Concept
Block Library LL984
Volume 1

840 USE 506 00 eng Version 2.6

33002261.00

Table of Contents

The chapters marked gray are not included in this volume.

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:

Instructions

1

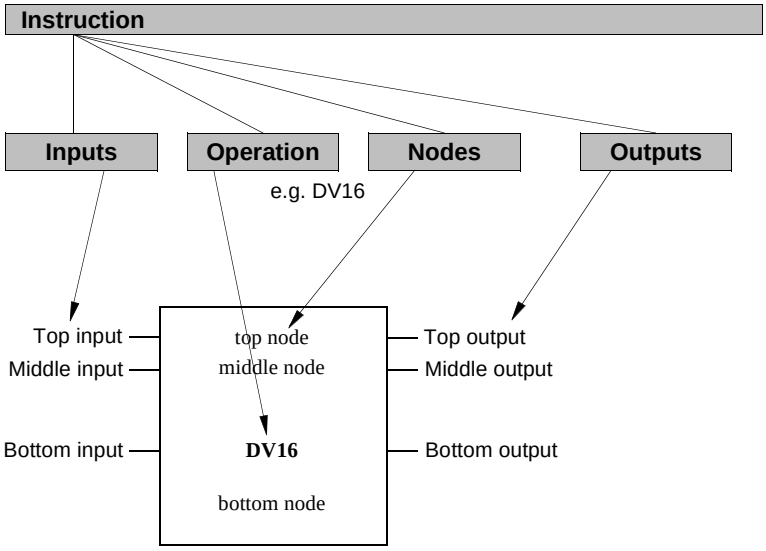
Parameter Assignment of Instuctions

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:

Instruction Groups

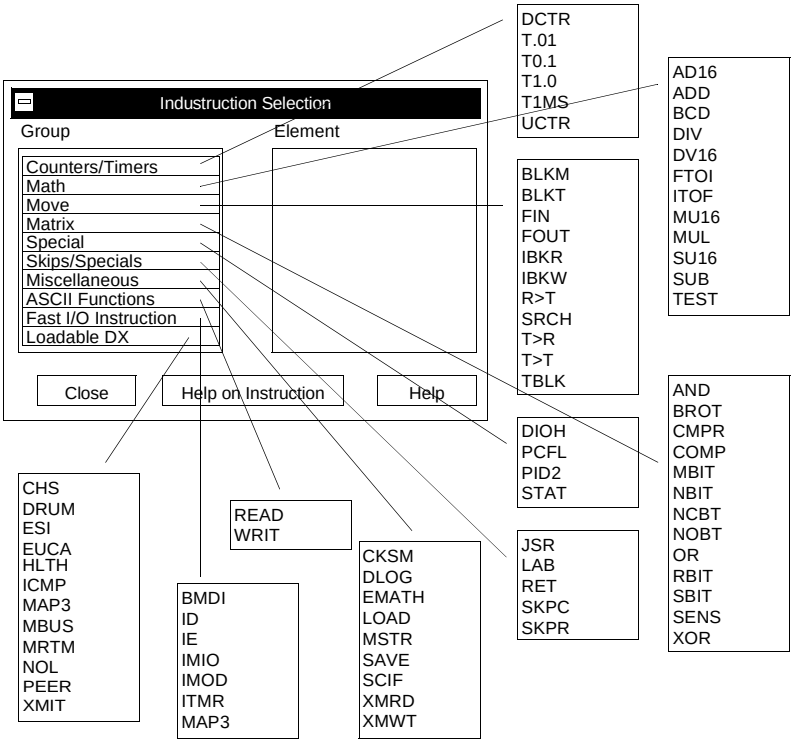
General

All instructions are attached to one of the following groups:

- ASCII Functions (See *ASCII Functions, p. 7*)
- Counters/Timers (See *Counters and Timers Instructions, p. 7*)
- Fast I/O Instructions (See *Fast I/O Instructions, p. 8*)
- Loadable DX (See *Loadable DX, p. 9*)
- Math (See *Math Instructions, p. 9*)
- Matrix (See *Matrix Instructions, p. 11*)
- Miscellaneous (See *Miscellaneous, p. 12*)
- Move (See *Move Instructions, p. 13*)
- Skips/Specials (See *Skips/Specials, p. 13*)
- Special (See *Special Instructions, p. 14*)
- *Coils, Contacts and Interconnects, p. 14*

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. 29*.

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	yes
DCTR	Counts down from a preset value to 0	yes	yes	yes	yes
T1.0	Timer that increments in seconds	yes	yes	yes	yes
T0.1	Timer that increments in tenths of a second	yes	yes	yes	yes
T.01	Timer that increments in hundredths of a second	yes	yes	yes	yes
T1MS	Timer that increments in one millisecond	yes (CPU 242 02 only)	yes	yes	yes

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	yes
ID	Disable interrupt	yes	yes	no	yes
IE	Enable interrupt	yes	yes	no	yes
IMIO	Immediate I/O instruction	yes	yes	no	yes
IMOD	Interrupt module instruction	yes	no	no	yes
ITMR	Interval timer interrupt	no	yes	no	yes

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

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	yes
ESI	Support of the ESI module 140 ESI 062 10	yes	no	no	no
EUCA	Engineering unit conversion and alarms	yes	yes	no	yes
HLTH	History and status matrices	yes	yes	no	yes
ICMP	Input comparison	yes	yes	no	yes
MAP3	MAP 3 Transaction	no	no	no	no
MBUS	MBUS Transaction	no	no	no	no
MRTM	Multi-register transfer module	yes	yes	no	yes
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. 41*.

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	yes
DIV	Division	yes	yes	yes	yes
MUL	Multiplication	yes	yes	yes	yes
SUB	Subtraction	yes	yes	yes	yes

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	yes
DV16	Divide 16 bit	yes	yes	yes	yes
MU16	Multiply 16 bit	yes	yes	yes	yes
SU16	Subtract 16 bit	yes	yes	yes	yes
TEST	Test of 2 values	yes	yes	yes	yes

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	yes
FTOI	Conversion from floating point to integer	yes	yes	yes	yes
ITOF	Conversion from integer to floating point	yes	yes	yes	yes

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	yes
BROT	Bit rotate	yes	yes	yes	yes
CMPR	Compare register	yes	yes	yes	yes
COMP	Complement a matrix	yes	yes	yes	yes
MBIT	Modify bit	yes	yes	yes	yes
NBIT	Bit control	yes	yes	no	yes
NCBT	Normally open bit	yes	yes	no	yes
NOBT	Normally closed bit	yes	yes	no	yes
OR	Logical OR	yes	yes	yes	yes
RBIT	Reset bit	yes	yes	no	yes
SBIT	Set bit	yes	yes	no	yes
SENS	Sense	yes	yes	yes	yes
XOR	Exclusive OR	yes	yes	yes	yes

Miscellaneous

Miscellaneous

This group provides the following instructions:

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

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	yes
BLKT	Table to block move	yes	yes	yes	yes
FIN	First in	yes	yes	yes	yes
FOUT	First out	yes	yes	yes	yes
IBKR	Indirect block read	yes	yes	no	yes
IBKW	Indirect block write	yes	yes	no	yes
R → T	Register to tabel move	yes	yes	yes	yes
SRCH	Search table	yes	yes	yes	yes
T → R	Table to register move	yes	yes	yes	yes
T → T	Table to table move	yes	yes	yes	yes
TBLK	Table to block move	yes	yes	yes	yes

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	yes
LAB	Label for a subroutine	yes	yes	yes	yes
RET	Return from a subroutine	yes	yes	yes	yes
SKPC	Skip (constant)	yes	yes	yes	yes
SKPR	Skip (register)	yes	yes	yes	yes

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	yes
PCFL	Process control function library	yes	yes	no	yes
PID2	Proportional integral derivative	yes	yes	yes	yes
STAT	Status	yes	yes	yes	yes

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:

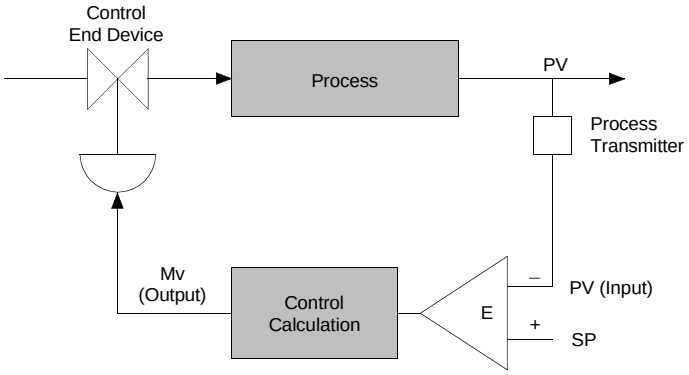
Closed Loop Control / Analog Values

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
$YI = 0$	

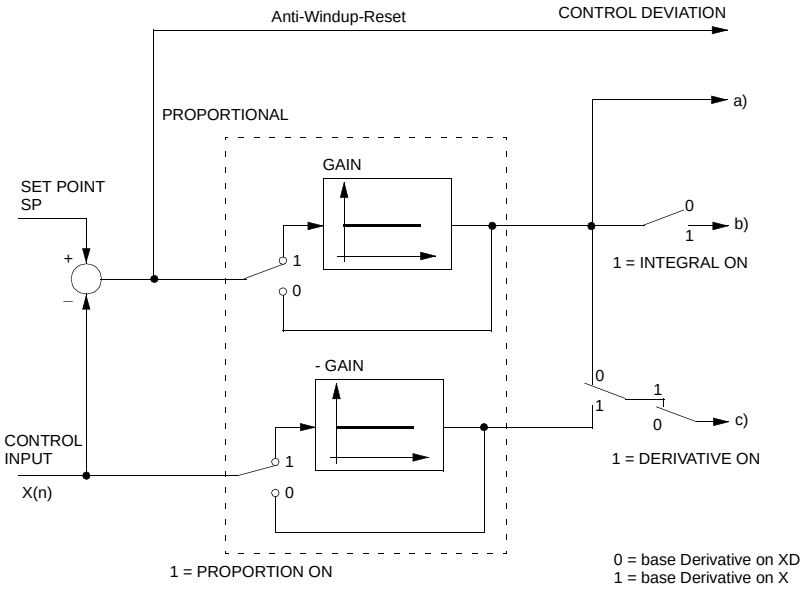
Derivative Calculation

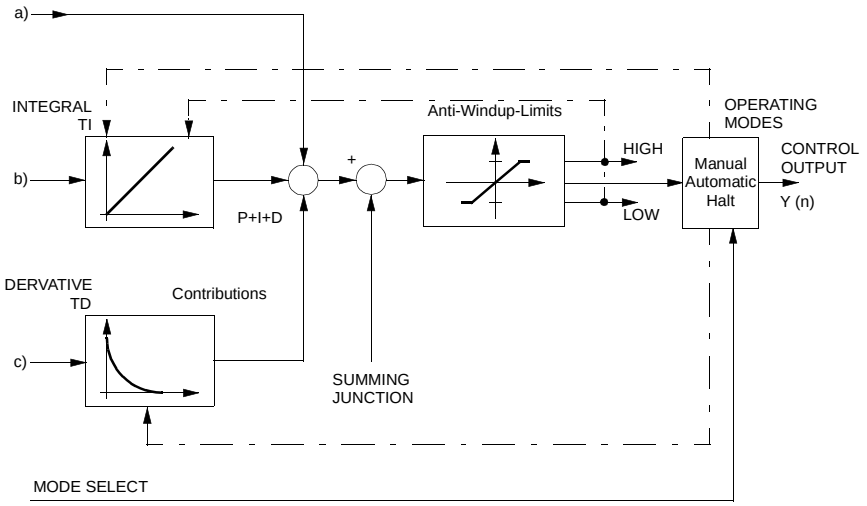
The following equations are valid:

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



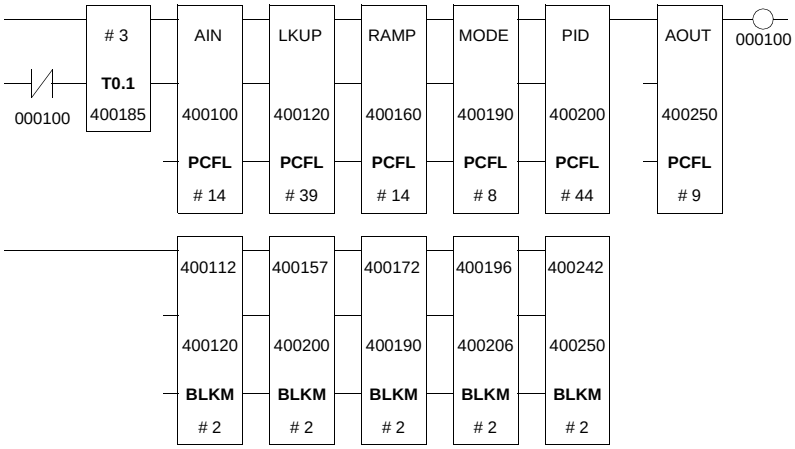


A PID Example

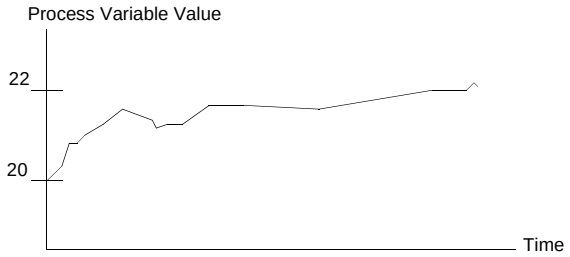
Description

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

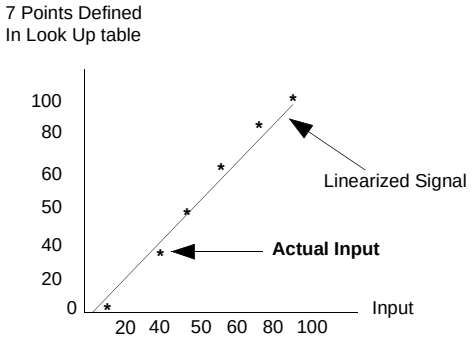


The process variable over time should look something like this:



Main PID Ladder Logic

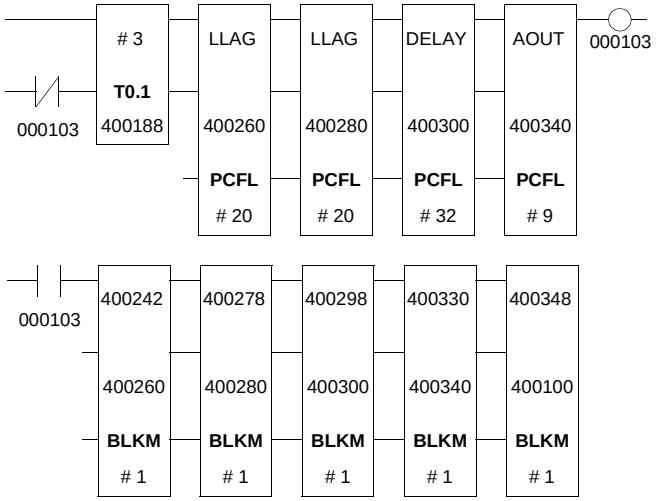
The AIN output is block moved to the LKUP function, which is used to scale the input signal. We do this because the input sensor is not likely to produce highly linear readings; the result is an ideal linear signal:



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.

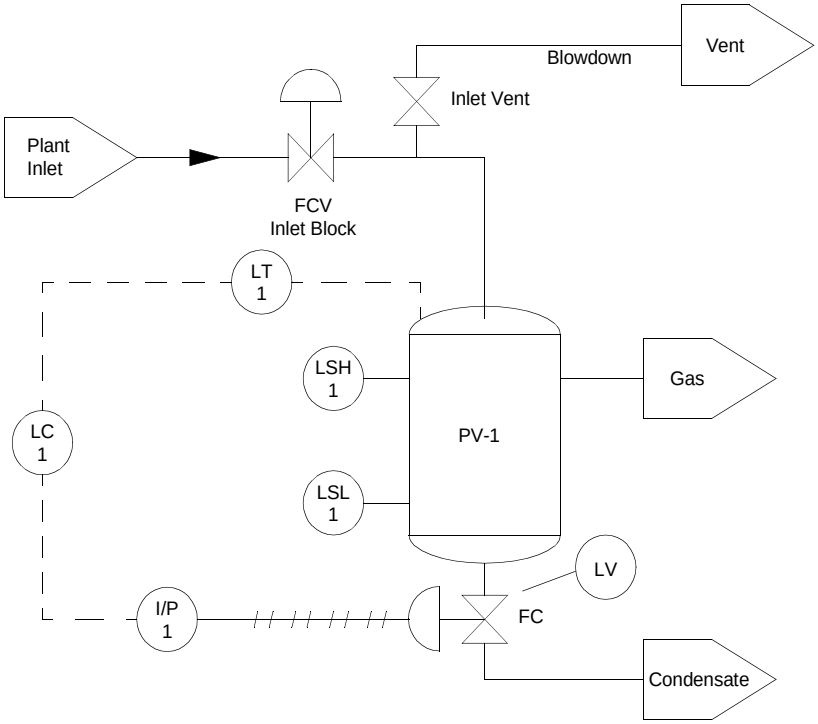
The entire control loop is preceded by a 0.1 s timer. The target solution interval for the entire loop is 1 s, and the full solve is 1 s. However, the nontime-dependent functions that are used (AIN, LKUP, MODE, and AOUT) do not need to be solved every scan. To reduce the scan time impact, these functions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

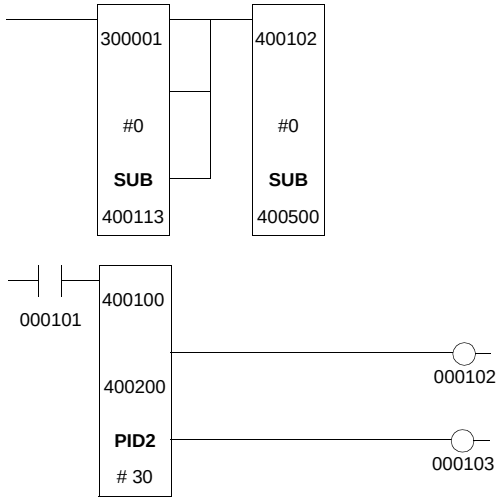
LC-1 level controller

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:

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).

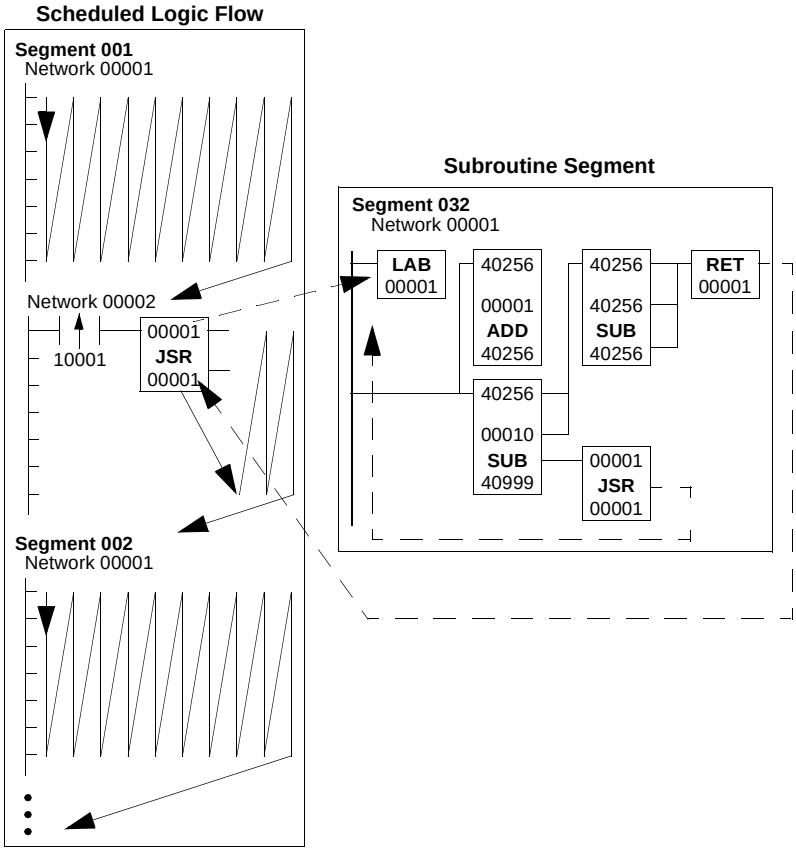
Subroutine Handling

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:

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 box 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 box 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:

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:

Short Description

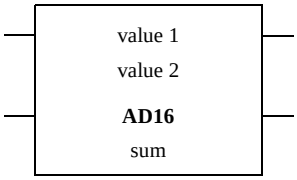
Function Description

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:

Short Description

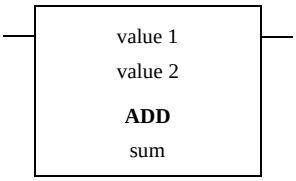
Function Description

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.

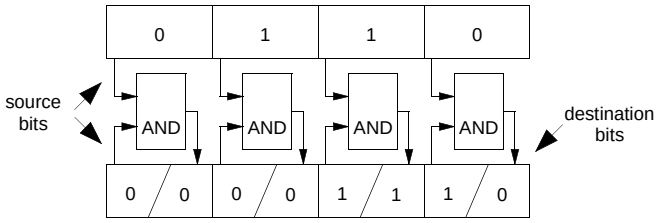
What's in this chapter?

This chapter contains the following topics:

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:

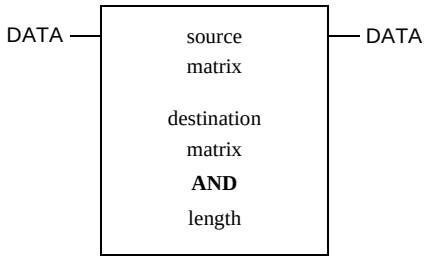


STOP	**WARNING**
	Overriding of any disabled coils within the destination matrix without enabling them. AND will override any disabled coils within the destination matrix without enabling them.This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's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

Short Description

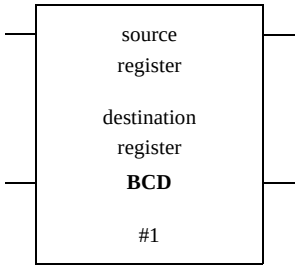
Function Description

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:

Short Description

Function Description

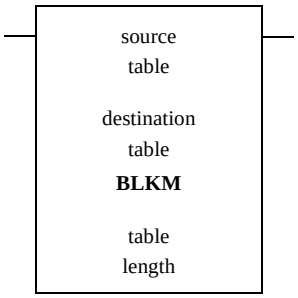
The BLKM (block move) instruction copies the entire contents of a source table to a destination table in one scan.

STOP	**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:

Short Description

Function Description

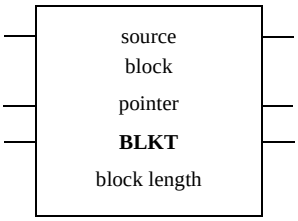
The BLKT (block-to-table) instruction combines the functions of R→T and BLKM in a single instruction. In one scan, it can copy data from a source block to a destination block in a table. The source block is of a fixed length. The block within the table is of the same length, but the overall length of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

STOP	**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:

Short Description

Function Description

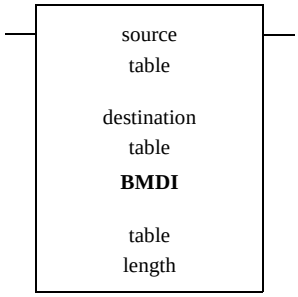
Note: This instruction is only available after configuring a CPU without extension.

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. 37*".

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = 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:

Short Description

Function Description

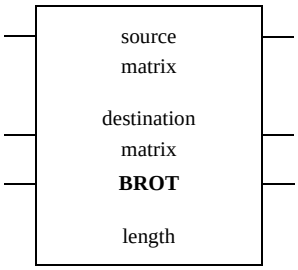
The BROT (bit rotate) instruction shifts the bit pattern in a source matrix, then posts the shifted bit pattern in a destination matrix. The bit pattern shifts left or right by one position per scan.

STOP	**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:

Short Description

Function Description

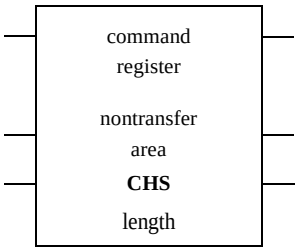
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

The 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.

STOP	**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. 84*

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:

Short Description

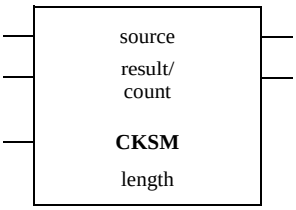
Function Description

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. 87*)	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:

Short Description

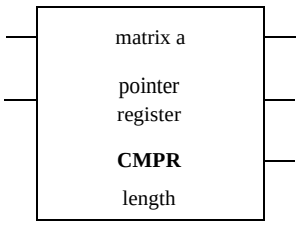
Function Description

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:

Short Description

Function Description

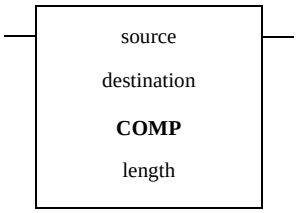
The COMP instruction complements the bit pattern, i.e. changes all 0's to 1's and all 1's to 0's, of a source matrix, then copies the complemented bit pattern into a destination matrix. The entire COMP operation is accomplished in one scan.

STOP	**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:

Short Description

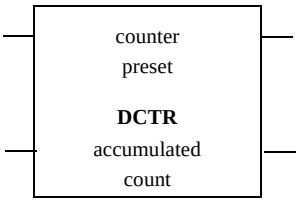
Function Description

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:

Short Description

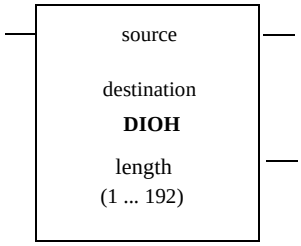
Function Description

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:

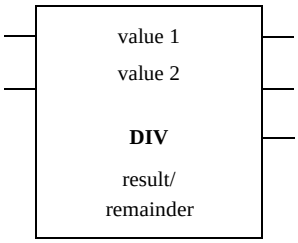
Short Description

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:

Short Description

Function Description

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

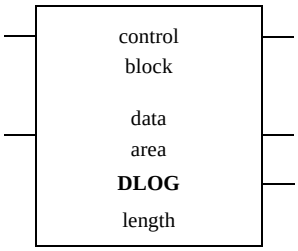
PCMCIA read and write support consists of a configuration extension to be implemented using a DLOG instruction. The DLOG instruction provides the facility for an application to copy data to a PCMCIA flash card, copy data from a PCMCIA flash card, erase individual memory blocks on a PCMCIA flash card,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

Short Description

Function Description

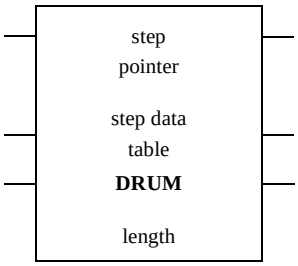
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

The 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:

Short Description

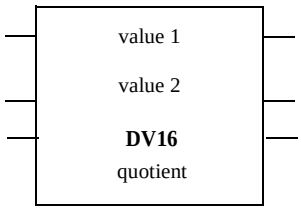
Function Description

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:

Short Description

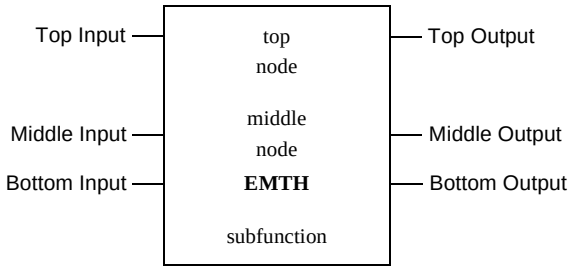
Function Description

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. 124*"
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. 124*"
Top output	0x	None	Depends on the selected EMTH function, see "*Inputs, Outputs and Bottom Node, p. 124*"
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. 126*)

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:

Short Description

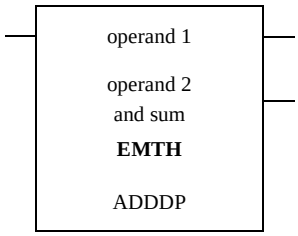
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 124*)".

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:

Short Description

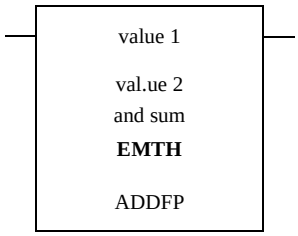
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

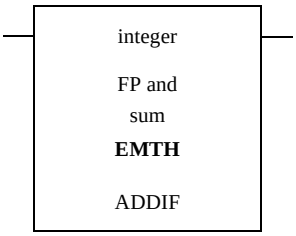
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

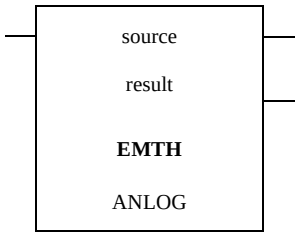
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 124*)".

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:

Short Description

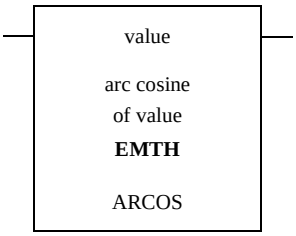
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 203*) 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:

Short Description

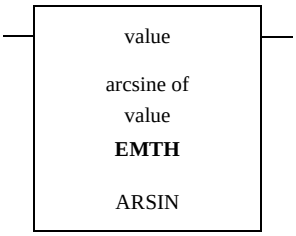
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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 -π/2 ... π/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. 203*) 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. 126*).

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:

Short Description

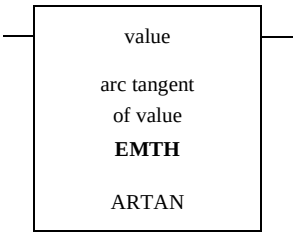
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

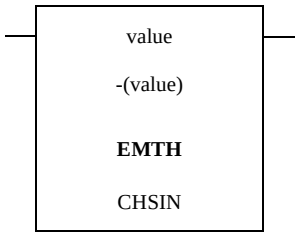
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

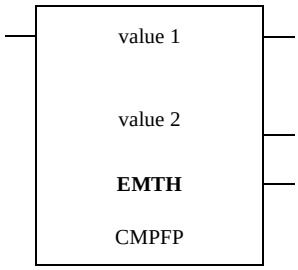
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

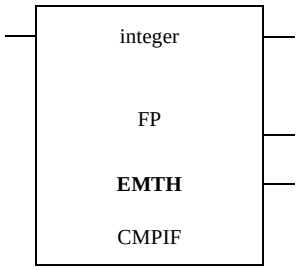
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

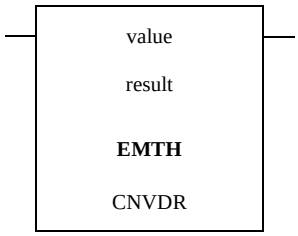
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*) 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. 126*) 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:

Short Description

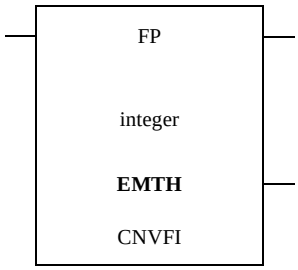
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 203*) 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:

Short Description

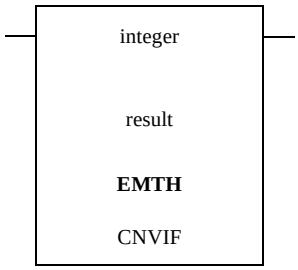
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*) 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. 203*) 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:

Short Description

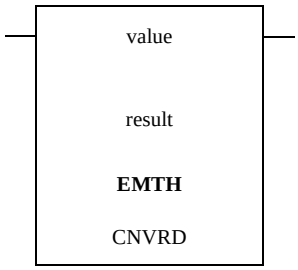
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*) 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. 126*) 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:

Short Description

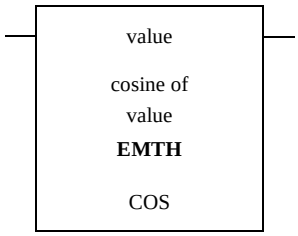
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 203*) 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. 126*).

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:

Short Description

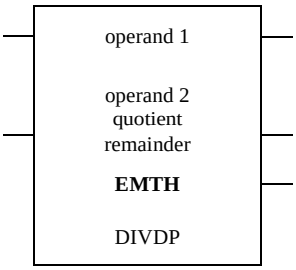
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 124*)".

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:

Short Description

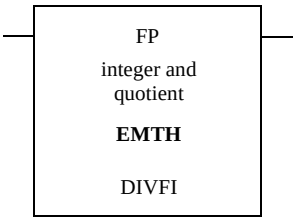
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

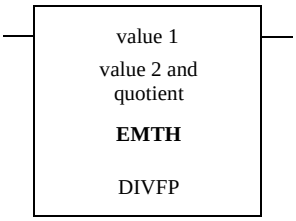
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

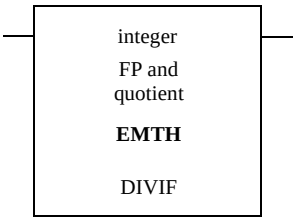
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

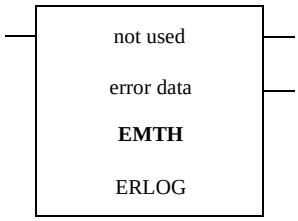
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 205*).
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:

Short Description

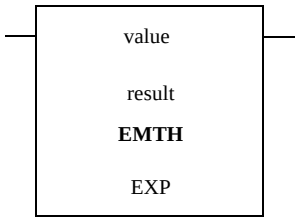
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*) 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. 126*).

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:

Short Description

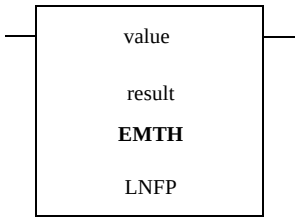
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*). 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. 126*).

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:

Short Description

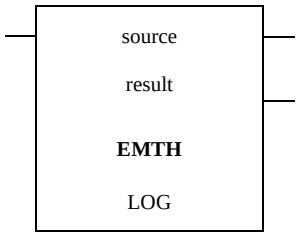
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 124*)".

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:

Short Description

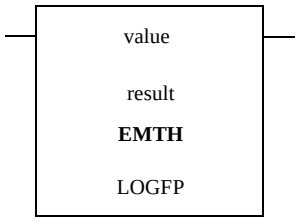
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*). 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. 126*).

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:

Short Description

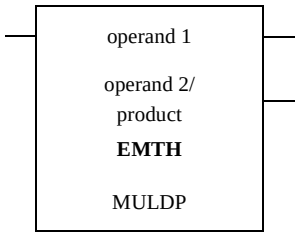
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 124*)".

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:

Short Description

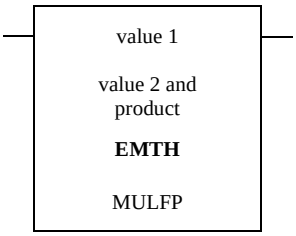
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

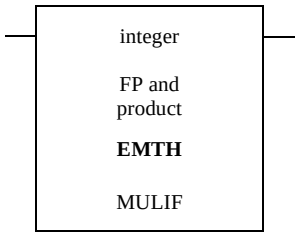
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

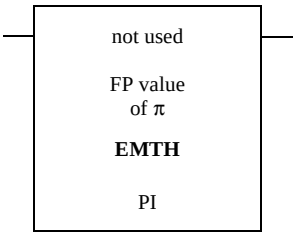
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

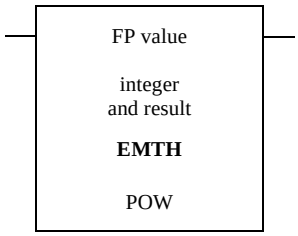
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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:

Short Description

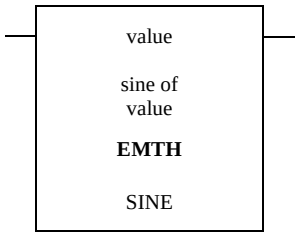
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 203*) 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. 126*).

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:

Short Description

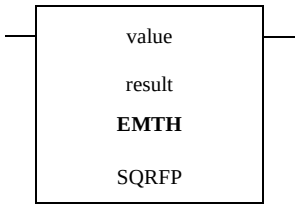
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

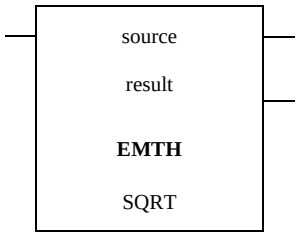
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Integer Math (See *Subfunctions for Integer Math, p. 124*)".

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:

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. 124*)".

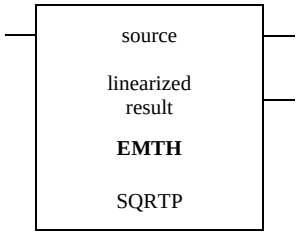
The process square root function tailors the standard square root function for closed loop analog control applications. It takes the result of the standard square root result, multiplies it by 63.9922 (the square root of 4 095) and stores that linearized result in the middle-node registers.

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.`**.

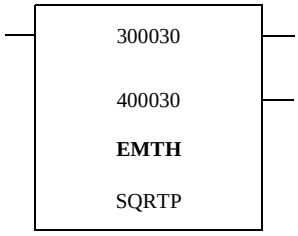
Register	Content
Displayed	This register stores the four-digit value to the left of the first decimal point.
First implied	This register stores the four-digit value to the right of the first decimal point.

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:

Short Description

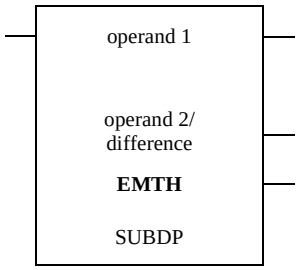
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Double Precision Math (See *Subfunctions for Double Precision Math, p. 124*)".

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:

Short Description

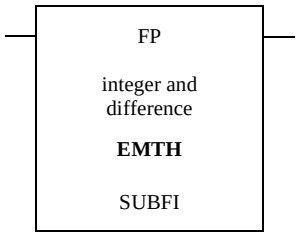
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

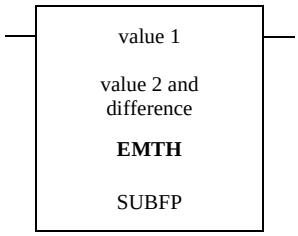
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

Short Description

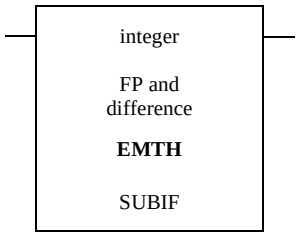
Function Description

This instruction is a subfunction of the EMTH instruction. It belongs to the category "Floating Point Math (See *Subfunctions for Floating Point Math, p. 125*)".

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. 126*).

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:

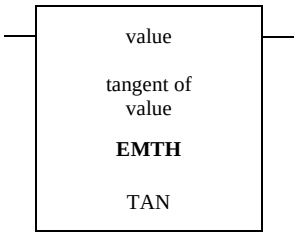
Short Description

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. 125*)".

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. 203*) 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. 126*).

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.

Index

D

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X