
MBUS, 407
MRTM, 417
NOL, 475
PEER, 603
XMIT, 725
Logarithmic Ramp to Set Point, 589
Logical And, 71
Logical OR, 479
Look-up Table, 545

M

MAP 3, 395
MAP Transaction, 395
Master, 423
Math
AD16, 67
ADD, 69
BCD, 75
DIV, 115
DV16, 129
FTOI, 329
ITOF, 383
MU16, 465
MUL, 467
SU16, 687
SUB, 689
TEST, 715
Matrix
AND, 71
BROT, 87
CMPR, 101
COMP, 105
MBIT, 403
NBIT, 469
NCBT, 471, 473
OR, 479
RBIT, 625
SBIT, 639
SENS, 647
XOR, 743
MBIT, 403
MBUS, 407
MBUS Transaction, 407
Miscellaneous
CKSM, 97
DLOG, 119
EMTH, 133
EMTH-ADDDP, 139
EMTH-ADDFP, 143
EMTH-ADDIF, 147
EMTH-ANLOG, 151
EMTH-ARCOS, 155, 195
EMTH-ARSIN, 159
EMTH-ARTAN, 163
EMTH-CHSIN, 167
EMTH-CMPFP, 171
EMTH-CMPIF, 175
EMTH-CNVDR, 179
EMTH-CNVFI, 183
EMTH-CNVIF, 187

EMTH-CNVRD, 191
EMTH-DIVDP, 199
EMTH-DIVFI, 203
EMTH-DIVFP, 207
EMTH-DIVIF, 211
EMTH-ERLOG, 215
EMTH-EXP, 219
EMTH-LNFP, 223
EMTH-LOG, 227
EMTH-LOGFP, 231
EMTH-MULDP, 235
EMTH-MULFP, 239
EMTH-MULIF, 243
EMTH-PI, 247
EMTH-POW, 251
EMTH-SINE, 255
EMTH-SQRFP, 259
EMTH-SQRT, 263
EMTH-SQRTP, 267
EMTH-SUBDP, 271
EMTH-SUBFI, 275
EMTH-SUBFP, 279
EMTH-SUBIF, 283
EMTH-TAN, 287
LOAD, 391
MSTR, 423
SAVE, 635
SCIF, 641
XMRD, 735
XMWT, 739
Modbus Plus
MSTR, 423
Modbus Plus Network Statistics
MSTR, 452
Modify Bit, 403
Move
BLKM, 77
BLKT, 81
FIN, 321
FOUT, 325
IBKR, 345
IBKW, 347
R --> T, 621
SRCH, 659
T-->R, 691
T-->T, 695
TBLK, 711
MRTM, 417
MSTR, 423
Clear Local Statistics, 437
Clear Remote Statistics, 442
CTE Error Codes for SY/MAX and TCP/IP Ethernet, 464
Get Local Statistics, 435
Get Remote Statistics, 441
Modbus Plus and SY/MAX Ethernet Error Codes, 458
Modbus Plus Network Statistics, 452
Peer Cop Health, 444
Read CTE (Config Extension Table), 447
Read Global Data, 440
Reset Option Module, 446
SY/MAX-specific Error Codes, 460
TCP/IP Ethernet Error Codes, 462
TCP/IP Ethernet Statistics, 457
Write CTE (Config Extension Table), 449
Write Global Data, 439
MU16, 465
MUL, 467
Multiply, 467
Multiply 16 Bit, 465
Multi-Register Transfer Module, 417

N

NBIT, 469
NCBT, 471
Network Option Module for Lonworks, 475
NOBT, 473
NOL, 475
Normally Closed Bit, 471
Normally Open Bit, 473

O

ON/OFF Values for Deadband, 557
One Hundredth Second Timer, 699
One Millisecond Timer, 705
One Second Timer, 703
One Tenth Second Timer, 701
OR, 479

P

PCFL, 483
PCFL Subfunctions
General, 29
PCFL-AIN, 491
PCFL-ALARM, 497
PCFL-AOUT, 501
PCFL-AVER, 505
PCFL-CALC, 511
PCFL-DELAY, 515
PCFL-EQN, 521
PCFL-INTEG, 527
PCFL-KPID, 531
PCFL-LIMIT, 537
PCFL-LIMV, 541
PCFL-LKUP, 545
PCFL-LLAG, 549
PCFL-MODE, 553
PCFL-ONOFF, 557
PCFL-PI, 563
PCFL-PID, 567
PCFL-RAMP, 573
PCFL-RATE, 579
PCFL-RATIO, 583
PCFL-RMPLN, 589
PCFL-SEL, 593
PCFL-Subfunction
PCFL-AIN, 491
PCFL-ALARM, 497
PCFL-AOUT, 501
PCFL-AVER, 505
PCFL-CALC, 511
PCFL-DELAY, 515
PCFL-EQN, 521
PCFL-INTEG, 527
PCFL-KPID, 531
PCFL-LIMIT, 537
PCFL-LIMV, 541
PCFL-LKUP, 545
PCFL-LLAG, 549
PCFL-MODE, 553
PCFL-ONOFF, 557
PCFL-PI, 563
PCFL-PID, 567
PCFL-RAMP, 573
PCFL-RATE, 579
PCFL-RATIO, 583
PCFL-RMPLN, 589
PCFL-SEL, 593
PCFL-TOTAL, 597
PCFL-TOTAL, 597
PEER, 603
PEER Transaction, 603
PID Algorithms, 567
PID Example, 33
PID2, 607
PID2 Level Control Example, 37
Process Control Function Library, 483
Process Square Root, 267
Process Variable, 28
Proportional Integral Derivative, 607
Put Input in Auto or Manual Mode, 553

R

R --> T, 621
Raising a Floating Point Number to an Integer Power, 251
Ramp to Set Point at a Constant Rate, 573
RBIT, 625
READ, 627
MSTR, 433
Read, 627
READ/WRIT Operations, 41
Register to Table, 621
Regulatory Control, 484
Reset Bit, 625
RET, 633
Return from a Subroutine, 633

S

SAVE, 635
Save Flash, 635
SBIT, 639
SCIF, 641
Search, 659
SENS, 647
Sense, 647
Sequential Control Interfaces, 641
Set Bit, 639

Set Point Vaiable, 28
Skip (Constants), 651
Skip (Registers), 655
Skips / Specials
    RET, 633
    SKPC, 651
    SKPR, 655
Skips/Specials
    JSR, 385
    LAB, 387
SKPC, 651
SKPR, 655
Special
    DIOH, 111
    PCFL, 483
    PCFL-, 501
    PCFL-AIN, 491
    PCFL-ALARM, 497
    PCFL-AVER, 505
    PCFL-CALC, 511
    PCFL-DELAY, 515
    PCFL-EQN, 521
    PCFL-KPID, 531
    PCFL-LIMIT, 537
    PCFL-LIMV, 541
    PCFL-LKUP, 545
    PCFL-LLAG, 549
    PCFL-MODE, 553
    PCFL-ONOFF, 557
    PCFL-PI, 563
    PCFL-PID, 567
    PCFL-RAMP, 573
    PCFL-RATE, 579
    PCFL-RATIO, 583
    PCFL-RMPLN, 589
    PCFL-SEL, 593
    PCFL-TOTAL, 597
    PCPCFL-INTEGFL, 527
    PID2, 607
    STAT, 663
SRCH, 659
STAT, 663
Status, 663
SU16, 687
SUB, 689
Subroutine Handling, 51
Subtract 16 Bit, 687
Subtraction, 689
Support of the ESI Module, 291

T

T.01 Timer, 699
T-->R, 691
T-->T, 695
T0.1 Timer, 701
T1.0 Timer, 703
T1MS Timer, 705
Table to Block, 711
Table to Register, 691
Table to Table, 695
TBLK, 711
TCP/IP Ethernet Statistics
    MSTR, 457
TEST, 715
Test of 2 Values, 715
Time Delay Queue, 515
Totalizer for Metering Flow, 597

U

UCTR, 717
Up Counter, 717

V

Velocity Limiter for Changes in the Pv, 541

W

WRIT, 719
Write, 719
    MSTR, 431

X

XMIT, 725
XMIT Communication Block, 725
XMRD, 735
XMWT, 739
XOR, 743