Concept
Block Library LL984
Volume 3

840 USE 506 00 eng Version 2.6

33002261.00

Table of Contents

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

PCFL: Process Control Function Library

96

At a Glance

Introduction

This chapter describes the instruction PCFL.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

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

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

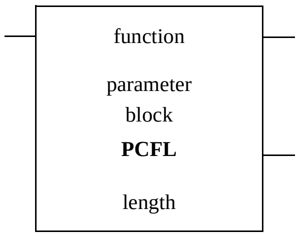
A PCFL function is selected from a list of alphabetical subfunctions in a pulldown menu in the panel software, and the subfunction is displayed in the top node of the instruction (see the table *Function (Top Node), p. 474* for a list of subfunctions and descriptions).

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
function (top node)			Selection of process control function (subfunction)
parameter block (middle node)	4x	INT, UINT, WORD	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
length (bottom node)		INT, UINT	Length of parameter block (depending on selected subfunction
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Function (Top Node)

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

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

Advanced Calculations

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

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

Signal Processing

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

Regulatory Control

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

Further information you will find in the section Closed Loop Control (See *PCFL Subfunctions, p. 17*).

Parameter Block (Middle Node)

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

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

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

Output Flags

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

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

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

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

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

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

Input Flags

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

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

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

Length (Bottom Node)

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

PCFL-AIN: Analog Input

97

At a Glance

Introduction

This chapter describes the subfunction PCFL-AIN.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

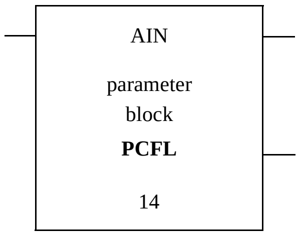
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

The AIN function scales the raw input produced by analog input modules to engineering values that can be used in the subsequent calculations.
Three scaling options are available:
- Auto input scaling
- Manual input scaling
- Implementing process square root on the input to linearize the signal before scaling

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
AIN (top node)			Selection of the subfunction AIN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction AIN (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

AIN supports the range resolutions for following device types:

Quantum Engineering Ranges

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

Quantum Thermocouple

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

Quantum Voltmeter

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

Parameter Block (Middle Node)

The length of the AIN parameter block is 14 registers:

Register	Content
Displayed	Input from a 3x register
First implied	Reserved
Second implied	*Output Status, p. 482*
Third implied	*Input Status, p. 483*
Fourth and fifth implied	Scale 100% engineering units
Sixth and seventh implied	Scale 0% engineering units
Eighth and ninth implied	Manual input
10th and 11th implied	Auto input
12th and 13th implied	Output

Output Status

Output Status

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

Bit	Function
1...5	Not used
6	1 = with TC PSQRT, invalid: in extrapolation range, PSQRT not used
7	1 = input out of range
8	1 = echo under range from input module
9	1 = echo over range from input module
10	1 = invalid output mode selected
11	1 = invalid Engineering Units
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 3	Standard input bits (flags) (See *Input Flags, p. 476*)
4 ... 8	Ranges (see following tables)
9	1 = process square root on raw input
10	1 = manual scaling mode 0 = auto scaling mode
11	1 = extrapolate over-/under-range for auto mode 0 = clamp over-/under-range for auto mode
12 ... 16	Not used

Quantum Engineering Ranges

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

Quantum Thermocouple

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

Quantum Voltmeter

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

Note: Bit 4 in this register is nonstandard use.

PCFL-ALARM: Central Alarm Handler

98

At a Glance

Introduction

This chapter describes the subfunction PCFL-Alarm.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

> **Note:** This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

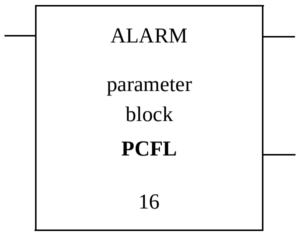
The ALARM function gives you a central block for alarm handling where you can set high (H), low (L), high high (HH), and low low (LL) limits on a process variable. ALARM lets you specify

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
ALARM (top node)			Selection of the subfunction ALARM
parameter block (middle node)	4x	INT, UINT, WORD	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction ALARM (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

The following operating modes are available:

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

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

Parameter Block (Middle Node)

The length of the ALARM parameter block is 16 registers:

Register	Content
Displayed and first implied	Input registers
Second implied	*Output Status, p. 488*
Third implied	*Input Status, p. 488*
Fourth and fifth implied	HH limit value
Sixth and seventh implied	H limit value
Eighth and ninth implied	L limit value
10th and 11th implied	LL limit value
12th and 13th implied	Deadband (DB) around limit
14th and 15th implied	Last input

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = DB set to negative number
6	1 = deviation mode chosen with DB option
7	1 = LL crossed (x ≤ LL
8	1 = L crossed (x ≤ L or LL < x ≤ L) with HH/LL option set
9	1 = H crossed (x ≥ H or H ≤ x < HH) with HH/LL option set
10	1 = HH crossed (x ≥ HH)
11	1 = invalid limits specified
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = deviation mode 0 = normal mode
6	1 = both H/L and HH/LL limits apply
7	1 = DB enabled
8	1 = retain H/L flag when HH/LL limits crossed
9 ... 16	Not used

PCFL-AOUT: Analog Output

99

At a Glance

Introduction

This chapter describes the subfunction PCFL-AOUT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

Formula

Formula of the AOUT function:

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

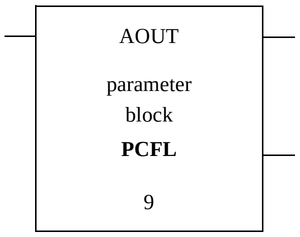
The meaning of the elements:

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
AOUT (top node)			Selection of the subfunction AOUT
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
9 (bottom node)		INT, UINT	Length of parameter block for subfunction AOUT (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the AOUT parameter block is 9 registers:

Register	Content
Displayed and first implied	Input in engineering units
Second implied	*Output Status, p. 492*
Third implied	*Input Status, p. 492*
Fourth and fifth implied	High engineering units
Sixth and seventh implied	Low engineering units
Eighth and ninth implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 7	Not used
8	1 = clamped low
9	1 = clamped high
10	not used
11	1 = invalid H/L limits
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-AVER: Average Weighted Inputs Calculate

100

At a Glance

Introduction

This chapter describes the subfunction PCFL-AVER.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 475*.

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

Formula

Formula of the AVER function:

$$\mathrm{RES} = \frac{(k + (w_1 \times \mathrm{In}_1) + (w_2 \times \mathrm{In}_2) + (w_3 \times \mathrm{In}_3) + (w_4 \times \mathrm{In}_4))}{1 + w_1 + w_2 + w_3 + w_4}$$

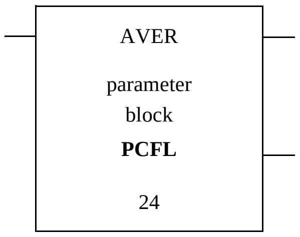
The meaning of the elements:

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables specified process control function
AVER (top node)			Selection of the subfunction AVER
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
24 (bottom node)		INT, UINT	Length of parameter block for subfunction AVER (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the AVER parameter block is 24 registers:

Register	Content
Displayed and first implied	reserved
Second implied	*Output Status, p. 496*
Third implied	*Input Status, p. 497*
Fourth and fifth implied	Value of In1
Sixth and seventh implied	Value of Inv2
Eighth and ninth implied	Value of In3
10th and 11th implied	Value of In4
12th and 13th implied	Value of k
14th and 15th implied	Value of wv1
16th and 17th implied	Value of wv2
18th and 19th implied	Value of wv3
20th and 21st implied	Value of wv4
22nd and 23rd implied	Value of result

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = no inputs activated
11	1 = result negative 0 = result positive
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = In4 and w4 are used
6	1 = In3 and w3 are used
7	1 = In2 and w2 are used
8	1 = In1 and w1 are used
9	1 = k is active
10 ... 16	Not used

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

PCFL-CALC: Calculated preset formula

101

At a Glance

Introduction

This chapter describes the subfunction PCFL-CALC.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

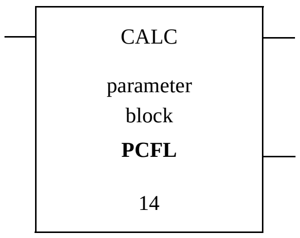
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 475*.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
CALC (top node)			Selection of the subfunction CALC
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction CALC (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the CALC parameter block is 14 registers:

Register	Content
Displayed and first implied	Reserved
Second implied	*Output Status, p. 501*
Third implied	*Input Status, p. 502*
Fourth and fifth implied	Value of input A
Sixth and seventh implied	Value of input B
Eighth and ninth implied	Value of input C
10th and 11th implied	Value of input D
12th and 13th implied	Value of the output

Output Status

Output Status

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

Bit	Function
1...10	Not used
11	1 = bad input code chosen
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 6	not used
7 ... 10	Formula Code
11 ... 16	Not used

Formula Code

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

PCFL-DELAY: Time Delay Queue

102

At a Glance

Introduction

This chapter describes the subfunction PCFL-DELAY.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

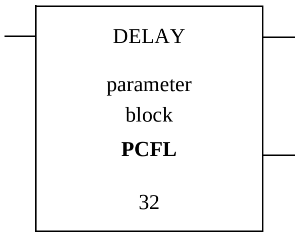
All values are carried along in registers, where register x[0] contains the current sampled input. The 10th delay period does not need to be stored. When the 10th instance in the sequence takes place, the value in register x[9] can be moved directly to the output

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables specified process control function
DELAY (top node)			Selection of the subfunction DELAY
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
32 (bottom node)		INT, UINT	Length of parameter block for subfunction DELAY (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the DELAY parameter block is 32 registers:

Register	Content
Displayed and first implied	Input at time n
Second implied	*Output Status, p. 506*
Third implied	*Input Status, p. 507*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	x[0] delay
12th and 13th implied	x[1] delay
14th and 15th implied	x[2] delay
...	...
28th and 29th implied	x[9] delay
30th and 31st implied	Output registers

Output Status

Output Status

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

Bit	Function
1...3	Not used
4	1 = k out of range
5 ... 8	Count of registers left to be initialized
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 8	Time Delay ≤ 10
9 ... 11	Echo number of registers left to be initialized
12 ... 16	Not used

PCFL-EQN: Formatted Equation Calculator

103

At a Glance

Introduction

This chapter describes the subfunction PCFL-EQN.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Advanced Calculations, p. 475*.

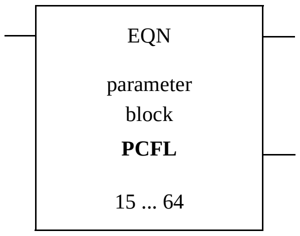
The EQN function is a formatted equation calculator. You must define the equation in the parameter block with various codes that specify operators, input selection and inputs.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
EQN (top node)			Selection of the subfunction EQN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
15 ... 64 (bottom node)		INT, UINT	Length of parameter block for subfunction EQN
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

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

Register	Content
Displayed and first implied	Reserved
Second implied	*Output Status, p. 511*
Third implied	*Input Status, p. 512*
Fourth and fifth implied	Variable A
Sixth and seventh implied	Variable B
Eighth and ninth implied	Variable C
10th and 11th implied	Variable D
12th and 13th implied	Output
14th implied	First *Formula Code, p. 512*
15th implied	Second possible formula code
...	...
63rd implied	Last possible formula code

Output Status

Output Status

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

Bit	Function
1	Stack error
2...3	Not used
4 ... 8	Code of last error logged
9	1 = bad operator selection code
10	1 = EQN not fully programmed
11	1 = bad input code chosen
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = Degree/radian option for trigonometry
6 ... 8	not used
9 ... 16	Equation size for display in Concept

Formula Code

Each formula code in the EQN function defines either an input selection code or an operator selection code.
Formula Code (Parameter Block)

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

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

Input Selection

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

Operator Selection

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

104 PCFL-INTEG: Integrate Input at Specified Interval

At a Glance

Introduction

This chapter describes the subfunction PCFL-INTEG.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

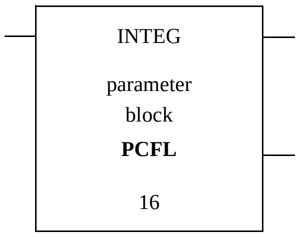
You can set flags to either initialize or restart the function after an undetermined down-time, and you can reset the integral sum if you wish. If you set the initialize flag, you must specify a reset value (zero or the last output in case of power failure), and calculations will be skipped for one sample.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
INTEG (top node)			Selection of the subfunction INTEG
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction INTEG (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the INTEG parameter block is 16 registers:

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 518*
Third implied	*Input Status, p. 518*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Reset value
14th and 15th implied	Result

Output Status

Output Status

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

Bit	Function
1...8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	Reset sum
6 ... 16	Not used

PCFL-KPID: Comprehensive ISA Non Interacting PID

105

At a Glance

Introduction

This chapter describes the subfunction PCFL-KPID.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

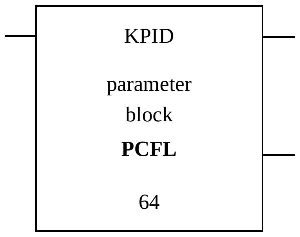
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

The KPID function offers a superset of the functionality of the PID function, with additional features that include:
- A gain reduction zone
- A separate register for bumpless transfer when the integral term is not used
- A reset mode
- An external set point for cascade control
- Built-in velocity limiters for set point changes and changes to a manual output
- A variable derivative filter constant
- Optional expansion of anti-reset wind-up limits

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
KPID (top node)			Selection of the subfunction KPID
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
64 (bottom node)		INT, UINT	Length of parameter block for subfunction KPID (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the KPID parameter block is 64 registers:

	Register	Content
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, Register 1, p. 523*
	Third implied	*Output Status, Register 2, p. 524*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 524*
Input Parameters	Sixth and seventh implied	Proportional rate, KP
	Eighth and ninth implied	Reset time, TI
	10th and 11th implied	Derivative action time, TD
	12th and 13th implied	Delay time constant, TD1
	14th and 15th implied	Gain reduction zone, GRZ
	16th and 17th implied	Gain reduction in GRZ, KGRZ
	18th and 19th implied	Limit rise of manual set point value
	20th and 21st implied	Limit rise of manual output
	22nd and 23rd implied	High limit for Y
	24th and 25th implied	Low limit for Y
	26th and 27th implied	Expansion for anti-reset wind-up limits
Inputs	28th and 29th implied	External set point for cascade
	30th and 31st implied	Manual set point
	32nd and 33rd implied	Manual Y
	34th and 35th implied	Reset for Y
	36th and 37th implied	Bias

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

Output Status, Register 1

Output Status Register 1

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4	1 = Cascade mode selected
5	1 = Auto mode selected
6	1 = Halt mode selected
7	1 = Manual mode selected
8	1 = Reset mode selected
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Output Status, Register 2

Output Status Register 2

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

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

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = Reset mode
6	1 = Manual mode
7	1 = Halt mode
8	1 = Cascade mode
9	1 = Solve proportional algorithm
10	1 = Solve integral algorithm
11	1 = Solve derivative algorithm
12	1 = solve derivative algorithm based on x 0 = solve derivative algorithm based on xd
13	1 = anti--reset wind-up on YI only 0 = normal anti--reset wind-up
14	1 = disable bumpless transfer 0 = bumpless transfer
15	1 = Manual Y tracks Y
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-LIMIT: Limiter for the Pv

106

At a Glance

Introduction

This chapter describes the subfunction PCFL-LIMIT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

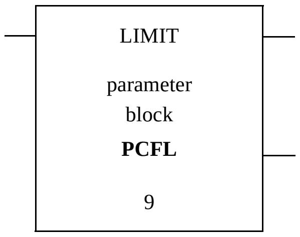
The LIMIT function limits the input to a range between a specified high and low value. If the high or low limit is reached, the function sets an H or L flag and clamps the output.

LIMIT returns a DXDONE message when the operation is complete.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
LIMIT (top node)			Selection of the subfunction LIMIT
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
9 (bottom node)		INT, UINT	Length of parameter block for subfunction LIMIT (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LIMIT parameter block is 9 registers:

Register	**Content**
Displayed and first implied	Current input
Second implied	*Output Status, p. 527*
Third implied	*Input Status, p. 527*
Fourth and fifth implied	Low limit
Sixth and seventh implied	High Limit
Eighth implied	Output register

Output Status

Output Status

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

Bit	**Function**
1...8	Not used
9	1 = input < low limit
10	1 = input > high limit
11	1 = invalid high/low limits (e.g., low ≥ high
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-LIMV: Velocity Limiter for Changes in the Pv

107

At a Glance

Introduction

This chapter describes the subfunction PCFL-LIMV.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

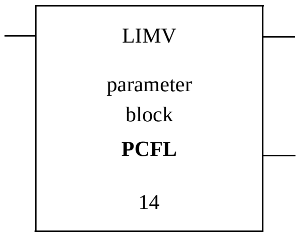
The LIMV function limits the velocity of change in the input variable between a specified high and low value. If the high or low limit is reached, the function sets an H or L flag and clamps the output.

LIMV returns a DXDONE message when the operation is complete.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
LIMV (top node)			Selection of the subfunction LIMV
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction LIMV (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LIMV parameter block is 14 registers:

Register	Content
Displayed and first implied	Input register
Second implied	*Output Status, p. 531*
Third implied	*Input Status, p. 531*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Velocity limit / sec
12th and 13th implied	Result

Output Status

Output Status

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

Bit	Function
1...5	Not used
6	1 = negative velocity limit
7	1 = input < low limit
8	1 = input > high limit
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-LKUP: Look-up Table

108

At a Glance

Introduction

This chapter describes the subfunction PCFL-LKUP.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

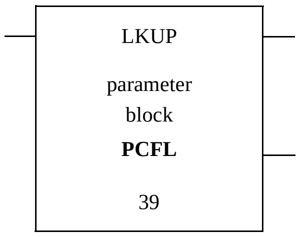
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
LKUP (top node)			Selection of the subfunction LKUP
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
39 (bottom node)		INT, UINT	Length of parameter block for subfunction LKUP (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

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

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

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

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

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

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

Parameter Block (Middle Node)

The length of the LKUP parameter block is 39 registers:

Register	Content
Displayed and first implied	Input
Second implied	*Output Status, p. 536*
Third implied	*Input Status, p. 536*
Fourth implied	Number of point pairs
Fifth and sixth implied	Point x1
Seventh and eighth implied	Point y1
Ninth and tenth implied	Point x2
11th and 12th implied	Point y2
. . .	. . .
33rd and 34th implied	Point x8
35th and 36th implied	Point y8
37th and 38th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = input clamped, i.e. out of table’s range
11	! = invalid number of points
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-LLAG: First-order Lead/Lag Filter

109

At a Glance

Introduction

This chapter describes the subfunction PCFL-LLAG.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

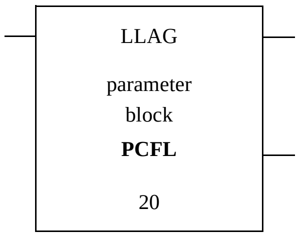
For best results, use lead and lag terms that are ≥ 4 *Δt. This will ensure sufficient granularity in the output response.

LLAG returns a DXDONE message when the operation completes

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
LLAG (top node)			Selection of the subfunction LLAG
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
20 (bottom node)		INT, UINT	Length of parameter block for subfunction LLAG (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the LLAG parameter block is 20 registers

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 540*
Third implied	*Input Status, p. 540*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Lead term
14th and 15th implied	Lag term
16th and 17th implied	Filter gain
18th and 19th implied	Result

Output Status

Output Status

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

Bit	Function
1...8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-MODE: Put Input in Auto or Manual Mode

110

At a Glance

Introduction

This chapter describes the subfunction PCFL-MODE.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

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

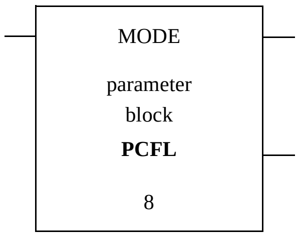
In auto mode, the input is copied to the output. In manual mode, the output is overwritten by a user entry.

MODE returns a DXDONE message when the operation completes.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables specified process control function
MODE (top node)			Selection of the subfunction MODE
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
8 (bottom node)		INT, UINT	Length of parameter block for subfunction MODE (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the MODE parameter block is 8 registers:

Register	Content
Displayed and first implied	Input
Second implied	*Output Status, p. 544*
Third implied	*Input Status, p. 544*
Fourth and fifth implied	Manual input
Sixth and seventh implied	Output register

Output Status

Output Status

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

Bit	Function
1 ... 10	Not used
11	Echo mode: 1 = manual mode 0 = auto mode
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = manual mode 0 = auto mode
6 ... 16	Not used

PCFL-ONOFF: ON/OFF Values for Deadband

111

At a Glance

Introduction

This chapter describes the subfunction PCFL-ONOFF.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

The ONOFF function is used to control the output signal between fully ON and fully OFF conditions so that a user can manually force the output ON or OFF.
You can control the output via either a direct or reverse configuration:

Configuration	IF Input...	Then Output...
Direct	< (SP - DB)	ON
	> (SP + DB)	OFF
Revers	> (SP + DB)	ON
	< (SP - DB)	OFF

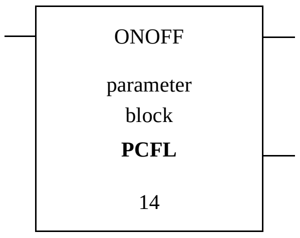
Manual Override

Two bits in the input status register (the third implied register in the parameter block) are used for manual override. When bit 6 is set to 1, manual mode is enforced. In manual mode, a 0 in bit 7 forces the output OFF, and a 1 in bit 7 forces the output ON. The state of bit 7 has meaning only in manual mode.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
ONOFF (top node)			Selection of the subfunction ONOFF
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction ONOFF (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the ONOFF parameter block is 14 registers:

Register	Content
Displayed and first implied	Current Input
Second implied	*Output Status, p. 548*
Third implied	*Input Status, p. 549*
Fourth and fifth implied	Set point, SP
Sixth and seventh implied	Deadband (DB) around SP
Eighth and ninth implied	Fully ON (maximum output)
10th and 11th implied	Fully OFF (minimum output)
12th and 13th implied	Output, ON or OFF

Output Status

Output Status

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

Bit	Function
1 ... 8	Not used
9	1 = DB set to negative number
10	Echo mode: 1 = manual override 0 = auto mode
11	1 = output set to ON 0 = output set to OFF
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = reverse configuration 0 = direct configuration
6	1 = manual override 0 = auto mode
7	1 = force output ON in manual mode 0 = force output OFF in manual mode
8 ... 16	Not used

PCFL-PI: ISA Non Interacting PI

112

At a Glance

Introduction

This chapter describes the subfunction PCFL-PI.

What’s in this chapter?

This chapter contains the following topics:

Short Description

Function Description

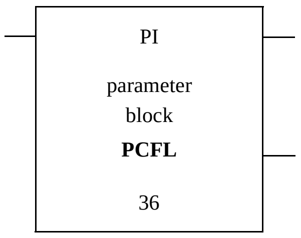
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

The PI function performs a simple proportional-integral operations using floating point math. It features halt / manual / auto operation modes. It is similar to the PID and KPID functions but does not contain as many options. It is available for higher-speed loops or inner loops in cascade strategies.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
PI (top node)			Selection of the subfunction PI
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
36 (bottom node)		INT, UINT	Length of parameter block for subfunction PI (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the PI parameter block is 36 registers:

	Register	**Content**
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, p. 554*
	Third implied	*Error Word, p. 554*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 554*
Inputs	Sixth and seventh implied	Set point, SP
	Eighth and ninth implied	Manual output
	10th and 11th implied	Calculated control difference (error), XD
Outputs	12th implied	Previous operating mode
	13th and 14th implied	Dt (in ms) since last solve
	15th and 16th implied	Previous system deviation, XD_1
	17th and 18th implied	Integral part of output Y
	19th and 20th implied	Previous input, X_1
	21st implied	Previous operating status
Timing Information	22nd implied	10 ms clock at time n
	23rd implied	Reserved
	24th and 25th implied	Solution interval (in ms)
Input Parameters	26th and 27th implied	Proportional rate, KP
	28th and 29th implied	Reset time, TI
	30th and 31st implied	High limit on output Y
	32nd and 33rd implied	Low limit on output Y
Output	34th and 35th implied	Manipulated variable output, Y

Output Status

Output Status

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Error Word

Error Word

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

Bit	Function
1...11	Not used
12 ... 16	Error Description

Error Description

Bit					Meaning
12	**13**	**14**	**15**	**16**	
1	0	1	1	0	Negative integral time constant
1	0	1	0	1	High/low limit error (low ≥ high)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	Not used
6	1 = Manual mode
7	1 = Halt mode
8 ... 15	Not used
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-PID: PID Algorithms

113

At a Glance

Introduction

This chapter describes the subfunction PCFL-PID.

What’s in this chapter?

This chapter contains the following topics:

Short Description

Function Description

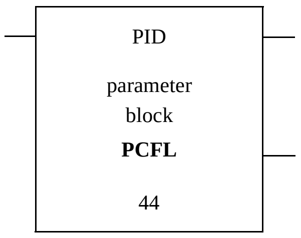
Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

The PID function performs ISA non-interacting proportional-integral-derivative (PID) operations using floating point math. Because it uses FP math (unlike PID2), round-off errors are negligible.
In the part "General Information" you will find *A PID Example, p. 21*.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
PID (top node)			Selection of the subfunction PID
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
44 (bottom node)		INT, UINT	Length of parameter block for subfunction PID (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the KPID parameter block is 44 registers:

	Register	**Content**
General Parameters	Displayed and first implied	Live input, x
	Second implied	*Output Status, p. 558*
	Third implied	*Error Word, p. 558*
	Fourth implied	Reserved
	Fifth implied	*Input Status, p. 559*
Inputs	Sixth and seventh implied	Set point, SP
	Eighth and ninth implied	Manual output
	10th and 11th implied	Summing junction, Bias
Outputs	12th and 13th implied	Error, XD
	14th implied	Previous operating mode
	15th and 16th implied	Elapsed time (in ms) since last solve
	17th and 18th implied	Previous system deviation, XD_1
	19th and 20th implied	Previous input, X_1
	21st and 22nd implied	Integral part of output Y, YI
	23rd and 24th implied	Differential part of output Y, YD
	25th and 26th implied	Proportional part of output Y, YP
	27th implied	Previous operating status
Timing Information	28th implied	Current time
	29th implied	Reserved
Inputs	30th and 31st implied	Solution interval (in ms)
	34th and 35th implied	Reset time, TI
	36th and 37th implied	Derivative action time, TD
	38th and 39th implied	High limit on output Y
	40th and 41st implied	Low limit on output Y
	42nd and 43rd implied	Manipulated control output, Y

Output Status

Output Status

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

Bit	Function
1	Error
2	1 = low limit exceeded
3	1 = high limit exceeded
4 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Error Word

Error Word

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

Bit	Function
1...11	Not used
12 ... 16	Error Description

Error Description

Bit					Meaning
12	**13**	**14**	**15**	**16**	
1	0	1	1	1	Negative derivative time constant
1	0	1	1	0	Negative integral time constant
1	0	1	0	1	High/low limit error (low ≥ high)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	Not used
6	1 = Manual mode
7	1 = Halt mode
8	Not used
9	1 = Solve proportional algorithm
10	1 = Solve integral algorithm
11	1 = Solve derivative algorithm
12	1 = solve derivative algorithm based on x 0 = solve derivative algorithm based on xd
13... 15	Not used
16	1 = reverse action for loop output 0 = direct action for loop output

PCFL-RAMP: Ramp to Set Point at a Constant Rate

114

At a Glance

Introduction

This chapter describes the subfunction PCFL-RAMP.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

The RAMP function allows you to ramp up linearly to a target set point at a specified approach rate.
You need to specify:
- The target set point, in the same units as the contents of the input register are specified
- The sampling rate
- A positive rate toward the target set point, negative rates are illegal

The direction of the ramp depends on the relationship between the target set point and the input, i.e. if x < SP, the ramp is up; if x > SP, the ramp is down.

You may use a flag to initialize after an undetermined down-time. The function will store a new sample, then wait for one cycle to collect the second sample. Calculations will be skipped for one cycle and the output will be left as is, after which the ramp will resume.

RAMP terminates when the entire ramping operation is complete (over multiple scans) and returns a DXDONE message.

Starting the Ramp

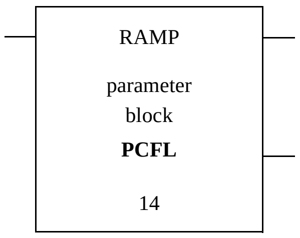
The following steps need to be done when starting the ramp (up/down) and each and every time you need to start or restart the ramp.

Step	Action
1	Set bit 1 of the standard input bits (See *Input Flags, p. 476*) to "1" (third implied register of the parameter block).
2	Retoggle the top input (enable input) to the instruction. Ramp will now start to ramp up/down from the initial value previously configured up/down to the previously configured setpoint. Monitor the 12th implied register of the parameter block for floating point value of the ramp value in progress.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables specified process control function
RAMP (top node)			Selection of the subfunction RAMP
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction RAMP (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RAMP parameter block is 14 registers:

Register	Content
Displayed and first implied	Set point (Input)
Second implied	*Output Status, p. 564*
Third implied	*Input Status, p. 564*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Rate of change (per second) toward set point
12th and 13th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = ramp rate is negative
6	1 = ramp complete 0 = ramp in progress
7	1 = ramping down
8	1 = ramping up
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

Top Output (Operation Succesfull)

The top output of the PCFL subfunction RAMP goes active at each successive discrete ramp step up/down. It happens so fast that it appears to be solidly on. This top output should **NOT** be used as "Ramp done bit".

Bit 6 of the output status (second impied register of the parameter block) should be monitored as "Ramp done bit".

PCFL-RATE: Derivative Rate Calculation over a Specified Time

115

At a Glance

Introduction

This chapter describes the subfunction PCFL-RATE.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

The RATE function calculates the rate of change over the last two input values. If you set an initialization flag, the function records a sample and sets the appropriate flags.

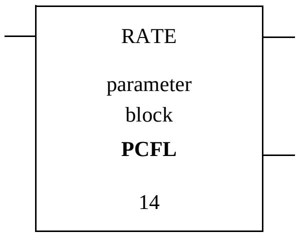
If a divide-by-zero operation is attempted, the function returns a DXERROR message.

It returns a DXDONE message when the operation completes successfully.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables specified process control function
RATE (top node)			Selection of the subfunction RATE
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction RATE (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RATE parameter block is 14 registers:

Register	Content
Displayed and first implied	Current input
Second implied	*Output Status, p. 570*
Third implied	*Input Status, p. 570*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input
12th and 13th implied	Result

Output Status

Output Status

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

Bit	Function
1 ... 8	Not used
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-RATIO: Four Station Ratio Controller

116

At a Glance

Introduction

This chapter describes the subfunction PCFL-RATIO.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

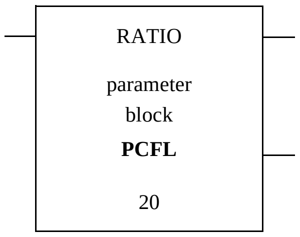
The RATIO function provides a four-station ratio controller. Ratio control can be used in applications where one or more raw ingredients are dependent on a primary ingredient. The primary ingredient is measured, and the measurement is converted to engineering units via an AIN function. The converted value is used to set the target for the other ratioed inputs.

Outputs from the ratio controller can provide set points for other controllers. They can also be used in an open loop structure for applications where feedback is not required.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
RATIO (top node)			Selection of the subfunction RATIO
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
20 (bottom node)		INT, UINT	Length of parameter block for subfunction RATIO (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RATIO parameter block is 20 registers:

Register	Content
Displayed and first implied	Live input
Second implied	*Output Status, p. 574*
Third implied	*Input Status, p. 575*
Fourth and fifth implied	Ratio for input 1
Sixth and seventh implied	Ratio for input 2
Eighth and ninth implied	Ratio for input 3
10th and 11th implied	Ratio for input 4
12th and 13th implied	Output for input 1
14th and 15th implied	Output for input 2
16th and 17th implied	Output for input 3
18th and 19th implied	Output for input 4

Output Status

Output Status

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

Bit	Function
1 ... 9	Not used
10	1 = parameter(s) out of range
11	1 = no inputs activated
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1= input 4 active
6	1= input 3 active
7	1= input 2 active
8	1= input 1 active
9 ... 16	Not used

PCFL-RMPLN: Logarithmic Ramp to Set Point

117

At a Glance

Introduction

This chapter describes the subfunction PCFL-RMPLN.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

The RMPLN function allows you to ramp up logarithmically to a target set point at a specified approach rate. At each successive call, it calculates the output until it is within a specified deadband (DB). DB is necessary because the incremental distance the ramp crosses decreases with each solve.

You need to specify:
- The target set point, in the same units as the contents of the input register are specified
- The sampling rate
- The time constant used for the logarithmic ramp, which is the time it takes to reach 63.2% of the new set point

For best results, use a t that is ≥4 *Δt. This will ensure sufficient granularity in the output response.

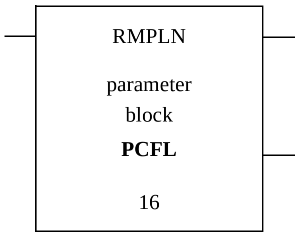
You may use a flag to initialize after an undetermined down-time. The function will store a new sample, then wait for one cycle to collect the second sample. Calculations will be skipped for one cycle and the output will be left as is, after which the ramp will resume.

RMPLN terminates when the input reaches the target set point + the specified DB and returns a DXDONE message.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
RMPLN (top node)			Selection of the subfunction RMPLN
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
16 (bottom node)		INT, UINT	Length of parameter block for subfunction RMPLN (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the RMPLN parameter block is 16 registers:

Register	Content
Displayed and first implied	Set point (Input)
Second implied	*Output Status, p. 580*
Third implied	*Input Status, p. 580*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Time constant, τ, (per second) of exponential ramp toward the target set point
12th and 13th implied	DB (in engineering units)
14th and 15th implied	Output

Output Status

Output Status

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

Bit	Function
1 ... 4	Not used
5	1 = DB or τ set to negative units
6	1 = ramp complete 0 = ramp in progress
7	1 = ramping down
8	1 = ramping up
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5 ... 16	Not used

PCFL-SEL: Input Selection

118

At a Glance

Introduction

This chapter describes the subfunction PCFL-SEL.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Signal Processing, p. 475*.

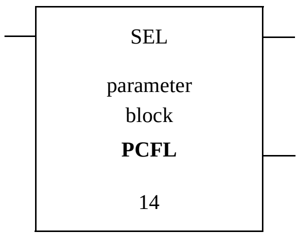
The SEL function compares up to four inputs and makes a selection based upon either the highest, lowest, or average value. You choose the inputs to be compared and the comparison criterion. The output is a copy of the selected input.

SEL returns a DXDONE message when the operation is complete.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
SEL (top node)			Selection of the subfunction SEL
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
14 (bottom node)		INT, UINT	Length of parameter block for subfunction SEL (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Parameter Block (Middle Node)

The length of the SEL parameter block is 14 registers:

Register	**Content**
Displayed and first implied	Reserved
Second implied	*Output Status, p. 583*
Third implied	*Input Status, p. 584*
Fourth and fifth implied	Input 1
Sixth and seventh implied	Input 2
Eighth and ninth implied	Input 3
10th and 11th implied	Input 4
12th and 13th implied	Output

Output Status

Output Status

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

Bit	**Function**
1 ... 9	Not used
10	Invalid selection modes
11	No inputs selected
12 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	Function
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = enable input 1 0 = disable input 1
6	1 = enable input 2 0 = dyeable input 2
7	1 = enable input 3 0 = dyeable input 3
8	1 = enable input 4 0 = dyeable input 4
9 ... 10	Selection mode
11 ... 16	Not used

Selection mode

Bit		Meaning
9	**10**	
0	0	Select average
0	1	Select high
1	0	Select low
1	1	reserved / invalid

PCFL-TOTAL: Totalizer for Metering Flow

119

At a Glance

Introduction

This chapter describes the subfunction PCFL-TOTAL.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

Note: This instruction is a subfunction of the PCFL instruction. It belongs to the category *Regulatory Control, p. 475*.

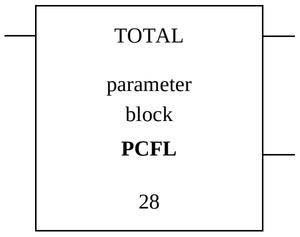
The TOTAL function provides a material totalizer for batch processing reagents. The input signal contains the units of weight or volume per unit of time. The totalizer integrates the input over time.

The algorithm reports three outputs:
- The integration sum
- The remainder left to meter in
- The valve output (in engineering units).

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables specified process control function
TOTAL (top node)			Selection of the subfunction TOTAL
parameter block (middle node)	4x	INT, UINT	First in a block of contiguous holding registers where the parameters for the specified subfunction are stored
28 (bottom node)		INT, UINT	Length of parameter block for subfunction TOTAL (can not be changed)
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = error

Parameter Description

Mode of Functioning

The function uses up to three different set points:
- A trickle flow set point
- A target set point
- An auxiliary trickle flow set point

The target set point is for the full amount to be metered in. Here the output will be turned OFF.

The trickle flow set point is the cut-off point when the output should be decreased from full flow to a percentage of full flow so that the target set point is reached with better granularity.
The auxiliary trickle flow set point is optional. It is used to gain another level of granularity. If this set point is enabled, the output is reduced further to 10% of the trickle output.

The totalizer works from zero as a base point. The set point must be a positive value

In normal operation, the valve output is set to 100% flow when the integrated value is below the trickle flow set point. When the sum crosses the trickle flow set point, the valve flow becomes a programmable percentage of full flow. When the sum reaches the desired target set point, the valve output is set to 0% flow.

Set points can be relative or absolute. With a relative set point, the deviation between the last summation and the set point is used. Otherwise, the summation is used in absolute comparison to the set point.

There is a halt option to stop the system from integrating.

When the operation has finished, the output summation is retained for future use. You have the option of clearing this sum. In some applications, it is important to save the sum, e.g. if the meters or load cells cannot handle the full batch in one charge and measurements are split up, if there are several tanks to fill for a batch and you want to keep track of batch and production sums.

Parameter Block (Middle Node)

The length of the TOTAL parameter block is 28 registers:

Register	Content
Displayed and first implied	Live input
Second implied	*Output Status, p. 589*
Third implied	*Input Status, p. 590*
Fourth implied	Time register
Fifth implied	Reserved
Sixth and seventh implied	Δt (in ms) since last solve
Eighth and ninth implied	Solution interval (in ms)
10th and 11th implied	Last input, X_1
12th and 13th implied	Reset value
14th and 15th implied	Set point, target
16th and 17th implied	Set point, trickle flow
18th and 19th implied	% of full flow for trickle flow set point
20th and 21st implied	Full flow
22nd and 23rd implied	Remaining amount to SP
24th and 25th implied	Resulting sum
26th and 27th implied	Output for final control element

Output Status

Output Status

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

Bit	Function
1 ... 2	Not used
3 ... 4	0 0 = OFF 0 1 = trickle flow 1 0 = full flow
5	1 = operation done
6	1 = totalizer running
7	1 = overshoot past set point by more than 5%
8	1 = parameter(s) out of range
9 ... 16	Standard output bits (flags) (See *Output Flags, p. 476*)

Input Status

Input Status

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

Bit	**Function**
1 ... 4	Standard input bits (flags) (See *Input Flags, p. 476*)
5	1 = reset sum
6	1 = halt integration
7	1 = deviation set point 0 = absolute set point
8	1 = use auxiliary trickle flow set point
9 ... 16	Not used

PEER: PEER Transaction

120

At a Glance

Introduction

This chapter describes the instruction PEER.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

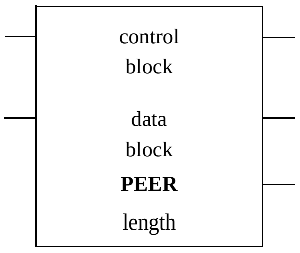
Note: This instruction is only available, if you have unpacked and installed the DX Loadables; further information in the chapter "*Installation of DX Loadables, p. 41*".

The S975 Modbus II Interface option modules use two loadable function blocks: MBUS and PEER. The PEER instruction can initiate identical message transactions with as many as 16 devices on Modbus II at one time. In a PEER transaction, you may only write register data.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Enable MBUS transaction
Middle input	0x, 1x	None	Repeat transaction in same scan
control block (top node)	4x	INT, UINT, WORD	First of 19 contiguous registers in the PEER control block
data block (middle node)	4x	INT, UINT	First register in a data block to be transmitted by the PEER function
length (bottom node)		INT, UINT	Length, i.e. the number of holding registers, of the data block; range: 1 ... 249.
Top output	0x	None	Transaction complete
Middle output	0x	None	Transaction in progress or new transaction starting
Bottom output	0x	None	Error detected in transaction

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of 19 contiguous registers in the PEER control block:

Register	Function
Displayed	Indicates the status of the transactions at each device, the leftmost bit being the status of device #1 and the rightmost bit the status of device #16: 0 = OK, 1 = transaction error
First implied	Defines the reference to the first 4x register to be written to in the receiving device; a 0 in this field is an invalid value and will produce an error (the bottom output will go ON)
Second implied	Time allowed for a transaction to be completed before an error is declared; expressed as a multiple of 10 ms, e.g. 100 indicates 1,000 ms; the default timeout is 250 ms
Third implied	The Modbus port 3 address of the first of the receiving devices; address range: 1 ... 255 (0 = no transaction requested)
Fourth implied	The Modbus port 3 address of the second of the receiving devices; address range: 1 ... 255 (0 = no transaction requested)
. . .	. . .
18th implied	The Modbus port 3 address of the 16th of the receiving devices (address range: 1 ... 255)

PID2: Proportional Integral Derivative

121

At a Glance

Introduction

This chapter describes the instruction PID2.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

The PID2 instruction implements an algorithm that performs proportional-integral-derivative operations. The algorithm tunes the closed loop operation in a manner similar to traditional pneumatic and analog electronic loop controllers. It uses a rate gain limiting (RGL) filter on the PV as it is used for the derivative term only, thereby filtering out higher-frequency PV noise sources (random and process generated).

Formula

Proportional Control

$$M_V = K_1 E + \text{bias}$$

Proportional-Integral Control

$$M_V = K_1\left(E + K_2\int_0^t E\Delta t\right)$$

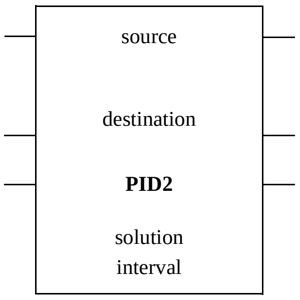
Proportional-Integral-Derivative Control

$$M_V = K_1\left(E + K_2\int_0^t E\Delta t + K_3\frac{\Delta PV}{\Delta t}\right)$$

Representation

Symbol

Representation of the instruction



Parameter Description

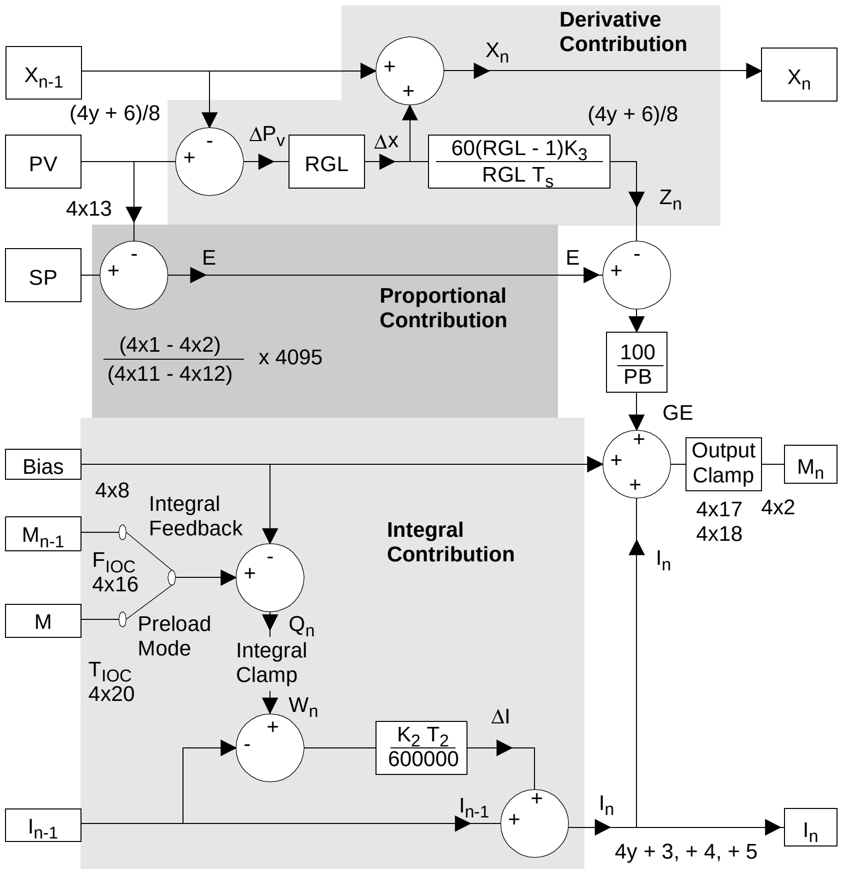
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	0 = Manual mode 1 = Auto mode
Middle input	0x, 1x	None	0 = Integral preload OFF 1 = Integral preload ON
Bottom input	0x, 1x	None	0 = Output increases as E increases 1 = Output decreases as E decreases
source (top node)	4x	INT, UINT	First of 21 contiguous holding registers in a source block
destination (middle node)	4x	INT, UINT	First of nine contiguous holding registers used for PID2 calculation. Do not load anything in these registers!
solution interval (bottom node)		INT, UINT	Contains a number ranging from 1 ... 255, indicating how often the function should be performed.
Top output	0x	None	1 = Invalid user parameter or Loop ACTIVE but not being solved
Middle output	0x	None	1 = PV ≥ high alarm limit
Bottom output	0x	None	1 = PV ≤ low alarm limit

Detailed Description

Block Diagram

Block Diagram



The elements in the block diagram have the following meaning:

Element	Meaning
E	Error, expressed in raw analog units
SP	Set point, in the range 0 ... 4095
PV	Process variable, in the range 0 ... 4095
x	Filtered PV
K2	Integral mode gain constant, expressed in 0.01 min^{-1}

Element	Meaning
K3	Derivative mode gain constant, expressed in hundredths of a minute
RGL	Rate gain limiting filter constant, in the range 2 ... 30
Ts	Solution time, expressed in hundredths of a second
PB	Proportional band, in the range 5 ... 500%
bias	Loop output bias factor, in the range 0 ... 4095
M	Loop output
GE	Gross error, the proportional-derivative contribution to the loop output
Z	Derivative mode contribution to GE
Qn	Unbiased loop output
F	Feedback value, in the range 0 ... 4095
I	Integral mode contribution to the loop output
I_{low}	Anti-reset-windup low SP, in the range 0 ... 4095
I_{high}	Anti-reset-windup high SP, in the range 0 ... 4095
K1	100/PB

Note: The integral mode contribution calculation actually integrates the difference of the output and the integral sum, this is effectively the same as integrating the error.

Proportional Control

With proportional-only control (P), you can calculate the manipulated variable by multiplying error by a proportional constant, K1, then adding a bias, see *Formula, p. 596*.

However, process conditions in most applications are changed by other system variables so that the bias does not remain constant; the result is offset error, where PV is constantly offset from the SP. This condition limits the capability of proportional-only control.

Note: The value in the integral term (in registers 4y + 3, 4y + 4, and 4y + 5) is always used, even when the integral mode is not enabled. Using this value is necessary to preserve bumpless transfer between modes. If you wish to disable bumpless transfer, these three registers must be cleared.

Proportional-Integral Control

To eliminate this offset error without forcing you to manually change the bias, an integral function can be added to the control equation, see *Formula, p. 596*. Proportional-integral control (PI) eliminates offset by integrating E as a function of time. K1 is the integral constant expressed as rep/min. As long as $E \neq 0$, the integrator increases (or decreases) its value, adjusting Mv. This continues until the offset error is eliminated.

Proportional-Integral-Derivative Control

You may want to add derivative functionality to the control equation to minimize the effects of frequent load changes or to override the integral function in order to get to the SP condition more quickly, see *Formula, p. 596*.

Proportional-integral-derivative (PID) control can be used to save energy in the process or as a safety valve in the event of a sudden, unexpected change in process flow. K3 is the derivative time constant expressed as min. DPV is the change in the process variable over a time period of Δt.

Example

An example to PID2 level control you will find in *PID2 Level Control Example, p. 25*.

Parameter Description

Source Block (Top Node)

The 4x register entered in the top node is the first of 21 contiguous holding registers in a source block. The contents of the fifth ... eighth implied registers determine whether the operation will be P, PI, or PID:

Operation	Fifth Implied	Sixth Implied	Seventh Implied	Eighth Implied
P	ON			ON
PI	ON	ON		
PID	ON	ON	ON	

The source block comprises the following register assignments:

Register	Name	Content
Displayed	Scaled PV	Loaded by the block each time it is scanned; a linear scaling is done on register 4x + 13 using the high and low ranges from registers 4x + 11 and 4x + 12: Scaled PV = (4x13 / 4095) * (4x11 - 4x12) + 4x12
First implied	SP	You must specify the set point in engineering units; the value must be < value in the 11th implied register and > value in the 12th implied register
Second implied	Mv	Loaded by the block every time the loop is solved; it is clamped to a range of 0 ... 4095, making the output compatible with an analog output module; the manipulated variable register may be used for further CPU calculations such as cascaded loops
Third implied	High Alarm Limit	Load a value in this register to specify a high alarm for PV (at or above SP); enter the value in engineering units within the range specified in the 11th and 12th implied registers
Fourth implied	Low Alarm Limit	Load a value in this register to specify a low alarm for PV (at or below SP); enter the value in engineering units within the range specified in the 11th and 12th implied registers
Fifth implied	Proportional Band	Load this register with the desired proportional constant in the range 5 ... 500; the smaller the number, the larger the proportional contribution; a valid number is required in this register for PID2 to operate

Register	Name	Content
Sixth implied	Reset Time Constant	Load this register to add integral action to the calculation; enter a value between 0000 ... 9999 to represent a range of 00.00 ... 99.99 repeats/min; the larger the number, the larger the integral contribution; a value > 9999 stops the PID2 calculation
Seventh implied	Rate Time Constant	Load this register to add derivative action to the calculation; enter a value between 0000 ... 9999 to represent a range of 00.00 ... 99.99 min; the larger the number, the larger the derivative contribution; a value > 9999 stops the PID2 calculation
Eighth implied	Bias	Load this register to add a bias to the output; the value must be between 000 .... 4095, and added directly to Mv, whether the integral term is enabled or not
Ninth implied	High Integral Windup Limit	Load this register with the upper limit of the output value (between 0 ... 4095) where the anti-reset windup takes effect; the updating of the integral sum is stopped if it goes above this value (this is normally 4095)
10th implied	Low Integral Windup Limit	Load this register with the lower limit of the output value (between 0 ... 4095) where the anti-reset windup takes effect (this is normally 0)
11th implied	High Engineering Range	Load this register with the highest value for which the measurement device is spanned, e.g. if a resistance temperature device ranges from 0 ... 500 degrees C, the high engineering range value is 500; the range must be given as a positive integer between 0001 ... 9999, corresponding to the raw analog input 4095
12th implied	Low Engineering Range	Load this register with the lowest value for which the measurement device is spanned; the range must be given as a positive integer between 0 ... 9998, and it must be less than the value in the 11th implied register; it corresponds to the raw analog input 0
13th implied	Raw Analog Measurement	The logic program loads this register with PV; the measurement must be scaled and linear in the range 0 ... 4095

Register	Name	Content
14th implied	Pointer to Loop Counter Register	The value you load in this register points to the register that counts the number of loops solved in each scan; the entry is determined by discarding the most significant digit in the register where the controller will count the loops solved/scan, e.g., if the PLC does the count in register 41236, load 1236 into the 14th implied register; the same value must be loaded into the 14th implied register in every PID2 block in the logic program
15th implied	Maximum Number of Loops	Solved In a Scan: If the 14th implied register contains a non-zero value, you may load a value in this register to limit the number of loops to be solved in one scan
16th implied	Pointer To Reset Feedback Input:	The value you load in this register points to the holding register that contains the value of feedback (F); drop the 4 from the feedback register and enter the remaining four digits in this register; integration calculations depend on the F value being should F vary from 0 ... 4095
17th implied	Output Clamp - High	The value entered in this register determines the upper limit of Mv (this is normally 4095)
18th implied	Output Clamp - Low	The value entered in this register determines the lower limit of Mv (this is normally 0)
19th implied	Rate Gain Limit (RGL) Constant	The value entered in this register determines the effective degree of derivative filtering; the range is from 2 ... 30; the smaller the value, the more filtering takes place
20th implied	Pointer to Integral Preload	The value entered in this register points to the holding register containing the track input (T) value; drop the 4 from the tracking register and enter the remaining four digits in this register; the value in the T register is connected to the input of the integral lag whenever the auto bit and integral preload bit are both true

Destination (MIddle Node)

The 4y register entered in the middle node is the first of nine contiguous holding register used for PID2 calculations. You do not need to load anything into these registers:

Register	**Name**	**Content**
Displayed	Loop Status Register	Twelve of the 16 bits in this register are used to define loop status.
First implied	Error (E) Status Bits	This register displays PID2 error codes.
Second implied	Loop Timer Register	This register stores the real-time clock reading on the system clock each time the loop is solved: the difference between the current clock value and the value stored in the register is the elapsed time; if elapsed time ≥ solution interval (10 times the value given in the bottom node of the PID2 block), then the loop should be solved in this scan
Third implied	For Internal Use	Integral (integer portion)
Fourth implied	For Internal Use	Integral-fraction 1 (1/3 000)
Fifth implied	For Internal Use	Integral-fraction 2 (1/600 000)
Sixth implied	Pv x 8 (Filtered)	This register stores the result of the filtered analog input (from register 4x14) multiplied by 8; this value is useful in derivative control operations
Seventh implied	Absolute Value of E	This register, which is updated after each loop solution, contains the absolute value of (SP - PV); bit 8 in register 4y + 1 indicates the sign of E
Eighth implied	For Internal Use	Current solution interval

Loop Status Register

LOOP Status

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

Bit	Function
1	Top output status (Node lockout or parameter error
2	Middle output status (High alarm)
3	Bottom output status (Low alarm)
4	Loop in AUTO mode and time since last solution ≥ solution interval
5	Wind-down mod (for REV B or higher)
6	Loop in AUTO mode but not being solved
7	4x14 register referenced by 4x15 is valid
8	Sign of E in 4y + 7: ● 0 = + (plus) ● 1 = - (minus)
9	Rev B or higher
10	Integral windup limit never set
11	Integral windup saturated
12	Negative values in the equation
13	Bottom input status (direct / reverse acting)
14	Middle input status (tracking mode) ● 1 = tracking ● 0 = no tracking
15	Top input status (MAN / AUTO)
16	Bit 16 is set after initial startup or installation of the loop. If you clear the bit, the following actions take place in one scan: ● The loop status register 4y is reset ● The current value in the real-time clock is stored in the first implied register (4y+1) ● Values in the third ... fifth registers (4y+2,3) are cleared ● The value in the13th implied register (4x+13) x 8 is stored in the sixth implied register (4y+6) ● The seventh and eighth implied registers (4y+7,8) are cleared

Solution Interval (Bottom Node)

The bottom node indicates that this is a PID2 function and contains a number ranging from 1 ... 255, indicating how often the function should be performed. The number represents a time value in tenths of a second, or example, the number 17 indicates that the PID function should be performed every 1.7 s.

Run Time Errors

Error Status Bit

The first implied register of the destination contains the error status bits:

Code	Explanation	Check these Registers in the Source Block (Top Node)
0000	No errors, all validations OK	None
0001	Scaled SP above 9999	First implied
0002	High alarm above 9999	Third implied
0003	Low alarm above 9999	Fourth implied
0004	Proportional band below 5	Fifth implied
0005	Proportional band above 500	Fifth implied
0006	Reset above 99.99 r/min	Sixth implied
0007	Rate above 99.99 min	Seventh implied
0008	Bias above 4095	Eighth implied
0009	High integral limit above 4095	Ninth implied
0010	Low integral limit above 4095	10th implied
0011	High engineering unit (E.U.) scale above 9999	11th implied
0012	Low E.U. scale above 9999	12th implied
0013	High E.U. below low E.U.	11th and 12th implied
0014	Scaled SP above high E.U.	First and 11th implied
0015	.Scaled SP below low E.U.	First and 12th implied
0016	Maximum loops/scan > 9999 **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero.	15th implied
0017	Reset feedback pointer out of range	16th implied
0018	High output clamp above 4095	17th implied
0019	Low output clamp above 4095	18th implied
0020	Low output clamp above high output clamp	17th and 18th implied
0021	RGL below 2	19th implied
0022	RGL above 30	19th implied
0023	Track F pointer out of range **Note**: Activated only if the track feature is ON, i.e. the middle input of the PID2 block is receiving power while in AUTO mode.	20th implied with middle input ON

Code	Explanation	Check these Registers in the Source Block (Top Node)
0024	Track F pointer is zero **Note**: Activated only if the track feature is ON, i.e. the middle input of the PID2 block is receiving power while in AUTO mode.	20th implied with middle input ON
0025	Node locked out (short of scan time) **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero. **Note:** If lockout occurs often and the parameters are all valid, increase the maximum number of loops/scan. Lockout may also occur if the counting registers in use are not cleared as required.	None
0026	Loop counter pointer is zero **Note:** Activated by maximum loop feature, i.e. only if 4x15 is not zero.	14th and 15th implied
0027	Loop counter pointer out of range	14th and 15th implied

R ––> T: Register to Table

122

At a Glance

Introduction

This chapter describes the instruction R → T.

What's in this chapter?

This chapter contains the following topics:

Short Description

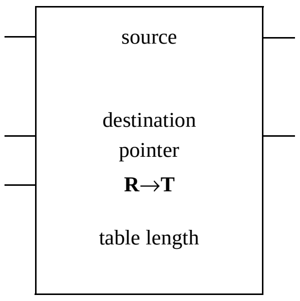
Function Description

The R→T instruction copies the bit pattern of a register or of a string of contiguous discretes stored in a word into a specific register located in a table. It can accommodate the transfer of one register/word per scan.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = copies source data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	Source data to be copied in the current scan
destination pointer (middle node)	4x	INT, UINT	Destination table where source data will be copied in the scan
table length (bottom node)		INT, UINT	Number of registers in the destination table, range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Top Input

The input to the top node initiates the DX move operation.

Middle Input

When the middle input goes ON, the current value stored in the destination pointer register is frozen while the DX operation continues. This causes new data being copied to the destination to overwrite the data copied on the previous scan.

Bottom Input

When the bottom input goes ON, the value in the destination pointer register is reset to zero. This causes the next DX move operation to copy source data into the first register in the destination table.

Source Data (Top Node)

When using register types 0x or 1x:
- First 0x reference in a string of 16 contiguous coils or discrete outputs
- First 1x reference in a string of 16 discrete inputs

Destination Pointer (Middle Node)

The 4x register entered in the middle node is a pointer to the destination table where source data will be copied in the scan. The first register in the destination table is the next contiguous 4x register following the pointer, i.e. if the pointer register is 400027, then the destination table begins at register 400028.

The value posted in the pointer register indicates the register in the destination table where the source data will be copied. A value of zero indicates that the source data will be copied to the first register in the destination table; a value of 1 indicates that the source data be copied to the second register in the destination table; etc.

Note: The value posted in the destination pointer register cannot be larger than the table length integer specified in this node.

Outputs

R→T can produce two possible outputs, from the top and middle nodes. The state of the output from the top node echoes the state of the top input. The output from the middle node goes ON when the value in the destination pointer register equals the specified table length. At this point, the instruction cannot increment any further.

RBIT: Reset Bit

123

At a Glance

Introduction

This chapter describes the instruction RBIT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

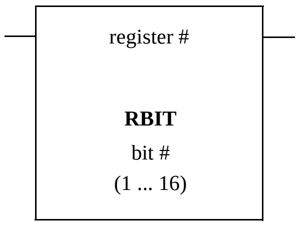
The reset bit (RBIT) instruction lets you clear a latched-ON bit by powering the top input. The bit remains cleared after power is removed from the input. This instruction is designed to clear a bit set by the SBIT instruction.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = clears the specified bit to 0. The bit remains cleared after power is removed from the input
register # (top node)	4x	WORD	Holding register whose bit pattern is being controlled
bit # (bottom node)		INT, UINT	Indicates which one of the 16 bits is beeing cleared
Top output	0x	None	ON = the specified bit has been cleared to 0

READ: Read

124

At a Glance

Introduction

This chapter describes the instruction READ.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

The READ instruction provides the ability to read data from an ASCII input device (keyboard, bar code reader, etc.) into the PLC’s memory via its RIO network. The connection to the ASCII device is made at an RIO interface.

In the process of handling the messaging operation, READ performs the following functions:
- Verifies the lengths of variable data fields
- Verifies the correctness of the ASCII communication parameters, e.g. the port number, the message number
- Performs error detection and recording
- Reports RIO interface status

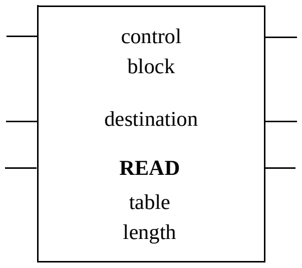
READ requires two tables of registers: a destination table where retrieved variable data (the message) is stored, and a control block where comm port and message parameters are identified.

Further information about formatting messages you will find in *Formatting Messages for ASCII READ/WRIT Operations, p. 29*.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = initiates a READ
Middle input	0x, 1x	None	ON = pauses READ operation
Bottom input	0x, 1x	None	ON = abort READ operation
control block (top node)	4x	INT, UINT, WORD	Control block (first of seven contiguous holding registers)
destination (middle node)	4x	INT, UINT, WORD	Destination table
table length (bottom node)		INT, UINT	Length of destination table (number of registers where the message data will be stored), range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = error in communication or operation has timed out (for one scan)
Bottom output	0x	None	ON = READ complete (for one scan)

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of seven contiguous holding register in the control block.

Register	Definition
Displayed	*Port Number and Error Code, p. 618*
First implied	Message number
Second implied	Number of registers required to satisfy format
Third implied	Count of the number of registers transmitted thus far
Fourth implied	Status of the solve
Fifth implied	Reserved
Sixth implied	Checksum of registers 0 ... 5

Port Number and Error Code

Port Number and Error Code

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

Bit	Function
1 ... 4	PLC error code
5	Not used
6	Input from the ASCII device not compatible with format
7	Input buffer overrun, data received too quickly at RIOP
8	USART error, bad byte received at RIOP
9	Illegal format, not received properly by RIOP
10	ASCII device off-line, check cabling
11	ASCII message terminated early (in keyboard mode
12 ... 16	Comm port # (1 ... 32)

PLC Error Code

Bit				Meaning
1	**2**	**3**	**4**	
0	0	0	1	Error in the input to RIOP from ASCII device
0	0	1	0	Exception response from RIOP, bad data
0	0	1	1	Sequenced number from RIOP differs from expected value
0	1	0	0	User register checksum error, often caused by altering READ registers while the block is active
0	1	0	1	Invalid port or message number detected
0	1	1	0	User-initiated abort, bottom input energized
0	1	1	1	No response from drop, communication error
1	0	0	0	Node aborted because of SKP instruction
1	0	0	1	Message area scrambled, reload memory
1	0	1	0	Port not configured in the I/O map
1	0	1	2	Illegal ASCII request
1	1	0	0	Unknown response from ASCII port
1	1	0	1	Illegal ASCII element detected in user logic
1	1	1	1	RIOP in the PLC is down

Destination (Middle Node)

The middle node contains the first 4x register in a destination table. Variable data in a READ message are written into this table. The length of the table is defined in the bottom node.

Consider this READ message:

please enter password**:**	**AAAAAAAAAA**
(Embedded Text)	(Variable Data)

Note: An ASCII READ message may contain the embedded text, placed inside quotation marks, as well as the variable data in the format statement, i.e., the ASCII message.
The 10-character ASCII field **AAAAAAAAAA** is the variable data field; variable data must be entered via an ASCII input device.

RET: Return from a Subroutine

125

At a Glance

Introduction

This chapter describes the instruction RET.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

The RET instruction may be used to conditionally return the logic scan to the node immediately following the most recently executed JSR block. This instruction can be implemented only from within the subroutine segment, the (unscheduled) last segment in the user logic program.

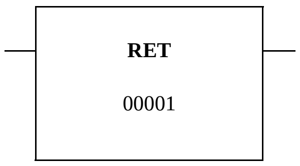
Note: If a subroutine does not contain a RET block, either a LAB block or the end-of-logic (whichever comes first) serves as the default return from the subroutine.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = return to previous logic
00001		INT, UINT	Constant value, can not be changed
Top output	0x	None	ON = error in the specified subroutine

SAVE: Save Flash

126

At a Glance

Introduction

This chapter describes the instruction SAVE.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

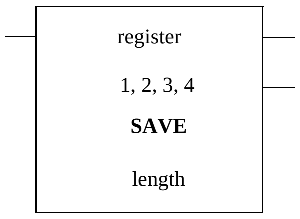
Note: This instruction is available with the PLC family TSX Compact, with Quantum CPUs 434 12/ 534 14 and Momentum CPUs CCC 960 x0/ 980 x0.

The SAVE instruction saves a block of 4x registers to state RAM where they are protected from unauthorized modification.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Start SAVE operation: it should remain ON until the operation has completed successfully or an error has occurred.
register (top node)	4x	INT, UINT, WORD	First of max. 512 contiguous 4x registers to be saved to state RAM
1, 2, 3, 4 (middle node)		INT	Integer value, which defines the specific buffer where the block of data is to be saved
length (bottom node)		INT	Number of words to be saved, range: 1 ... 512
Top output	0x	None	ON = SAVE is active
Middle output	0x	None	ON = SAVE is not allowed

Parameter Description

1, 2, 3, 4 (Middle Node)	The middle node defines the specific buffer, within state RAM, where the block of data is to be saved. Four 512 word buffers are allowed. Each buffer is defined by placing its corresponding value in the middle node, that is, the value 1 represents the first buffer, value 2 represents the second buffer and so on. The legal values are 1, 2, 3, and 4. When the PLC is started all four buffers are zeroed. Therefore, you may not save data to the same buffer without first loading it with the instruction LOAD. When this is attempted the middle output goes ON. In other words, once a buffer is used, it may not be used again until the data has been removed.
Middle Output	The output from the middle node goes ON when previously saved data has not been accessed using the LOAD instruction. This prevents inadvertent overwriting of data in the SAVE buffer.

SBIT: Set Bit

127

At a Glance

Introduction

This chapter describes the instruction SBIT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

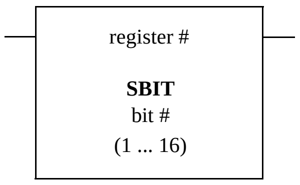
The set bit (SBIT) instruction lets you set the state of the specified bit to ON (1) by powering the top input.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = sets the specified bit to 1. The bit remains set after power is removed from the input
register # (top node)	4x	WORD	Holding register whose bit pattern is being controlled
bit # (bottom node)		INT, UINT	Indicates which one of the 16 bits is being set
Top output	0x	None	Goes ON, when the specified bit is set and remains ON until it is cleared (via the RBIT instruction)

SCIF: Sequential Control Interfaces

128

At a Glance

Introduction

This chapter describes the instruction SCIF.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

The SCIF instruction performs either a drum sequencing operation or an input comparison (ICMP) using the data defined in the step data table.

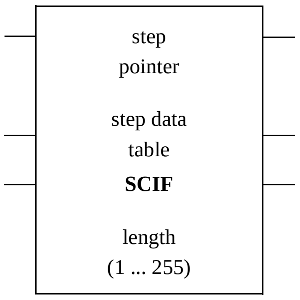
The choice of operation is made by defining the value in the first register of the step data table:

- **0 = drum mode:**
  The instruction controls outputs in the drum sequencing application.
- **1 = ICMP mode:**
  The instruction reads inputs to ensure that limit switches, proximity switches, pushbuttons, etc. are properly positioned to allow drum outputs to be fired.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates specified sequence control operation
Middle input	0x, 1x	None	**Drum mode:** step pointer increments to the next step **ICMP mode:** compare status is shown at the middle output
Bottom input	0x, 1x	None	**Drum mode:** ON = reset step pointer to 0 **ICMP mode:** not used
step pointer (top node)	4x	INT, UINT	Number of the current step in the step data table
step data table (middle node)	4x	INT, UINT	First register in the step data table
length (bottom node)		INT, UINT	Number of application-specific registers used in the step data table
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	**Drum mode**: step pointer = length **ICMP mode**: indicates a valid input comparison
Bottom output	0x	None	ON = error is detected

Parameter Description

Step Data Table (Middle Node)

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

Register	Register Name	Description
Displayed	subfunction type	0 = drum mode; 1 = ICMP mode (entry of any other value in this register will result in all outputs OFF)
First implied	masked output data (in drum mode)	Loaded by SCIF each time the block is solved; the register contains the contents of the current step data register masked with the output mask register
	raw input data (in ICMP mode)	Loaded by the user from a group of sequential inputs to be used by the block in the current step
Second implied	current step data	Loaded by SCIF each time the block is solved; the register contains data from the current step (pointed to by the step pointer)
Third implied	output mask (in drum mode)	Loaded by the user before using the block, the contents will not be altered during logic solving; contains a mask to be applied to the data for each sequencer step
	input mask (in ICMP mode)	Loaded by the user before using the block, it contains a mask to be ANDed with raw input data for each step, masked bits will not be compared; the masked data are put in the masked input data register
Fourth implied	masked input data (in ICMP mode)	Loaded by SCIF each time the block is solved, it contains the result of the ANDed input mask and raw input data
	not used in drum mode	
Fifth implied	compare status (in ICMP mode)	Loaded by SCIF each time the block is solved, it contains the result of an XOR of the masked input data and the current step data; unmasked inputs that are not in the correct logical state cause the associated register bit to go to 1, non-zero bits cause a miscompare and turn ON the middle output from the SCIF block
	not used in drum mode	
Sixth implied	start of data table	First of K registers in the table containing the user-specified control data **Note:** This and the rest of the registers represent application-specific step data in the process being controlled.

Length of Step Data Table (Bottom Node)

The integer value entered in the bottom node is the length, i.e. the number of application-specific registers, used in the step data table. The length can range from 1 ... 255.

The total number of registers required in the step data table is the length + 7. The length must be ≥ the value placed in the steps used register in the middle node.

SENS: Sense

129

At a Glance

Introduction

This chapter describes the instruction SENS.

What's in this chapter?

This chapter contains the following topics:

Short Description

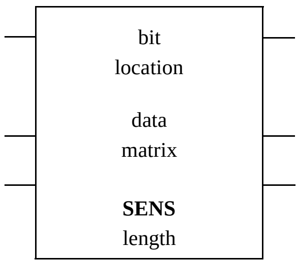
Function Description

The SENS instruction examines and reports the sense (1 or 0) of a specific bit location in a data matrix. One bit location is sensed per scan.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = senses the bit location
Middle input	0x, 1x	None	Increment bit location by one on next scan
Bottom input	0x, 1x	None	Reset bit location to 1
bit location (top node)	3x, 4x	WORD	Specific bit location to be sensed in the data matrix, entered explicitly as an integer or stored in a register; range: 1 ... 9 600
data matrix (middle node)	0x, 4x	BOOL, WORD	First word or register in the data matrix
length (bottom node)		INT, UINT	Matrix length; range 1 ... 600
Top output	0x	None	Echoes state of the top input

Middle output	0x	None	ON = bit sense is 1
			OFF = bit sense is 0

Bottom output	0x	None	ON = error: bit location > matrix length

Parameter Description

Bit Location (Top Node)

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

Matrix Length (Bottom Node)

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

SKPC: Skip (Constants)

130

At a Glance

Introduction

This chapter describes the instruction SKPC.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

When a SKPC instruction is implemented, skipped networks in the ladder logic program are not solved. SKPC instructions can be used to reduce scan time and, in effect, establish subroutines within the scheduled logic.
A SKPC operation cannot pass the boundary of a segment. No matter how many extra networks you specify to be skipped, the instruction will stop if it reaches the end of a segment.

Note: A SKPC instruction can be activated only if you specify in the configurator editor that skips are allowed.

WARNING

Inputs and outputs could be unintentionally skipped or not skipped.

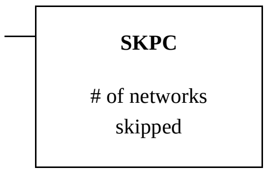
SKPC is a dangerous instruction that should be used carefully. If inputs and outputs that normally effect control are unintentionally skipped (or not skipped), the result can create hazardous conditions for personnel and application equipment.

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = skip operation is performed on every scan
# of networks skipped (bottom node)	3x, 4x	INT	Number of networks to be skipped, specified explicitly as an integer constant (range: 1 ... 999) or stored in a register

Parameter Description

Number of Networks skipped (Bottom Node)

The value entered in the node specifies the number of networks to be skipped.

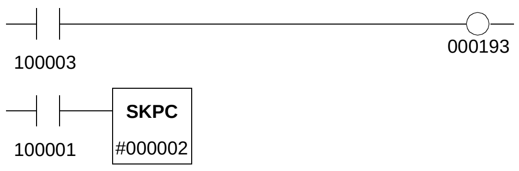
The node value includes the network that contains the SKPC instruction. The nodal regions in the network where the SKPC resides that have not already been scanned will be skipped; this counts as one of the networks specified to be skipped. The CPU continues to skip networks until the total number of networks skipped equals the value specified.

Example

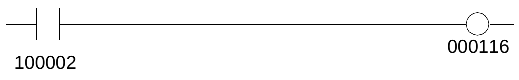
A simple SKPC Example

The illustration is showing two contiguous networks of ladder logic. The first network contains a SKPC instruction that specifies that two networks will be skipped when contact 100001 passes power.

Network 1



Network 2



When N.O. contact 100001 is closed, the remainder of the top network and all of the bottom network are skipped. The power flow display for these two networks becomes invalid, and your system displays an information message to that effect.

Coil 000193 is still controlled by contact 100003 because the solution of coil 000193 occurs before the SKPC instruction. Coil 000116 will remain in whatever state it was in when the bottom network was skipped.

SKPR: Skip (Registers)

131

At a Glance

Introduction

This chapter describes the instruction SKPR.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

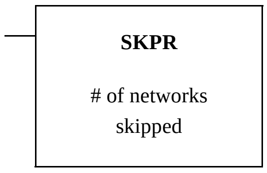
When a SKPR instruction is implemented, skipped networks in the ladder logic program are not solved. SKPR instructions can be used to reduce scan time and, in effect, establish subroutines within the scheduled logic.
A SKPR operation cannot pass the boundary of a segment. No matter how many extra networks you specify to be skipped, the instruction will stop if it reaches the end of a segment.

	WARNING
STOP	**Inputs and outputs could be unintentionally skipped or not skipped.** SKPR is a dangerous instruction that should be used carefully. If inputs and outputs that normally effect control are unintentionally skipped (or not skipped), the result can create hazardous conditions for personnel and application equipment. **Failure to observe this precaution can result in severe injury or equipment damage.**

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = skip operation is performed on every scan
# of networks skipped (bottom node)	3x, 4x	INT	Number of networks to be skipped, specified explicitly as an integer constant (range: 1 ... 999) or stored in a register

Parameter Description

Number of Networks skipped (Bottom Node)

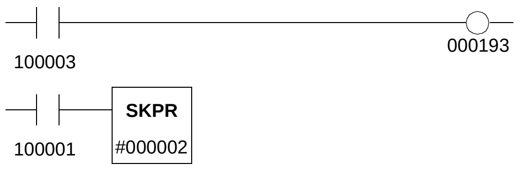
The value entered in the node specifies the number of networks to be skipped.

The node value includes the network that contains the SKPR instruction. The nodal regions in the network where the SKPR resides that have not already been scanned will be skipped; this counts as one of the networks specified to be skipped. The CPU continues to skip networks until the total number of networks skipped equals the value specified.

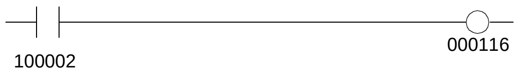
Example

A simple SKPR Example

The illustration is showing two contiguous networks of ladder logic. The first network contains a SKPR instruction that specifies that two networks will be skipped when contact 100001 passes power.
Network 1:



Network 1:



When N.O. contact 100001 is closed, the remainder of the top network and all of the bottom network are skipped. The power flow display for these two networks becomes invalid, and your system displays an information message to that effect.

Coil 000193 is still controlled by contact 100003 because the solution of coil 000193 occurs before the SKPR instruction. Coil 000116 will remain in whatever state it was in when the bottom network was skipped.

SRCH: Search

132

At a Glance

Introduction

This chapter describes the instruction SRCH.

What's in this chapter?

This chapter contains the following topics:

Short Description

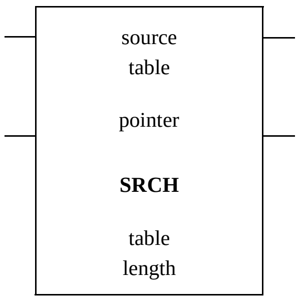
Function Description

The SRCH instruction searches the registers in a source table for a specific bit pattern.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates search
Middle input	0x, 1x	None	OFF = search from beginning ON = search from last match
source table (top node)	3x, 4x	INT, UINT, WORD	Source table to be searched
pointer (middle node)	4x	INT, UINT	Pointer into the source table
table length (bottom node)		INT, UINT	Number of registers in the source table; range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = match found

Parameter Description

Pointer (Middle Node)	The 4x register entered in the middle node is the pointer into the source table. It points to the source register that contains the same value as the value stored in the next contiguous register after the pointer, e.g. if the pointer register is 400015, then register 400016 contains a value that the SRCH instruction will attempt to match in source table.

STAT: Status

133

At a Glance

Introduction

This chapter describes the instruction STAT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

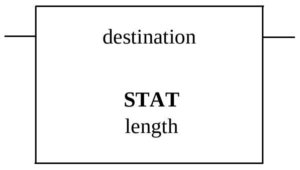
The STAT instruction accesses a specified number of words in a status table in the PLC's system memory. Here vital diagnostic information regarding the health of the PLC and its remote I/O drops is posted.

This information includes:
- PLC status
- Possible error conditions in the I/O modules
- Input-to-PLC-to-output communication status

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = copies specified number of words from the status table
destination (top node)	0x, 4x	INT, UINT, BOOL, WORD	First position in the destination block
length (bottom node)		INT, UINT	number of registers or 16-bit words in the destination block
Top output	0x	None	ON = operation successful

Parameter Description

Mode of Functioning

With the STAT instruction, you can copy some or all of the status words into a block of registers or a block of contiguous discrete references.

The copy to the STAT block always begins with the first word in the table up to the last word of interest to you. For example, if the status table is 277 words long and you are interested only in the statistics provided in word 11, you need to copy only words 1 ... 11 by specifying a length of 11 in the STAT instruction.

Destination Block (Top Node)

The reference number entered in the top node is the first position in the destination block, i.e. the block where the current words of interest from the status table will be copied.

The number of holding registers or 16-bit words in the destination block is specified in the bottom node (length).

Note: We recommend that you do not use discretes in the STAT destination node because of the excessive number required to contain status information.

Length (Bottom Node)

The integer value entered in the bottom node specifies the number of registers or 16-bit words in the destination block where the current status information will be written. The maximum allowable length the Quantum PLCs with S908 RIO protocol is 1 ... 277.

Description of the Status Table

General

The STAT instruction is used to display the Status of Controller and I/O system for Quantum, Atrium, TSX Compact and Momentum.

The first 11 status words are used by Quantum and Momentum in the same way and by TSX Compact and Atrium in the same way. The following have a different meaning for Quantum, TSX Compact and Momentum.

Quantum Overview

The 277 words in the status table are organized in three sections:
- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 171)
- I/O Communications Health (words 172 ... 277)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	Controller Status	01
2	Hot Standby Status	02
3	Controller Status	03
4	RIO Status	04
5	Controller Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	RIO Redundancy and Timeout	08
9	ASCII Message Status	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	Drop 1, Rack 1	0C
13	Drop 1, Rack 2	0D
. . .	. . . . . .	. . .
16	Drop 1, Rack 5	0F
17	Drop 2, Rack 1	10
18	Drop 2, Rack 2	11
. . .	. . . . . .	. . .
171	Drop 32, Rack 5	AB
172	S908 Startup Error Code	AC
173	Cable A Errors	AD
174	Cable A Errors	AE
175	Cable A Errors	AF
176	Cable B Errors	B0
178	Cable B Errors	B1
178	Cable B Errors	B2
179	Global Communication Errors	B3
180	Global Communication Errors	B4
181	Global Communication Errors	B5

Decimal Word	Word Content	Hex Word
182	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	B6
183	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Second word)	B7
184	Drop 1 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Third word)	B8
185	Drop 2 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	B9
. . .	. . . . . .	. . .
275	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (First word)	113
276	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Second word)	114
277	Drop 32 Errors/Health Status and Retry Counters (in the TSX Compact 984 Controllers) (Third word)	115

Momentum Overview

The 20 words in the status table are organized in two sections:
- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 20)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	Controller Status	01
2	Hot Standby Status	02
3	Controller Status	03
4	RIO Status	04
5	Controller Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	RIO Redundancy and Timeout	08
9	ASCII Message Status	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	Local Momentum I/O Module Health	0C
13	I/O Bus Module Health	0D
14	I/O Bus Module Health	0E
15	I/O Bus Module Health	0F
16	I/O Bus Module Health	10
17	I/O Bus Module Health	11
18	I/O Bus Module Health	12
19	I/O Bus Module Health	13
20	I/O Bus Module Health	14

TSX Compact and Atrium Overview

The 184 words in the status table are organized in three sections:
- Controller Status (words 1 ... 11)
- I/O Module Health (words 12 ... 15)
- Not used (16 ... 181)
- Global Health and Communications retry status (words 182 ... 184)

Words of the status table:

Decimal Word	Word Content	Hex Word
1	CPU Status	01
2	not used	02
3	Controller Status	03
4	not used	04
5	CPU Stop State	06
6	Number of Ladder Logic Segments	06
7	End-of-logic (EOL) Pointer	07
8	not used	08
9	not used	09
10	RUN/LOAD/DEBUG Status	0A
11	not used	0B
12	I/O Health Status Rack 1	0C
13	I/O Health Status Rack 2	0D
14	I/O Health Status Rack 3	0E
15	I/O Health Status Rack 4	0F
16 ... 181	not used	10 ... B5
182	Health Status	B6
183	I/O Error Counter	B7
184	PAB Bus Retry Counter	B8

Controller Status Words 1 - 11 for Quantum and Momentum

Controller Status (Word 1)

Word 1 displays the following aspects of the PLC status:

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

Bit	Function
1 - 5	Not used
6	1 = enable constant sweep
7	1 = enable single sweep delay
8	1 = 16 bit user logic 0 = 24 bit user logic
9	1 = AC power on
10	1 = RUN light OFF
11	1 = memory protect OFF
12	1 = battery failed
13 - 16	Not used

Hot Standby Status (Word 2)

Word 2 displays the Hot Standby status for 984 PLCs that use S911/R911 Hot Standby Modules:

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

Bit	Function
1	1 = S911/R911 present and healthy
2 - 10	Not used
11	0 = controller toggle set to A 1 = controller toggle set to B
12	0 = controllers have matching logic 1 = controllers do not have matching logic
13, 14	Remote system state: 0 1 = Off line (1 dec) 1 0 = primary (2 dec) 1 1 = standby (3 dec)
15, 16	Local system state: 0 1 = Off line (1 dec) 1 0 = primary (2 dec) 1 1 = standby (3 dec)

Controller Status (Word 3)

Word 3 displays more aspects of the controller status:

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

Bit	Function
1	1 = first scan
2	1 = start command pending
3	1 = constant sweep time exceeded
4	1 = Existing DIM AWARENESS
5 - 12	Not used
13 - 16	Single sweeps

RIO Status (Word 4)

Word 4 is used for IOP information:

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

Bit	Function
1	1 = IOP bad
2	1 = IOP time out
3	1 = IOP loop back
4	1 = IOP memory failure
5 - 12	Not used
13 - 16	00 = IO did not respond 01 = no response 02 = failed loopback

Controller Stop State (Word 5)

Word 5 displays the PLC's stop state conditions:

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

Bit	Function
1	1 = peripheral port stop
2	Extended memory parity error (for chassis mount controllers) or traffic cop/S908 error (for other controllers) If the bit = 1 in a **984B controller**, an error has been detected in extended memory; the controller will run, but the error output will be ON for XMRD/XMWT functions If the bit = 1 for **any other controller than a chassis mount**, then either a traffic cop error has been detected or the S908 is missing from a multi-drop configuration.
3	1 = controller in DIM AWARENESS
4	1 = illegal peripheral intervention
5	1 = segment scheduler invalid
6	1 = start of node did not start segment
7	1 = state RAM test failed
8	1 = invalid traffic cop
9	1 = watchdog timer expired
10	1 = real time clock error
11	CPU logic solver failed (for chassis mount controllers) or Coil Use TABLE (for other controllers) If the bit = 1 in a chassis mount controller, the internal diagnostics have detected CPU failure. If the bit = 1 in any controller other than a chassis mount, then the Coil Use Table does not match the coils in user logic.
12	1 = IOP failure
13	1 = invalid node
14	1 = logic checksum
15	1 = coil disabled in RUN mode (see **Caution** below)
16	1 = bad config

⚠	**CAUTION**
	Using a Quantum or 984-684E/785E PLC If you are using a Quantum or 984-684E/785E PLC, bit 15 in word 5 is never set. These PLCs can be started and run with coils disabled in RUN (optimized) mode. Also all the bits in word 5 must be set to 0 when one of these PLCs is running. **Failure to observe this precaution can result in injury or equipment damage.**

Controller Stop State (Word 6)

Word 6 displays the number of segments in ladder logic; a binary number is shown:

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

Bit	Function
1 - 16	Number of segments (expressed as a decimal number)

Controller Stop State (Word 7)

Word 7 displays the address of the end-of-logic (EOL) pointer:

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

Bit	Function
1 - 16	EOL pointer address

RIO Redundancy and Timeout (Word 8)

Word 8 uses its most significant bit to display whether or not redundant coaxial cables are run to the remote I/O drops, and it uses its four least significant bits to display the remote I/O timeout constant:

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

Bit	Function
1	RIO redundant cables? 0 = NO 1 = YES
2 - 12	Not used
13 - 16	RIO timeout constant

ASCII Message Status (Word 9)

Word 9 uses its four least significant bits to display ASCII message status:

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

Bit	Function
1 ... 12	Not used
13	1 = Mismatch between numbers of messages and pointers
14	1 = Invalid message pointer
15	1 = Invalid message
16	1 = Message checksum error

RUN/LOAD/ DEBUG Status (Word 10)

Word 10 uses its two least significant bits to display RUN/LOAD/DEBUG status:

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

Bit	Function
1 ... 14	Not used
15, 15	0 0 = Debug (0 dec) 0 1 = Run (1 dec) 1 0 = Load (2 dec)

Word 11

This word is not used.

I/O Module Health Status Words 12 - 20 for Momentum

I/O Module Health Status

Status words 12 ... 20 display I/O module health status.

1 word is reserved for each of up to 1 Local drop, 8 words are used to represent the health of up to 128 I/O Bus Modules

Local Momentum I/O Module Health

Word 12 displays the Local Momentum I/O Module health:

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

Bit	Function
1	1 = Local Module
2 - 16	Not used

Momentum I/O Bus Module Health

Word 13 through 20 display the health status for Momentum I/O Bus Modules as follows:

Word	I/O Bus Modules
13	1 ... 16
14	17 ... 32
15	33 ... 48
16	49 ... 64
17	65 ... 80
18	81 ... 96
19	97 ... 112
20	113 ... 128

Each Word display the Momentum I/O Bus Module health as follows:

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

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

I/O Module Health Status Words 12 - 171 for Quantum

RIO Status Words

Status words 12 ... 20 display I/O module health status.

Five words are reserved for each of up to 32 drops, one word for each of up to five possible racks (I/O housings) in each drop. Each rack may contain up to 11 I/O modules; bits 1 ... 11 in each word represent the health of the associated I/O module in each rack.

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

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

Four conditions must be met before an I/O module can indicate good health:
- The slot must be traffic copped
- The slot must contain a module with the correct personality
- Valid communications must exist between the module and the RIO interface at remote drops
- Valid communications must exist between the RIO interface at each remote drop and the I/O processor in the controller

Status Words for the MMI Operator Panels

The status of the 32 Element Pushbutton Panels and PanelMate units on an RIO network can also be monitored with an I/O health status word. The Pushbutton Panels occupy slot 4 in an I/O rack and can be monitored at bit 4 of the appropriate status word. A PanelMate on RIO occupies slot 1 in rack 1 of the drop and can be monitored at bit 1 of the first status word for the drop.

Note: The ASCII Keypad's communication status can be monitored with the error codes in the ASCII READ/WRIT blocks.

Communication Status Words 172 - 277 for Quantum

DIO Status

Status words 172 ... 277 contain the I/O system communication status. Words 172 ... 181 are global status words. Among the remaining 96 words, three words are dedicated to each of up to 32 drops, depending on the type of PLC.

Word 172 stores the Quantum Startup Error Code. This word is always 0 when the system is running. If an error occurs, the controller does not start-it generates a stop state code of 10 (word 5 (See *Controller Stop State (Word 5), p. 660*)).

Quantum Start-up Error Codes

Code	Error	Meaning (Where the error has occurred)
01	BADTCLEN	Traffic Cop length
02	BADLNKNUM	Remote I/O link number
03	BADNUMDPS	Number of drops in Traffic Cop
04	BADTCSUM	Traffic Cop checksum
10	BADDDLEN	Drop descriptor length
11	BADDRPNUM	I/O drop number
12	BADHUPTIM	Drop holdup time
13	BADASCNUM	ASCII port number
14	BADNUMODS	Number of modules in drop
15	PRECONDRP	Drop already configured
16	PRECONPRT	Port already configured
17	TOOMNYOUT	More than 1024 output points
18	TOOMNYINS	More than 1024 input points
20	BADSLTNUM	Module slot address
21	BADRCKNUM	Module rack address
22	BADOUTBC	Number of output bytes

Code	Error	Meaning (Where the error has occurred)
23	BADINBC	Number of input bytes
25	BADRF1MAP	First reference number
26	BADRF2MAP	Second reference number
27	NOBYTES	No input or output bytes
28	BADDISMAP	Discrete not on 16-bit boundary
30	BADODDOUT	Unpaired odd output module
31	BADODDIN	Unpaired odd input module
32	BADODDREF	Unmatched odd module reference
33	BAD3X1XRF	1x reference after 3x register
34	BADDMYMOD	Dummy module reference already used
35	NOT3XDMY	3x module not a dummy
36	NOT4XDMY	4x module not a dummy
40	DMYREAL1X	Dummy, then real 1x module
41	REALDMY1X	Real, then dummy 1x module
42	DMYREAL3X	Dummy, then real 3x module
43	REALDMY3X	Real, then dummy 3x module

Status of Cable A

Words 173 ... 175 are Cable A error words:

Word 173

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

Bit	Function
1 ... 8	Counts framing errors
9 ... 16	Counts DMA receiver overruns

Word 174

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

Bit	Function
1 ... 8	Counts receiver errors
9 ... 16	Counts bad drop receptions

Word 175

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

Bit	Function
1	1 = Short frame
2	1 = No end-of- frame
3 ... 12	Not used
13	1 = CRC error
14	1 = Alignment error
15	1 =Overrun error
16	Not used

Status of Cable B

Words 176 ... 178 are Cable A error words:

Word 176

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

Bit	Function
1 ... 8	Counts framing errors
9 ... 16	Counts DMA receiver overruns

Word 177

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

Bit	Function
1 ... 8	Counts receiver errors
9 -...16	Counts bad drop receptions

Word 178

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

Bit	Function
1	1 = Short frame
2	1 = No end-of- frame
3 ... 12	Not used
13	1 = CRC error
14	1 = Alignment error
15	1 =Overrun error
16	Not used

Status of Global Communication (Words 179 ... 181)

Word 179 displays global communication status:

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

Bit	Function
1	1 = Comm health
2	1 = Cable A status
3	1 = Cable B status
4	Not used
5 ... 8	Lost communication counter
9 ... 16	Cumulative retry counter

Word 180 is the global cumulative error counter for Cable A:

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

Bit	Function
1 ... 8	Counts detected errors
9 ... 162	Counts No responses

Word 181 is the global cumulative error counter for Cable B:

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

Bit	Function
1 ... 8	Counts detected errors
9 ... 162	Counts No responses

Status of Remote I/O (Words 182 ... 277)

Words 182 ... 277 are used to describe remote I/O drop status; three status words are used for each drop.

The first word in each group of three displays communication status for the appropriate drop:

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

Bit	Function
1	1 = Communication health
2	1 = Cable A status
3	1 = Cable B status
4	Not used
5 ... 8	Lost communication counter
9 ... 16	Cumulative retry counter

The second word in each group of three is the drop cumulative error counter on Cable A for the appropriate drop:

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

Bit	Function
1 ... 8	At least one error in words 173 ...175
9 ... 162	Counts No responses

The third word in each group of three is the drop cumulative error counter on Cable B for the appropriate drop:

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

Bit	Function
1 ... 8	At least one error in words 176 ...178
9 ... 162	Counts No responses

Note: For PLCs where drop 1 is reserved for local I/O, status words 182 ... 184 are used as follows:

Word 182 displays local drop status:

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

Bit	Function
1	1 = All modules healthy
2 ... 8	Always 0
9 ... 162	Number of times a module has been seen as unhealthy; counter rolls over at 255

Word 183 is used as a 16-bit I/O bus error counter.

Word 184 is used as a 16-bit I/O bus retry counter.

Controller Status Words 1 - 11 for TSX Compact and Atrium

CPU Status (Word 1)

Word 1 displays the following aspects of the CPU status:

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

Bit	Function
1 - 5	Not used
6	1 = enable constant sweep
7	1 = enable single sweep delay
8	1 = 16 bit user logic 0 = 24 bit user logic
9	1 = AC power on
10	1 = RUN light OFF
11	1 = memory protect OFF
12	1 = battery failed
13 - 16	Not used

Word 2

This word is not used.

Controller Status (Word 3)

Word 3 displays aspects of the controller status:

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

Bit	Function
1	1 = first scan
2	1 = start command pending
3	1 = scan time has exceed constant scan target
4	1 = existing DIM AWARENESS
5 - 12	Not used
13 - 16	Single sweeps

Word 4

This word is not used.

CPU Stop State (Word 5)

Word 5 displays the CPU's stop state conditions:

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

Bit	Function
1	1 = peripheral port stop
2	1 = XMEM parity error
3	1 = DIM AWARENESS
4	1 = illegal peripheral intervention
5	1 = invalid segment scheduler
6	1 = no start-of-network (SON) at the start of a segment
7	1 = state RAM test failed
8	1 = no end of logic (EOL), (bad Tcop)
9	1 = watch dog timer has expired
10	1 = real time clock error
11	1 = CPU failure
12	Not used
13	1 = invalid node in ladder logic
14	1 = logic checksum error
1	1 = coil disabled in RUN mode
16	1 = bad PLC setup

Number of Segments in program (Word 6)

Word 6 displays the number of segments in ladder logic; a binary number is shown. This word is confirmed during power up to be the number of EOS (DOIO) nodes plus 1 (for the end of logic nodes), if untrue, a stop code is set, causing the run light to be off:

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

Bit	Function
1 - 16	Number of segments in the current ladder logic program (expressed as a decimal number)

Address of the End of Logic Pointer (Word 7)

Word 7 displays the address of the end-of-logic (EOL) pointer:

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

Bit	Function
1 - 16	EOL pointer address

Word 8, Word 9

These words are not used.

RUN/LOAD/ DEBUG Status (Word 10)

Word 10 uses its two least significant bits to display RUN/LOAD/DEBUG status:

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

Bit	Function
1 ... 14	Not used
15, 16	0 0 = Debug (0 dec) 0 1 = Run (1 dec) 1 0 = Load (2 dec)

Word 11

This word is not used.

I/O Module Health Status Words 12 - 15 for TSX Compact

TSX Compact I/O Module Health

Words 12 ... 15 are used to display the health of the A120 I/O modules in the four racks:

Word	Rack No.
12	1
13	2
14	3
15	4

Each word contains the health status of up to five A120 I/O modules. The most significant (left-most) bit represents the health of the module in Slot 1 of the rack:

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

Bit	Function
1	1 = Slot 1
2	1 = Slot 2
3	1 = Slot 3
4	1 = Slot 4
5	1 = Slot 5
6 ... 16	Not used

If a module is I/O Mapped and ACTIVE, the bit will have a value of "1". If a module is inactive or not I/O Mapped, the bit will have a value of "0".

Note: Slots 1 and 2 in Rack 1 (Word 12) are not used because the controller itself uses those two slots.

Global Health and Communications Retry Status Words 182 ... 184 for TSX Compact

Overview

There are three words that contain health and communication information on the installed I/O modules. If monitored with the Stat block, they are found in Words 182 through 184. This requires that the length of the Stat block is a minimum of 184 (Words 16 through 181 are not used).

Words 16 ... 181

These words are not used.

Health Status (Word 182)

Word 182 increments each time a module becomes bad. After a module becomes bad, this counter does not increment again until that module becomes good and then bad again.

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

Bit	Function
1	1 = All modules healthy
2 ... 9	Not used
10 ... 16	"Module went unhealthy" counter

I/O Error Counter (Word 183)

This counter is similar to the above counter, except this word increments every scan that a module remains in the bad state.

PAB Bus Retry Counter (Word 184)

Diagnostics are performed on the communications through the bus. This word should normally be all zeroes. If after 5 retries, a bus error is still detected, the controller will stop and error code 10 will be displayed. An error could occur if there is a short in the backplane or from noise. The counter rolls over while running. If the retries are less than 5, no bus error is detected.

SU16: Subtract 16 Bit

134

At a Glance

Introduction

This chapter describes the instruction SU16.

What's in this chapter?

This chapter contains the following topics:

Short Description

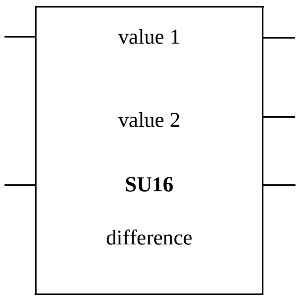
Function Description

The SU16 instruction performs a signed or unsigned 16-bit subtraction (value 1 - value 2) on the top and middle node values, then posts the signed or unsigned difference in a 4x holding register in the bottom node.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = enables value 1 - value 2

Bottom input	0x, 1x	None	ON = signed operation
			OFF = unsigned operation

value 1 (top node)	3x, 4x	INT, UINT	Minuend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Subtrahend, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
difference (bottom node)	4x	INT, UINT	Difference
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

SUB: Subtraction

135

At a Glance

Introduction

This chapter describes the instruction SUB.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

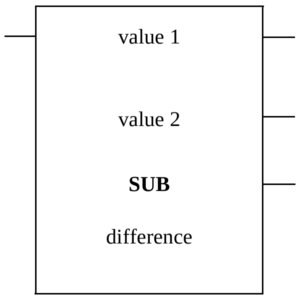
The SUB instruction performs a signed or unsigned 16-bit subtraction (value 1 - value 2) on the top and middle node values, then posts the signed or unsigned difference in a 4x holding register in the bottom node.

Note: SUB is often used as a comparator where the state of the outputs identifies whether value 1 is greater than, equal to, or less than value 2.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables value 1 - value 2
value 1 (top node)	3x, 4x	INT, UINT	Minuend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Subtrahend, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register
difference (bottom node)	4x	INT, UINT	Difference
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

T––>R: Table to Register

136

At a Glance

Introduction

This chapter describes the instruction T→R.

What's in this chapter?

This chapter contains the following topics:

Short Description

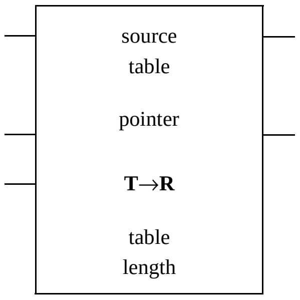
Function Description

The T→R instruction copies the bit pattern of a register or 16 contiguous discretes in a table to a specific holding register. It can accommodate the transfer of one register per scan. It has three control inputs and produces two possible outputs.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = copies source data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register or discrete reference in the source table. A register or string of contiguous discretes from this table will be copied in a scan.
pointer (middle node)	4x	INT, UINT	Pointer to the destination where the source data will be copied
table length (bottom node)		INT, UINT	Length of the source table: number of registers that may be copied; range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Middle Input

When the middle input goes ON, the current value stored in the pointer register is frozen while the DX operation continues. This causes the same table data to be written to the destination register on each scan.

Bottom Input

When the bottom input goes ON, the value in the pointer is reset to zero. This causes the next DX move operation to copy the first destination register in the table.

Pointer (Middle Node)

The 4x register entered in the middle node is a pointer to the destination where the source data will be copied. The destination register is the next contiguous 4x register after the pointer. For example, if the middle node displays a pointer of 400100, then the destination register for the T→R copy is 400101.

The value stored in the pointer register indicates which register in the source table will be copied to the destination register in the current scan. A value of 0 in the pointer indicates that the bit pattern in the first register of the source table will be copied to the destination; a value of 1 in the pointer register indicates that the bit pattern in the second register of the source table will be copied to the destination register; etc.

T––>T: Table to Table

137

At a Glance

Introduction

This chapter describes the instruction T→T.

What's in this chapter?

This chapter contains the following topics:

Short Description

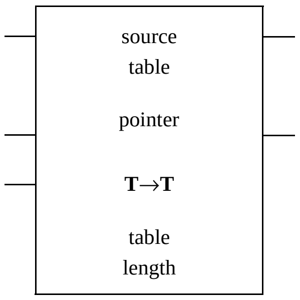
Function Description

The T→T instruction copies the bit pattern of a register or of 16 discretes from a position within one table to an equivalent position in another table of registers. It can accommodate the transfer of one register per scan. It has three control inputs and produces two possible outputs.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = copies source data and increments the pointer value
Middle input	0x, 1x	None	ON = freezes the pointer value
Bottom input	0x, 1x	None	ON = resets the pointer value to zero
source table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register or discrete reference in the source table. A register or string of contiguous discretes from this table will be copied in a scan.
pointer (middle node)	4x	INT, UINT	Pointer into both the source and destination table
table length (bottom node)		INT, UINT	Length of the source and the destination table (must be equal in length); range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = pointer value = table length (instruction cannot increment any further)

Parameter Description

Middle Input	When the input to the middle node goes ON, the current value stored in the pointer register is frozen while the DX operation continues. This causes new data being copied to the destination to overwrite the data copied on the previous scan.
Bottom Input	When the input to the bottom node goes ON, the value in the pointer register is reset to zero. This causes the next DX move operation to copy source data into the first register in the destination table.
Pointer (Middle Node)	The 4x register entered in the middle node is a pointer into both the source and destination tables, indicating where the data will be copied from and to in the current scan. The first register in the destination table is the next contiguous 4x register following the pointer. For example, if the middle node displays a a pointer reference of 400100, then the first register in the destination table is 400101. The value stored in the pointer register indicates which register in the source table will be copied to which register in the destination table. Since the length of the two tables is equal and T→T copy is to the equivalent register in the destination table, the current value in the pointer register also indicates which register in the destination table the source data will be copied to. A value of 0 in the pointer register indicates that the bit pattern in the first register of the source table will be copied to the first register of the destination table; a value of 1 in the pointer register indicates that the bit pattern in the second register of the source table will be copied to the second register of the destination register; etc.

T.01 Timer: One Hundredth Second Timer

138

At a Glance

Introduction

This chapter describes the instruction T.01 Timer.

What's in this chapter?

This chapter contains the following topics:

Short Description

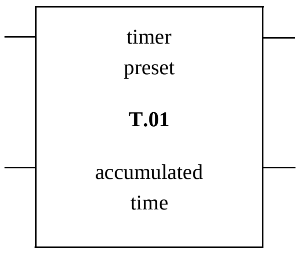
Function Description

The T.01 instruction measures time in hundredth of a second intervals. It can be used for timing an event or creating a delay. T.01 has two control inputs and can produce one of two possible outputs.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	OFF → ON = initiates the timer operation: time accumulates in hundredths-of-a-second when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of hundredth-of-a-second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in hundredth-of-a-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T0.1 Timer: One Tenth Second Timer

139

At a Glance

Introduction

This chapter describes the instruction T0.1 Timer.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

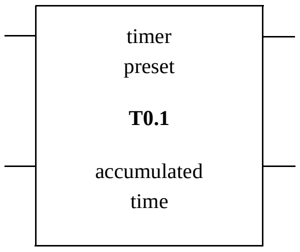
The T0.1 instruction measures time in tenth-of-a-second increments. It can be used for timing an event or creating a delay. T0.1 has two control inputs and can produce one of two possible outputs.

Note: If you cascade T0.1 timers with presets of 1, the timers will time-out together; to avoid this problem, change the presets to 10 and substitute a T.01 timer.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	OFF → ON = initiates the timer operation: time accumulates in tenth-of-a-second when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of tenth-of-a-second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in tenth-of-a-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T1.0 Timer: One Second Timer

140

At a Glance

Introduction

This chapter describes the instruction T1.0 Timer.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

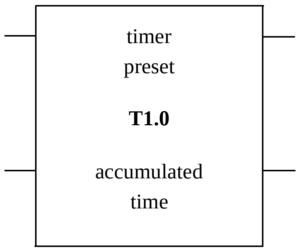
The T1.0 timer instruction measures time in one-second increments. It can be used for timing an event or creating a delay. T1.0 has two control inputs and can produce one of two possible outputs.

Note: If you cascade T1.0 timers with presets of 1, the timers will time-out together; to avoid this problem, change the presets to 10 and substitute a T0.1 timer.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	OFF → ON = initiates the timer operation: time accumulates in seconds when top and bottom input are ON
Bottom input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of one second increments), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (bottom node)	4x	INT, UINT	Accumulated time count in one-second increments.
Top output	0x	None	ON = accumulated time = timer preset
Bottom output	0x	None	ON = accumulated time < timer preset

T1MS Timer: One Millisecond Timer

141

At a Glance

Introduction

This chapter describes the instruction T1MS Timer.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

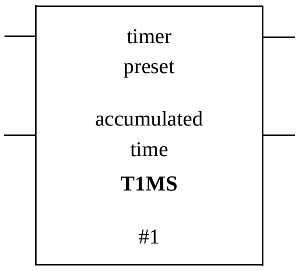
Note: This instruction is available in Micro PLC models and the Quantum CPU 424 02 PLC.

The T1MS timer instruction measures time in one-millisecond increments. It can be used for timing an event or creating a delay.

Representation

Symbol

Representation of the instruction



Parameter Description

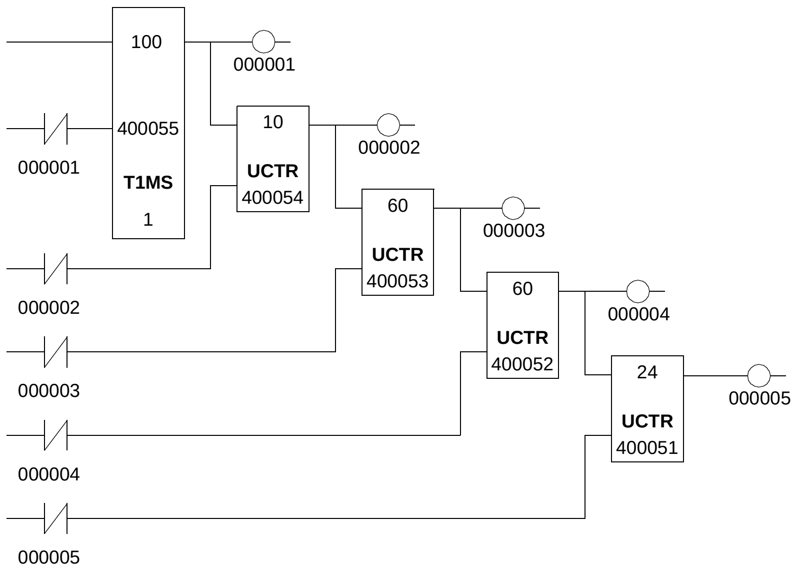
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates the timer operation: time accumulates in milliseconds when top and middle input are ON
Middle input	0x, 1x	None	OFF = accumulated time reset to 0 ON = timer accumulating
timer preset (top node)	3x, 4x	INT, UINT	Preset value (number of millisecond increments the timer can accumulate), can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
accumulated time (middle node)	4x	INT, UINT	Accumulated time count in millisecond increments.
#1 (bottom node)		INT, UINT	Constant value of #1
Top output	0x	None	ON = accumulated time = timer preset
Middle output	0x	None	ON = accumulated time < timer preset

Example

A Millisecond Timer Example

Here is the ladder logic for a real-time clock with millisecond accuracy:



The T1MS instruction is programmed to pass power at 100 ms intervals; it is followed by a cascade of four up-counters (See *UCTR: Up Counter, p. 705*) that store the time respectively in hundredth-of-a-second units, tenth-of-a-second units, one- second units, one-minute units, and one-hour units.

When logic solving begins, the accumulated time value begins incrementing in register 40055 of the T1MS block. After 100 one-ms increments, the top output passes power and energizes coil 00001. At this point, the value in register 40055 in the timer is reset to 0. The accumulated count value in register 40054 in the first UCTR block increments by 1, indicating that 100 ms have passed. Because the accumulated time count in T1MS no longer equals the timer preset, the timer begins to re-accumulate time in ms.

When the accumulated count in register 40054 of the first UCTR instruction increments to 10, the top output from that instruction block passes power and energizes coil 00002. The value in register 40054 then resets to 0, and the accumulated count in register 40053 of the second UCTR block increments by 1.

As the times accumulate in each counter, the time of day can be read in five holding registers as follows:

Register	Unit of Time	Valid Range
40055	Thousandths-of-a-second	0 ... 100
40054	Tenths-of-a-second	0 ... 10
40053	Seconds	0 ... 60
40052	Minutes	0 ... 60
40051	Hours	0 ... 24

TBLK: Table to Block

142

At a Glance

Introduction

This chapter describes the instruction TBLK.

What's in this chapter?

This chapter contains the following topics:

Short Description

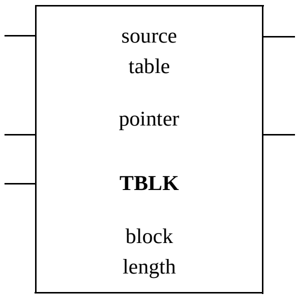
Function Description

The TBLK (table-to-block) instruction combines the functions of T→R and the BLKM in a single instruction. In one scan, it can copy up to 100 contiguous 4x registers from a table to a destination block. The destination block is of a fixed length. The block of registers being copied from the source table is of the same length, but the overall length of the source table is limited only by the number of registers in your system configuration.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates move operation
Middle input	0x, 1x	None	ON = hold pointer
Bottom input	0x, 1x	None	ON = reset pointer to zero
source table (top node)	4x	INT, UINT, WORD	First holding register in the source table
pointer (middle node)	4x	INT, UINT	Pointer to the source block, destination block
block length (bottom node)		INT, UINT	Number of registers of the destination block and of the blocks within the source table; range: 1 ... 100
Top output	0x	None	ON = move successful
Middle output	0x	None	ON = error / move not possible

Parameter Description

Middle Input

When the middle input is ON, the value in the pointer register is frozen while the TBLK operation continues. This causes the same source data block to be copied to the destination table on each scan.

Bottom Input

When the bottom input is ON, the pointer value is reset to zero. This causes the TBLK operation to copy data from the first block of registers in the source table.

CAUTION

Confine the value in the destination pointer to a safe range.

You should use external logic in conjunction with the middle and the bottom inputs to confine the value in the destination pointer to a safe range.

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

Source Table (Top Node)

The 4x register entered in the top node is the first holding register in the source table.

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

Pointer (Middle Node)

The 4x register entered in the middle node is the pointer to the source block. The first register in the destination block is the next contiguous register after the pointer. For example, if the pointer is register 400107, then the first register in the destination block is 400108.

The value stored in the pointer indicates which block of data from the source table will be copied to the destination block. This value specifies a block number within the source table.

TEST: Test of 2 Values

143

At a Glance

Introduction

This chapter describes the instruction TEST.

What's in this chapter?

This chapter contains the following topics:

Short Description

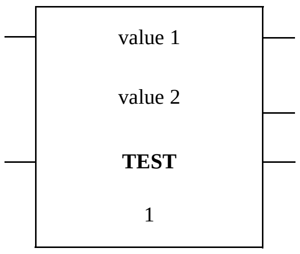
Function Description

The TEST instruction compares the signed or unsigned size of the 16-bit values in the top and middle nodes and describes the relationship via the block outputs.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = compares value 1 and value 2

Bottom input	0x, 1x	None	ON = signed operation
			OFF = unsigned operation

value 1 (top node)	3x, 4x	INT, UINT	Value 1, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Value 2, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
1 (bottom node)		INT, UINT	Constant value, cannot be changed
Top output	0x	None	ON = value 1 > value 2
Middle output	0x	None	ON = value 1 = value 2
Bottom output	0x	None	ON = value 1 < value 2

UCTR: Up Counter

144

At a Glance

Introduction

This chapter describes the instruction UCTR.

What's in this chapter?

This chapter contains the following topics:

Short Description

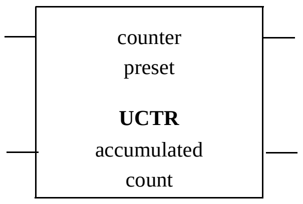
Function Description

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

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	OFF → ON = initiates the counter operation

Bottom input	0x, 1x	None	OFF = reset accumulator to 0
			ON = counter accumulating

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

WRIT: Write

145

At a Glance

Introduction

This chapter describes the instruction WRIT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

The WRIT instruction sends a message from the PLC over the RIO communications link to an ASCII display (screen, printer, etc.).

In the process of sending the messaging operation, WRIT performs the following functions:
- Verifies the correctness of the ASCII communication parameters, e.g. the port number, the message number
- Verifies the lengths of variable data fields
- Performs error detection and recording
- Reports RIO interface status

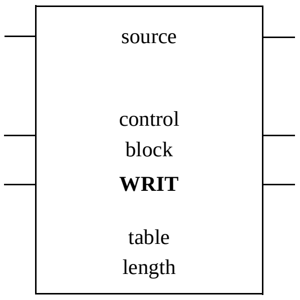
WRIT requires two tables of registers: a source table where variable data (the message) is copied, and a control block where comm port and message parameters are identified.

Further information about formatting messages you will find in *Formatting Messages for ASCII READ/WRIT Operations, p. 29*.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates a WRIT
Middle input	0x, 1x	None	ON = pauses WRIT operation
Bottom input	0x, 1x	None	ON = abort WRIT operation
source (top node)	3x, 4x	INT, UINT, WORD	Source table
control block (middle node)	4x	INT, UINT, WORD	ASCII Control block (first of seven contiguous holding registers)
table length (bottom node)		INT, UINT	Length of source table (number of registers where the message data will be stored), range: 1 ... 999
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON = error in communication or operation has timed out (for one scan)
Bottom output	0x	None	ON = WRIT complete (for one scan)

Parameter Description

Source Table (Top Node)

The top node contains the first 3x or 4x register in a source table whose length is specified in the bottom node. This table contains the data required to fill the variable field in a message.

Consider the following WRIT message

vessel #1 temperature is**:** **III**

The 3-character ASCII field `III` is the variable data field; variable data are loaded, typically via DX moves, into a table of variable field data.

Control Block (Middle Node)

The 4x register entered in the middle node is the first of seven contiguous holding register in the control block.

Register	**Definition**
Displayed	*Port Number and Error Code, p. 711*
First implied	Message number
Second implied	Number of registers required to satisfy format
Third implied	Count of the number of registers transmitted thus far
Fourth implied	Status of the solve
Fifth implied	Reserved
Sixth implied	Checksum of registers 0 ... 5

Port Number and Error Code

Port Number and Error Code

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

Bit	Function
1 ... 4	PLC error code (see table below)
5	Not used
6	Input from the ASCII device not compatible with format
7	Input buffer overrun, data received too quickly at RIOP
8	USART error, bad byte received at RIOP
9	Illegal format, not received properly by RIOP
10	ASCII device off-line, check cabling
11	ASCII message terminated early (in keyboard mode
12 ... 16	Comm port # (1 ... 32)

PLC Error Code

Bit				**Meaning**
1	**2**	**3**	**4**	
0	0	0	1	Error in the input to RIOP from ASCII device
0	0	1	0	Exception response from RIOP, bad data
0	0	1	1	Sequenced number from RIOP differs from expected value
0	1	0	0	User register checksum error, often caused by altering READ registers while the block is active
0	1	0	1	Invalid port or message number detected
0	1	1	0	User-initiated abort, bottom input energized
0	1	1	1	No response from drop, communication error
1	0	0	0	Node aborted because of SKP instruction
1	0	0	1	Message area scrambled, reload memory
1	0	1	0	Port not configured in the I/O map
1	0	1	1	Illegal ASCII request
1	1	0	0	Unknown response from ASCII port
1	1	0	1	Illegal ASCII element detected in user logic
1	1	1	1	RIOP in the PLC is down

XMIT: XMIT Communication Block

146

At a Glance

Introduction

This chapter describes the instruction XMIT.

What's in this chapter?

This chapter contains the following topics:

Short Description

Function Requirements

The following steps are necessary before using this instruction:

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

	CAUTION
⚠	**The outputs of the instruction turn on, regardless of the input states** When the NSUP loadable is not installed or is installed after the XMIT loadable or is installed in a Quantum PLC with an executive < V2.0, all three outputs turn on, regardless of the input states. **Failure to observe this precaution can result in injury or equipment damage.**

Step	Action
2	Unpack and install the DX Loadable XMIT; further information you will find in the chapter *Installation of DX Loadables, p. 41*.

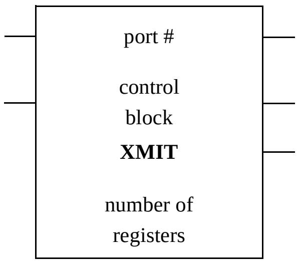
Function Description

The XMIT instruction is provided to receive and transmit ASCII messages and Modbus master messages using the PLC ports.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	**State RAM Reference**	**Data Type**	**Meaning**
Top input	0x, 1x	None	ON = Start XMIT operation
Middle input	0x, 1x	None	Abort XMIT operation
port # (top node)	4x	INT	PLC port selection
control block (middle node)	4x	INT, UINT	Control block (first of sixteen contiguous registers)
length (bottom node)		INT	Number of registers used by the XMIT instruction (must be a constant equal to 16)
Top output	0x, 1x	None	Operation is active
Middle output	0x, 1x	None	Operation terminated unsuccessfully (error detected or aborted)
Bottom output	0x, 1x	None	Operation has been successfully completed

Detailed Description

Mode of Functioning

The XMIT (Transmit) instruction sends Modbus messages from a master PLC to multiple slave PLCs or sends ASCII character strings from the PLC's Modbus slave port#1 or port#2 to ASCII printers and terminals. XMIT sends these messages over telephone dialup modems, radio modems, or simply direct connection.

XMIT comes with three modes:
- communication mode
- port status mode
- conversion mode.

XMIT performs general ASCII input functions in the communication mode including simple ASCII and terminated ASCII. You may use an additional XMIT block for reporting port status information into registers while another XMIT block performs the ASCII communication function. You may import and export ASCII or binary data into your PLC and convert it into various binary data or ASCII to send to DCE (Data Communication Equipment) devices based upon the needs of your application. The block has built-in diagnostics that checks to make sure no other XMIT blocks are active in the PLC. Within the XMIT block a control table allows you to control the communications link between the PLC and DCE (Data Communication Equipment) devices attached to Modbus port #1 or port#2 of the PLC. The XMIT block does NOT activate the port LED when it is transmitting data.
Further information you will find in the *Modicon XMIT Function Block User Guide*.

	CAUTION
⚠	**Contention and Collision when using the XMIT instruction in a network with multiple masters** Remember, the Modbus protocol is a master/ slave protocol. Modbus is designed to have only one master polling multiple slaves. Therefore, when using the XMIT instruction in a network with multiple masters, contention resolution and collision avoidance is your responsibility and may easily be addressed through ladder logic programming. **Failure to observe this precaution can result in injury or equipment damage.**

Top Input

The top input begins an XMIT operation and it should remain ON until the operation has completed successfully or an error has occurred.

Middle Input

The middle input aborts any active XMIT operation and forces the port to slave mode. An abort code (121) is placed into the fault status register. The port remains closed as long as this input is ON.

Note: To reset an XMIT fault and clear the fault register, the top input must go OFF for at least one PLC scan.

Port # (Top Node)

In the top node you select the PLC port number, from where the messages are sent or received.
The top node must contain one of the following constants:
- #0001 = PLC port #1
- #0002 = PLC port #2

Control Block (Middle Node)

The 4x register entered in the middle node is the first of sixteen contiguous 4x registers that comprise the control block:

Register	Content
Displayed	Current revision number of XMIT block.
First implied	*Fault Status, p. 718*
Second implied	Available to user May be used as pointers for instructions like TBLK.
Third implied	Data Rate: 50, 75, 110, 134, 150, 300, 600, 1200, 2400, 9600 and 19200
Fourth implied	Data Bits: 7 for ASCII mode 8 for RTU mode
Fifth implied	Parity: 0 = no parity 1 = odd parity 2 = even parity
Sixth implied	Stop Bits 1 = one stop bit 2 = two stop bits
Seventh implied	Available to user May be used as pointers for instructions like TBLK.
Eighth implied	*Command Word, p. 720* 0000-0001-0000-0000 (256Dec)
Ninth implied	Pointer to message table (See *Message Pointer, p. 722*) Limited by the range of 4x registers configured
10th implied	Length of message Range: 0...512

Register	Content
11th implied	Response Time Out (ms) Range: 0 ... 65535
12th implied	Retry limit Range: 0 ... 65535
13th implied	Start of transmission delay (ms) Range: 0 ... 65535
14th implied	End of transmission delay (ms) Range: 0 ... 65535
15th implied	Current number of retry attempts made by the instruction

WARNING

No modification of the control block address

Do **not** modify the address in the middle node of the XMIT block or delete it from the program while it is active. This locks up the port preventing communications.

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

Fault Status

The following fault code is generated by the XMIT instruction:

Fault Code	Fault Description
1	Modbus exception Illegal function
2	Modbus exception Illegal data address
3	Modbus exception Illegal data value
4	Modbus exception Slave device failure
5	Modbus exception Acknowledge
6	Modbus exception Slave device busy
7	Modbus exception Negative acknowledge
8	Modbus exception Memory parity error
9 ... 99	Reserved
100	Slave PLC data area cannot equal zero
101	Master PLC data area cannot equal zero
102	Coil (0x) not configured
103	Holding register (4x) not configured
104	Data length cannot equal zero

Fault Code	Fault Description
105	Pointer to message table cannot equal zero
106	Pointer to message table is outside the range of configured holding registers (4x)
107	Transmit message timeout This error is generated when the UART cannot complete a transmission in 10 seconds or less. This error bypasses the retry counter and will activate the error output on the first error.
108	Undefined error
109	Modem returned ERROR
110	Modem returned NO CARRIER
111	Modem returned NO DIALTONE
112	Modem returned BUSY
113	Invalid LRC checksum from the slave PLC
114	Invalid CRC checksum from the slave PLC
115	Invalid Modbus function code
116	Modbus response message timeout
117	Modem reply timeout
118	XMIT could not gain access to PLC communications port #1 or port #2
119	XMIT could not enable PLC port receiver
120	XMIT could not set PLC UART
121	User issued an abort command
122	Top node of XMIT not equal to zero, one or two
123	Bottom node of XMIT is not equal to seven, eight or sixteen
124	Undefined internal state
125	Broadcast mode not allowed with this Modbus function code
126	DCE did not assert CTS
127	Illegal configuration (data rate, data bits, parity, or stop bits)
128	Unexpected response received from Modbus slave
129	Illegal command word setting
130	Command word changed while active
131	Invalid character count
132	Invalid register block
133	ASCII input FIFO overflow error
134	Invalid number of start characters or termination characters

Command Word

Command Word

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

BIT	Function	Definition
Bit 1 (msb)		Reserved
Bit 2	Enable RTS/ CTS modem control	Set to 1 when a DCE connected to the PLC requires hardware handshaking using RTS/CTS control. This bit may be used in conjunction with values contained in (4x + 13) and (4x + 14). Start of transmission delay (4x + 13) keeps RTS asserted for (X ms) before XMIT sends message out of PLC port #1. Likewise, end of transmission delay (4x + 14) keeps RTS asserted for (X ms) after XMIT has finished sending a message out of the PLC port #1. Once the end of transmission delay expires XMIT de-assert RTS.
Bit 3	Enable RS485 mode	Set to 1 when the selected port should operate in RS485 mode. Otherwise it defaults to 0, which is RS232 mode.
Bit 4		Reserved
Bit 5	Terminated ASCII input	Set to 1 to remove and discard all characters from FIFO until the starting string is matched, then these starting characters and subsequent characters are written into a contiguous 4x register destination block until the terminator sequence is matched. The terminator string is also written into the 4x register destination block.
Bit 6	Simple ASCII input	Set to 1 to remove the ASCII characters from FIFO for writing into a contiguous 4x register block. The Message pointer (4x + 9) specifics the 4x register block.
Bit 7	Enable ASCII string messaging	Set to 1 when you want to send ASCII messages out of the PLC. XMIT sends ASCII strings up to 1024 characters in length. You program each ASCII message into contiguous 4x registers of the PLC. Two characters allowed per register. Only use Bit 7 OR Bit 8, do not try to use both.
Bit 8	Enable Modbus messaging	Set to 1 when you want to send Modbus messages out of the PLC. Modbus messages may be in either RTU or ASCII formats. When data bits=8, XMIT uses Modbus RTU format. When data bits=7, XMIT uses Modbus ASCII format. Only use Bit 7 OR Bit 8, do not try to use both.
Bit 9	Enable ASCII receive FIFO	Set to 1 to allow the XMIT block to take control over the selected port (1 or 2) from the PLC. The block begins to receive ASCII characters into an empty 512 byte circular FIFO.
Bit 10	Enable back space	Set to 1 to allow special handling of ASCII back space character (BS, 8Hex). When using either Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 5) each back space character is removed from FIFO and may or may NOT be stored into a 4x register destination block.

BIT	Function	Definition
Bit 11	Enable RTS/ CTS flow control	Set to 1 to allow full duplex hardware flow control using the RTS and CTS handshaking signals for ASCII massaging. The RTS/CTS operates in both the input and output modes.
Bit 12	Enable Xon/ Xoff flow control	Set to 1 to allow full duplex software flow control using the ASCII Xon character (DC1, 11 Hex) and the ASCII Xoff character (DC3, 13 Hex). The Xon/Xoff operates in both the input and output modes.
Bit 13	Pulse dial modem	Set to 1 when using a Hayes compatible dial-up modem and you wish to pulse dial a telephone number. You program the phone number into contiguous 4x registers of the PLC. A pointer to these registers must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Pulse dialed numbers are sent to the modem automatically preceded by ATDP and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 14	Disconnect modem	Set to 1 when using a Hayes compatible dial-up modem and you want to disconnect the modem. You must use ladder logic to turn this bit ON. Since the disconnect message is an ASCII string, bit 7 must be ON prior to sending the message. disconnect messages are sent to the modem automatically preceded by +++AT and with carriage return <CR> and line feed <LF> appended. XMIT looks for a correct disconnect response from the modem before it turns ON the bottom output, noting a successful completion.
Bit 15	Tone dial modem	Set to 1 when using a Hayes compatible dial-up modem and you wish to tone dial a telephone number. You program the dial message into contiguous 4x registers of the PLC. A pointer to the dial message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Tone dial numbers are sent to the modem automatically preceded by ATDT and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 16	Initialize modem	Set to 1 when using a Hayes compatible dial-up modem and you want to initialize the modem. You program the initialization message into contiguous 4x registers of the PLC. A pointer to the initialization message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). All messages are sent to the modem automatically preceded by AT and with a carriage return <CR> and line feed <LF> appended. Since the initialization message is an ASCII string, bit 7 must be ON prior to sending the message.

Detailed information about the bits of the command word you will find in the *Modicon XMIT Function Block User Guide*.

Message Pointer

You enter a pointer that points to the beginning of the message table. There are two different handlings of the pointer depending on using ASCII character strings or Modbus messages.

For **ASCII character strings**, the pointer is the register offset to the first register of the ASCII character string. Each register holds up to two ASCII characters. Each ASCII string may be up to 1024 characters in length. For example, when you want to send 10 ASCII messages out of the PLC, you must program 10 ASCII character strings into 4x registers of the PLC and then through ladder logic set the pointer to the start of each message after each successful operation of XMIT.
For **Modbus messages**, the pointer is the register offset to the first register of the Modbus definition table. The Modbus definition table has different length depending on the used Modbus function code and you must program it for successful XMIT operation.
Detailed information about the bits of the command word you will find in the *Modicon XMIT Function Block User Guide*.

Outputs

CAUTION

All three outputs of the instruction turn on, regardless of the input states

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

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

XMRD: Extended Memory Read

147

At a Glance

Introduction

This chapter describes the instruction XMRD.

What's in this chapter?

This chapter contains the following topics:

Short Description

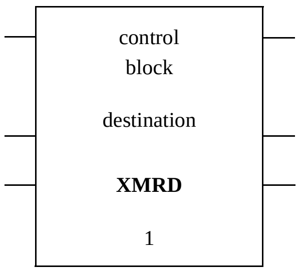
Function Description

The XMRD instruction is used to copy a table of 6x extended memory registers to a table of 4x holding registers in state RAM.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction’s parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = activates read operation
Middle input	0x, 1x	None	OFF = clears offset to 0 ON = does not clear offset
Bottom input	0x, 1x	None	OFF = abort on error ON = do not abort on error
control block (top node)	4x	INT, UINT, WORD	First of six contiguous holding registers in the extended memory
destination (middle node)	4x	INT, UINT, WORD	The first 4x holding register in a table of registers that receive the transferred data from the 6x extended memory storage registers
1 (bottom node)		INT, UINT	Contains the constant value 1, which cannot be changed
Top output	0x	None	Read transfer active
Middle output	0x	None	Error condition detected
Bottom output	0x	None	ON = operation complete

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of six contiguous holding registers in the extended memory control block.

Reference	Register Name	Description
Displayed	status word	Contains the diagnostic information about extended memory (see *Status Word of the Control Block, p. 726*)
First implied	file number	Specifies which of the extended memory files is currently in use (range: 1 ... 10)
Second implied	start address	Specifies which 6x storage register in the current file is the starting address; 0 = 60000, 9999 = 69999
Third implied	count	Specifies the number of registers to be read or written in a scan when the appropriate function block is powered; range: 0 ... 9999, not to exceed number specified in max registers (fifth implied)
Fourth implied	offset	Keeps a running total of the number of registers transferred thus far
Fifth implied	max registers	Specifies the maximum number of registers that may be transferred when the function block is powered (range: 0 ... 9999)

If you are in multi-scan mode, these six registers should be unique to this function block.

Status Word of the Control Block

Status Word of the Control Block

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

Bit	Function
1	1 = power-up diagnostic error
2	1 = parity error in extended memory
3	1 = extended memory does not exist
4	0 = transfer not running 1 = busy
5	0 = transfer in progress 1 = transfer complete
6	1 = file boundary crossed
7	1 = offset parameter too large
8 - 9	Not used
10	1 = nonexistent state RAM
11	Not used
12	1 = maximum registers parameter error
13	1 = offset parameter error
14	1 = count parameter error
15	1 = starting address parameter error
16	1 = file number parameter error

XMWT: Extended Memory Write

148

At a Glance

Introduction

This chapter describes the instruction XMWT.

What's in this chapter?

This chapter contains the following topics:

Short Description

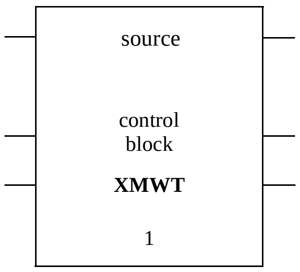
Function Description

The XMWT instruction is used to write data from a block of input registers or holding registers in state RAM to a block of 6x registers in an extended memory file.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = activates write operation
Middle input	0x, 1x	None	OFF = clears offset to 0 ON = does not clear offset
Bottom input	0x, 1x	None	OFF = abort on error ON = do not abort on error
source (top node)	3x, 4x	INT, UINT, WORD	The first 3x or 4x register in a block of contiguous source registers, i.e. input or holding registers, whose contents will be written to 6x extended memory registers
control block (middle node)	4x	INT, UINT, WORD	First of six contiguous holding registers in the extended memory
1 (bottom node)		INT, UINT	Contains the constant value 1, which cannot be changed
Top output	0x	None	Write transfer active
Middle output	0x	None	Error condition detected
Bottom output	0x	None	ON = operation complete

Parameter Description

Control Block (Top Node)

The 4x register entered in the middle node is the first of six contiguous holding registers in the extended memory control block.

Reference	Register Name	Description
Displayed	status word	Contains the diagnostic information about extended memory (see *Status Word of the Control Block, p. 730*)
First implied	file number	Specifies which of the extended memory files is currently in use (range: 1 ... 10)
Second implied	start address	Specifies which 6x storage register in the current file is the starting address; 0 = 60000, 9999 = 69999
Third implied	count	Specifies the number of registers to be read or written in a scan when the appropriate function block is powered; range: 0 ... 9999, not to exceed number specified in max registers (fifth implied)
Fourth implied	offset	Keeps a running total of the number of registers transferred thus far
Fifth implied	max registers	Specifies the maximum number of registers that may be transferred when the function block is powered (range: 0 ... 9999)

If you are in multi-scan mode, these six registers should be unique to this function block.

Status Word of the Control Block

Status Word of the Control Block

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

Bit	Function
1	1 = power-up diagnostic error
2	1 = parity error in extended memory
3	1 = extended memory does not exist
4	0 = transfer not running 1 = busy
5	0 = transfer in progress 1 = transfer complete
6	1 = file boundary crossed
7	1 = offset parameter too large
8 - 9	Not used
10	1 = nonexistent state RAM
11	Not used
12	1 = maximum registers parameter error
13	1 = offset parameter error
14	1 = count parameter error
15	1 = starting address parameter error
16	1 = file number parameter error

XOR: Exclusive OR

149

At a Glance

Introduction

This chapter describes the instruction XOR.

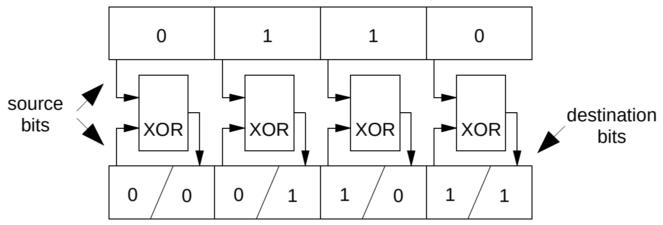
What's in this chapter?

This chapter contains the following topics:

Short Description

Function Description

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

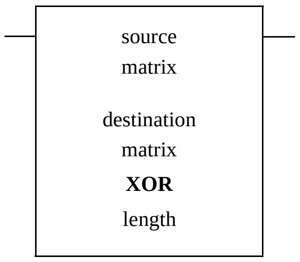


	WARNING
STOP	**XOR will override any disabled coils within the destination matrix without enabling them.** This can cause personal injury if a coil has disabled an operation for maintenance or repair because the coil's state can be changed by the XOR operation. **Failure to observe this precaution can result in severe injury or equipment damage.**

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

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

Parameter Description

Matrix Length (Bottom Node)

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

Glossary

A

active window	The window, which is currently selected. Only one window can be active at any one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.
Actual parameter	Currently connected Input/Output parameters.
Addresses	(Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules. The display/input of direct addresses is possible in the following formats: ● Standard format (400001) ● Separator format (4:00001) ● Compact format (4:1) ● IEC format (QW1)
ANL_IN	ANL_IN stands for the data type "Analog Input" and is used for processing analog values. The 3x References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.
ANL_OUT	ANL_OUT stands for the data type "Analog Output" and is used for processing analog values. The 4x-References of the configured analog output module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.
ANY	In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

ANY_BIT	In the existing version, "ANY_BIT" covers the data types BOOL, BYTE and WORD.
ANY_ELEM	In the existing version "ANY_ELEM" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD.
ANY_INT	In the existing version, "ANY_INT" covers the data types DINT, INT, UDINT and UINT.
ANY_NUM	In the existing version, "ANY_NUM" covers the data types DINT, INT, REAL, UDINT and UINT.
ANY_REAL	In the existing version "ANY_REAL" covers the data type REAL.
Application window	The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.
Argument	Synonymous with Actual parameters.
ASCII mode	American Standard Code for Information Interchange. The ASCII mode is used for communication with various host devices. ASCII works with 7 data bits.
Atrium	The PC based controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB)	The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (**Objects** → **Source**). If a back up file can be assigned, the name of the source file can be given.
Base 16 literals	Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 8 literal

Base 8 literals function as the input of whole number values in the octal system. The base must be denoted by the prefix 3.63kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
8#3_1111 or 8#377 (decimal 255)
8#34_1111 or 8#340 (decimal 224)

Basis 2 literals

Base 2 literals function as the input of whole number values in the dual system. The base must be denoted by the prefix 0.91kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_1111 or 2#11100000 (decimal 224)

Binary connections

Connections between outputs and inputs of FFBs of data type BOOL.

Bit sequence

A data element, which is made up from one or more bits.

BOOL

BOOL stands for the data type "Boolean". The length of the data elements is 1 bit (in the memory contained in 1 byte). The range of values for variables of this type is 0 (FALSE) and 1 (TRUE).

Bridge

A bridge serves to connect networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not deployed via bridges.

BYTE

BYTE stands for the data type "Bit sequence 8". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 8 bit. A numerical range of values cannot be assigned to this data type.

C

Cache	The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.
Call up	The operation, by which the execution of an operation is initiated.
Coil	A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.
Compact format (4:1)	The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.
Connection	A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.
Constants	Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).
Contact	A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

Data transfer settings

Settings, which determine how information from the programming device is transferred to the PLC.

Data types

The overview shows the hierarchy of data types, as they are used with inputs and outputs of Functions and Function blocks. Generic data types are denoted by the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System data types (IEC extensions)
- Derived (from "ANY" data types)

DCP I/O station

With a Distributed Control Processor (D908) a remote network can be set up with a parent PLC. When using a D908 with remote PLC, the parent PLC views the remote PLC as a remote I/O station. The D908 and the remote PLC communicate via the system bus, which results in high performance, with minimum effect on the cycle time. The data exchange between the D908 and the parent PLC takes place at 1.5 Megabits per second via the remote I/O bus. A parent PLC can support up to 31 (Address 2-32) D908 processors.

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs under Windows. The DDE interface can be used in the extended monitor to call up its own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data onto the PLC via the server. Data can therefore be altered directly in the PLC, while it monitors and analyzes the results. When using this interface, the user is able to make their own "Graphic-Tool", "Face Plate" or "Tuning Tool", and integrate this into the system. The tools can be written in any DDE supporting language, e.g. Visual Basic and Visual-C++. The tools are called up, when the one of the buttons in the dialog box extended monitor uses Concept Graphic Tool: Signals of a projection can be displayed as timing diagrams via the DDE connection between Concept and Concept Graphic Tool.

Decentral Network (DIO)	A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.
Declaration	Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.
Definition data file (Concept EFB)	The definition file contains general descriptive information about the selected FFB and its formal parameters.
Derived data type	Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept. Distinctions are made between global data types and local data types.
Derived Function Block (DFB)	A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol. The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version. Distinctions are made between local and global DFBs.
DINT	DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from –2 exp (31) to 2 exp (31) –1.
Direct display	A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.
Document window	A window within an Application window. Several document windows can be opened at the same time in an application window. However, only one document window can be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.
Dummy	An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.

DX Zoom	This property enables connection to a programming object to observe and, if necessary, change its data value.

E

Elementary functions/ function blocks (EFB)	Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.
EN / ENO (Enable / Error display)	If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands **Objects** → **Properties...** or via a double click on the FFB.
Error	When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output is set to "0".
Evaluation	The process, by which a value for a Function or for the outputs of a Function block during the Program execution is transmitted.
Expression	Expressions consist of operators and operands.

F

FFB (functions/ function blocks)	Collective term for EFB (elementary functions/function blocks) and DFB (derived function blocks)
Field variables	Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.
FIR filter	Finite Impulse Response Filter
Formal parameters	Input/Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.
Function (FUNC)	A Program organization unit, which exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values. Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.
Function block (item) (FB)	A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s). Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places. The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described in the corresponding definitions.

Function block dialog (FBD)	One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.
Function block type	A language element, consisting of: 1. the definition of a data structure, subdivided into input, output and internal variables, 2. A set of operations, which is used with the elements of the data structure, when a function block type instance is called up. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (called up) several times.
Function counter	The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m n = Section number (number running) m = Number of the FFB object in the section (number running)

G

Generic data type	A Data type, which stands in for several other data types.
Generic literal	If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.
Global derived data types	Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Global DFBs	Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Global macros	Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Groups (EFBs)	Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBs, especially in extensive libraries.

I

I/O component list	The I/O and expert assemblies of the various CPUs are configured in the I/O component list.
IEC 61131-3	International norm: Programmable controllers – part 3: Programming languages.
IEC format (QW1)	In the place of the address stands an IEC identifier, followed by a five figure address: ● %0x12345 = %Q12345 ● %1x12345 = %I12345 ● %3x12345 = %IW12345 ● %4x12345 = %QW12345
IEC name conventions (identifier)	An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö,ü, é, õ) can be used, taken from project and DFB names. Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively. Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier. Identifiers are not permitted to be Key words.
IIR filter	Infinite Impulse Response Filter
Initial step (starting step)	The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.
Initial value	The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.
Input bits (1x references)	The 1/0 status of input bits is controlled via the process data, which reaches the CPU from an entry device. **Note:** The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.
Input parameters (Input)	When calling up a FFB the associated Argument is transferred.

Input words (3x references)	An input word contains information, which come from an external source and are represented by a 16 bit figure. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.
Instantiation	The generation of an Item.
Instruction (IL)	Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.
Instruction (LL984)	When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.
Instruction list (IL)	IL is a text language according to IEC 1131, in which operations, e.g. conditional/unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.
INT	INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from –2 exp (15) to 2 exp (15) –1.
Integer literals	Integer literals function as the input of whole number values in the decimal system. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant. Example -12, 0, 123_456, +986
INTERBUS (PCP)	To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01. The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.

Item name	An Identifier, which belongs to a certain Function block item. The item name serves as a unique identifier for the function block in a program organization unit. The item name is automatically generated, but can be edited. The item name must be unique throughout the Program organization unit, and no distinction is made between upper/lower case. If the given name already exists, a warning is given and another name must be selected. The item name must conform to the IEC name conventions, otherwise an error message appears. The automatically generated instance name always has the structure: FBI_n_m
	
	FBI = Function block item
	n = Section number (number running)
	m = Number of the FFB object in the section (number running)

J

Jump	Element of the SFC language. Jumps are used to jump over areas of the chain.

K

Key words	Key words are unique combinations of figures, which are used as special syntactic elements, as is defined in appendix B of the IEC 1131-3. All key words, which are used in the IEC 1131-3 and in Concept, are listed in appendix C of the IEC 1131-3. These listed keywords cannot be used for any other purpose, i.e. not as variable names, section names, item names etc.

L

Ladder Diagram (LD)	Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.
Ladder Logic 984 (LL)	In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols),which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant. The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance. In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.
Landscape format	Landscape format means that the page is wider than it is long when looking at the printed text.
Language element	Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.
Library	Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries. EFB libraries can be subdivided into Groups.
Literals	Literals serve to directly supply values to inputs of FFBs, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated. Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears as Base 2 literal, Base 8 literal, Base 16 literal, Integer literal, Real literal or Real literal with exponent.
Local derived data types	Local derived data types are only available in a single Concept project and its local DFBs and are contained in the DFB directory under the project directory.

Local DFBs	Local DFBs are only available in a single Concept project and are contained in the DFB directory under the project directory.
Local link	The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.
Local macros	Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.
Local network nodes	The local node is the one, which is projected evenly.
Located variable	Located variables are assigned a state RAM address (reference addresses 0x,1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses. Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro	Macros are created with help from the software Concept DFB. Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration). Distinctions are made between local and global macros. Macros have the following properties: ● Macros can only be created in the programming languages FBD and LD. ● Macros only contain one single section. ● Macros can contain any complex section. ● From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro. ● Calling up DFBs in a macro ● Variable declaration ● Use of macro-own data structures ● Automatic acceptance of the variables declared in the macro ● Initial value for variables ● Multiple instancing of a macro in the whole program with different variables ● The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).
MMI	Man Machine Interface
Multi element variables	Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY. Distinctions are made between Field variables and structured variables.

N

Network	A network is the connection of devices to a common data path, which communicate with each other via a common protocol.
Network node	A node is a device with an address (164) on the Modbus Plus network.

Node address	The node address serves a unique identifier for the network in the routing path. The address is set directly on the node, e.g. with a rotary switch on the back of the module.

O

Operand	An operand is a Literal, a Variable, a Function call up or an Expression.
Operator	An operator is a symbol for an arithmetic or Boolean operation to be executed.
Output parameters (Output)	A parameter, with which the result(s) of the Evaluation of a FFB are returned.
Output/discretes (0x references)	An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of the State RAM.
Output/marker words (4x references)	An output/marker word can be used to save numerical data (binary or decimal) in the State RAM, or also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor	The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.
PLC	Programmable controller
Program	The uppermost Program organization unit. A program is closed and loaded onto a single PLC.
Program cycle	A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit	A Function, a Function block, or a Program. This term can refer to either a Type or an Item.
Programming device	Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.
Programming redundancy system (Hot Standby)	A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.
Project	General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation. General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.
Project data bank	The data bank in the Programming device, which contains the projection information for a Project.
Prototype data file (Concept EFB)	The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal

R

REAL

REAL stands for the data type "real". The input appears as Real literal or as Real literal with exponent. The length of the data element is 32 bit. The value range for variables of this data type reaches from 8.43E-37 to 3.36E+38.

Note: Depending on the mathematic processor type of the CPU, various areas within this valid value range cannot be represented. This is valid for values nearing ZERO and for values nearing INFINITY. In these cases, a number value is not shown in animation, instead NAN (**N**ot **A N**umber) oder INF (**INF**inite).

Real literal

Real literals function as the input of real values in the decimal system. Real literals are denoted by the input of the decimal point. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent

Real literals with exponent function as the input of real values in the decimal system. Real literals with exponent are denoted by the input of the decimal point. The exponent sets the key potency, by which the preceding number is multiplied to get to the value to be displayed. The basis may be preceded by a negative sign (-). The exponent may be preceded by a positive or negative sign (+/-). Single underline signs (_) between figures are not significant. (Only between numbers, not before or after the decimal poiont and not before or after "E", "E+" or "E-")

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference	Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM. 0x area = Discrete outputs 1x area = Input bits 3x area = Input words 4x area = Output bits/Marker words 6x area = Register in the extended memory **Note:** The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.
Register in the extended memory (6x reference)	6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.
RIO (Remote I/O)	Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.
RP (PROFIBUS)	RP = Remote Peripheral
RTU mode	Remote Terminal Unit The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.
Rum-time error	Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.

S

SA85 module	The SA85 module is a Modbus Plus adapter for an IBM-AT or compatible computer.
Section	A section can be used, for example, to describe the functioning method of a technological unit, such as a motor. A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section. Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.
Separator format (4:00001)	The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).
Sequence language (SFC)	The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.
Serial ports	With serial ports (COM) the information is transferred bit by bit.
Source code data file (Concept EFB)	The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.
Standard format (400001)	The five figure address is located directly after the first figure (the reference).
Standardized literals	If the data type for the literal is to be automatically determined, use the following construction: 'Data type name'#'Literal value'. Example INT#15 (Data type: Integer, value: 15), BYTE#00001111 (data type: Byte, value: 00001111) REAL#23.0 (Data type: Real, value: 23.0) For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0. Entering a comma will automatically assign the data type REAL.

State RAM	The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretes, input words, and discrete words are located in the state RAM.
Statement (ST)	Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.
Status bits	There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.
Step	SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.
Step name	The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears. The automatically generated step name always has the structure: S_n_m S = Step n = Section number (number running) m = Number of steps in the section (number running)
Structured text (ST)	ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.
Structured variables	Variables, one of which is assigned a Derived data type defined with STRUCT (structure). A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).
SY/MAX	In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are performed when highlighting and including in the I/O population of the Concept configuration.
Symbol (Icon)	Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB)	The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.
TIME	TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.
Time span literals	Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted. Example t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS
Token	The network "Token" controls the temporary property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.
Traffic Cop	The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.
Transition	The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB	User defined elementary functions/function blocks Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.
UDINT	UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1.
UINT	UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to (2exp16)-1.
Unlocated variable	Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names. Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.

V

Variables	Variables function as a data exchange within sections between several sections and between the Program and the PLC. Variables consist of at least a variable name and a Data type. Should a variable be assigned a direct Address (Reference), it is referred to as a Located variable. Should a variable not be assigned a direct address, it is referred to as an unlocated variable. If the variable is assigned a Derived data type, it is referred to as a Multi-element variable. Otherwise there are Constants and Literals.
Vertical format	Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning	When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output remains at "1".
WORD	WORD stands for the data type "Bit sequence 16". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 16 bit. A numerical range of values cannot be assigned to this data type.

Index