Concept
IEC block library
Part: ANA_IO
Volume 2

840 USE 504 00 eng Version 2.6

33002209.00

Table of Contents

About the book IX

The chapters marked gray are not included in this volume.

Part I General information on the ANA_IO block library 1
Overview 1

Chapter 1 Parameterizing functions and function blocks 3
Parameterizing functions and function blocks. 3

Chapter 2 Operating Analog Modules 7
Overview 7
Editing analog values 8
Scaling and configuration sections 8
Configuration EFBs 11
Scaling EFBs. 17
Debugging-EFBs. 18
Application example for Quantum 19

Chapter 3 Operating INTERBUS with Compact. .21
Diagram of the INTERBUS structure with EFBs 21

Chapter 4 Analog value processing with Momentum. 23
At a Glance 23
Procedure for Analog value processing with Momentum 24
Example analog value processing with Momentum 26

Part II EFB descriptions 29
Overview 29

Chapter 5 ACI030: Configuring the Quantum module ACI 030 00. 33

Chapter 6 ACI040: Configuring the Quantum module ACI 040 00. 37

Chapter 7 ACO020: Configuring the Quantum module ACO 020 00 . . . 41

Chapter 8 ACO130: Configuring the Quantum module ACO 130 00 . . . 45

Chapter 9 ADU204: Configuring the Compact module ADU 204 49

Chapter 10 ADU205: Configuring the Compact module ADU 205 53

Chapter 11 ADU206: Configuring the Compact module ADU 206/ADU 256 57

Chapter 12 ADU214: Configuring the Compact module ADU 214 61

Chapter 13 AII330: Configuring the Quantum module AII 330 00 65

Chapter 14 AII33010: Configuring the Quantum module AII 330 10 69

Chapter 15 AIO330: Configuring the Quantum module AIO 330 00 73

Chapter 16 AMM090: Configuring the Quantum module AMM 090 77

Chapter 17 ANA_16I: Configuring the module AAI 140 00 81

Chapter 18 ANA_4I_M: Configuring the module AAI 520 40 87

Chapter 19 ANA_4I_2O: Configuring the TIO-module BAM 096 00 95

Chapter 20 ANA_4I_2O_C: Configuring the TIO-module BAM 096 00 . . 105

Chapter 21 ANA_4I_2O_V: Configuring the TIO-module BAM 096 00 . . 109

Chapter 22 ANA_4O: Configuring the module BAO 126 00. 113

Chapter 23 ANA_8I: Configuring the module AAI 030 00, BAI 036 00 . . 119

Chapter 24 ARI030: Configuring the Quantum module ARI 030 10 125

Chapter 25 ATI030: Configuring the Quantum module ATI 030 00. 129

Chapter 26 AVI030: Configuring the Quantum module AVI 030 00 133

Chapter 27 AVO020: Configuring the Quantum module AVO 020 00. . . 137

Chapter 28 BKF_201: Configuring the Compact module BKF 201 139

Chapter 29 BNO_671: Configuring the TIO-module BNO 671 00 147

Chapter 30 COMPACT: Configuring a main rack 153

Chapter 31 DAU202: Configuring the Compact module DAU 202 / DAU 252 / DAU 282. 157

Chapter 32 DAU204: Configuring the Compact module DAU 204 161

Chapter 33 DAU208: Configuring the Compact module DAU 208 165

Chapter 34 DIG_16I: Configuring the TIO-module BDI 346 00 / 546 50 / 746 50 169

Chapter 35 DIG_16I_12O_MON: Configuring the module ADM 390 10 . . 175

Chapter 36 DIG_16I_16O: Configuring the TIO-module BDM 346 00 . . . 181

Chapter 37 DIG_16O: Configuring the TIO-modules BDO 346 00 / BDO 946 50. 187

Chapter 38 DROP: Configuring a I/O Station subrack 193

Chapter 39 I_DBSET: Writing internal data structure ANL_IN 197

Chapter 40 I_DEBUG: Monitoring internal data structure ANL_IN 199

Chapter 41 I_FILTER: Linearization for analog-inputs 201

Chapter 42 I_NORM: Standardized analog input. 207

Chapter 43 I_NORM_WARN: Standardized analog-input with warning status 209

Chapter 44 I_PHYS: Physical analog-input 213

Chapter 45 I_PHYS_WARN: Physical analog-input with warning-status 215

Chapter 46 I_RAW: Raw value analog input 219

Chapter 47 I_RAWSIM: Simulated raw value analog input. 221

Chapter 48 I_SCALE: Scaled analog input 223

Chapter 49 I_SCALE_WARN: Scaled analog input with warnings status 227

Chapter 50 I_SET: Set information from analog input channels 231

Chapter 51 IMIO_IN: Immediate I/O module input 239

Chapter 52 IMIO_OUT: Immediate I/O module output. 243

Chapter 53 MIX_4I_2O: Configuring the AMM module 090 00 247

Chapter 54 **NOA_611: Configuring the Quantum module NOA 611 00/NOA 611 10 . . . . 255**

Chapter 55 **O_DBSET: Write internal data structure ANL_OUT 261**

Chapter 56 **O_DEBUG: Monitoring internal data structure ANL_OUT . . 263**

Chapter 57 **O_FILTER: Linearization for analog outputs. 265**

Chapter 58 **O_NORM: Standardized analog output 271**

Chapter 59 **O_NORM_WARN: Standardized analog output with warning status 273**

Chapter 60 **O_PHYS: Physical analog output. 277**

Chapter 61 **O_PHYS_WARN: Physical analog output with warning-status 279**

Chapter 62 **O_RAW: Raw value analog output. 283**

Chapter 63 **O_SCALE: Scaled analog output 285**

Chapter 64 **O_SCALE_WARN: Scaled analog output with warnings status 289**

Chapter 65 **O_SET: Set information from analog output channels 293**

Chapter 66 **QPR_16I_12O: Configuring the TIO-module QPR 346 00 / 10 / 20 / 21 . . . . 301**

Chapter 67 **QUANTUM: Configuring a main rack 307**

Chapter 68 **R_INT_WORD: Type conversion (REAL -> INT -> WORD) . . 311**

Chapter 69 **R_UINT_WORD: Type conversion (REAL -> UINT -> WORD) 313**

Chapter 70 **SCALRTOW: Scaling (REAL -> WORD) 315**

Chapter 71 **SCALWTOR: Scaling (WORD -> REAL) 319**

Chapter 72 **UNI_I: Configuring universal TIO input modules 323**

Chapter 73 **UNI_I_O: Configuring universal TIO input/output modules 327**

Chapter 74 **UNI_O: Configuring universal TIO output modules 331**

Chapter 75 **W_INT_REAL: Type conversion (WORD -> INT -> REAL) . . 335**

Chapter 76 W_UINT_REAL: Type conversion (WORD -> UINT -> REAL) 337

Chapter 77 XBP: Configuring a primary backplane expander 339

Chapter 78 XDROP: Configuring a I/O Station Backplane 343

Glossary 347

Index i

About the book

At a Glance

Document Scope

This documentation is designed to help with the configuration of functions and function blocks.

Validity Note

This documentation applies to Concept 2.6 under Microsoft Windows 98, Microsoft Windows 2000, Windows XP and Microsoft Windows NT 4.x.

Note: There is additional up to date tips in the README data file in Concept.

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept User Manual	840 USE 503 00
Concept EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

I_DBSET: Writing internal data structure ANL_IN

39

Overview

At a Glance

This chapter describes the block I_DBSET.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	198
Representation	198

Brief description

Function description

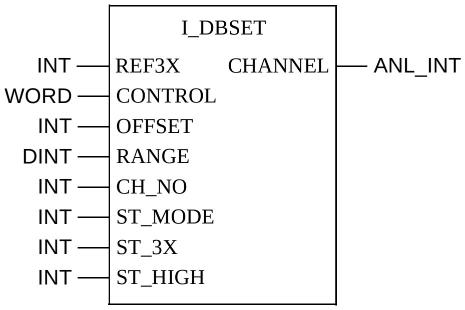
Note: The Function block is not usually needed.

The function block can be used to set information for the input channels (ANL_IN). EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
REF3X	INT	3x raw value register
CONTROL	WORD	Control word (internal use only)
OFFSET	INT	Input null shift
RANGE	DINT	Input range (resolution)
CH_NO	INT	Channel number
ST_MODE	INT	Status mode (internal use only)
ST_3X	INT	3x status register
ST_HIGH	INT	Identifies high byte or low byte of status register
CHANNEL	ANL_IN	Channel to be written

I_DEBUG: Monitoring internal data structure ANL_IN

40

Overview

At a Glance

This chapter describes the block I_DEBUG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Representation	200
Brief description	200

Brief description

Function description

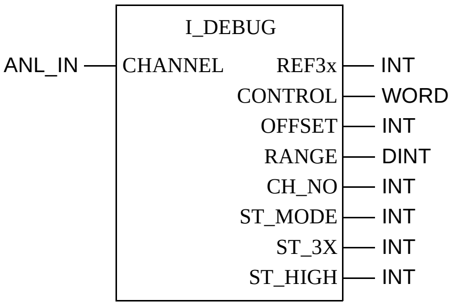
Note: The Function block is not usually needed.

The function block can be used to display information for the input channels (ANL_IN).
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	channel to be monitored
REF3X	INT	3x raw value register
CONTROL	WORD	Control word (internal use only)
OFFSET	INT	Input null shift
RANGE	DINT	Input range (resolution)
CH_NO	INT	Channel number
ST_MODE	INT	Status mode (internal use only)
ST_3X	INT	3x status register
ST_HIGH	INT	Identifies high byte or low byte of status register

I_FILTER: Linearization for analog-inputs

41

Overview

At a Glance

This chapter describes the block I_FILTER.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	202
Representation	202
Detailed description	203
Runtime error	205

Brief description

Function description

The function enables the adjustment of characteristic curves for analog input values.

3 different adjustments are available:

- Linearizing with square root (standardized range)
- Correction of the "Offset" (zero offset compensation)
- Correction of "Range" (gain)

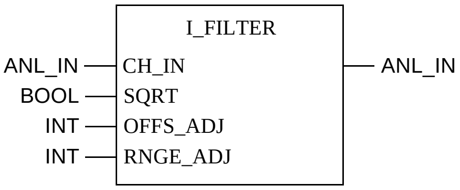
Note: Correction of the automatically set values for "Offset" and "Range" is not normally necessary.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

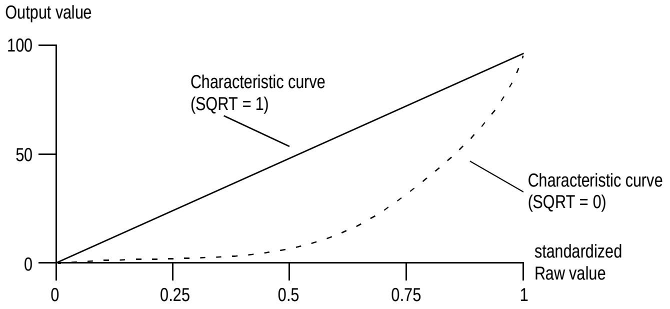
Block parameter description:

Parameter	Data type	Meaning
CH_IN	ANL_IN	Input value
SQRT	BOOL	Square root filter 1: Filter active 0: Filter inactive
OFFS_ADJ	INT	Adjusting offset
RNGE_ADJ	INT	Adjusting gain
OUT	ANL_IN	Output value

Detailed description

Linearizing with square root (standardized range)

Use the parameter SQRT to linearize an analog input value.
The square root filter acts according to the following functions:
f(0) = 0, f(0.5) = 0.707, f(1) = 1.
Characteristic curve of the square root filter



Correction of the "Offset" (zero offset compensation)

The OFFS_ADJ parameters can be used to modify (adjust) the calculated offset value of the output

Note: Correction of the automatically set value (OFFS_ADJ = 0) is not normally necessary. Nevertheless, if corrections are made, they should be monitored using the I_DEBUG Function block, because there will be a modification of the ANL_IN data type (of the output).

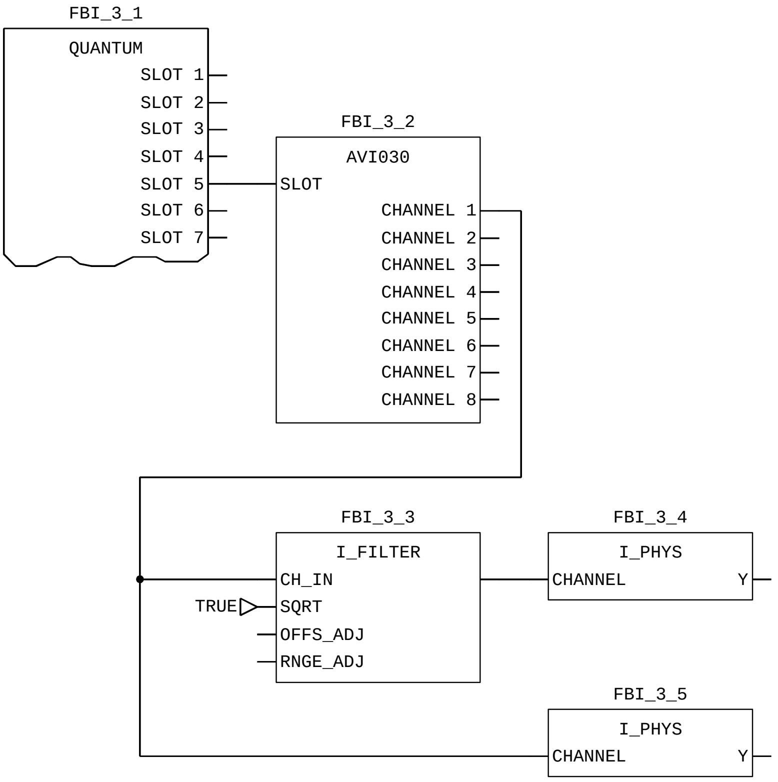
Correction of "Range" (gain)

The RNGE_ADJ parameter can be used to modify (adjust) the calculated gain of the output.

Note: Correction of the automatically set value (RNGE_ADJ = 0) is not normally necessary. If corrections are made, they should be monitored using the I_DEBUG Function block, because there will be a modification of the ANL_IN data type (of the output).

Example

Structure with I_FILTER



The outputs OFFS_ADJ and RNGE_ADJ of the I_FILTER (FBI_3_3) Function block are not used. They are therefore set per default to "0".
The following values apply for function block I_PHYS (FBI_3_4):

Input values **(AVI030 10 V)**	**Output values** **(I_PHYS)**
0 V	0.0
2.5 V	5.0
5 V	7.07
10 V	10.0

The following values apply for function block I_PHYS (FBI_3_5):

Input values **(AVI030 10 V)**	**Output values** **(I_PHYS)**
0 V	0.0
2.5 V	2.5
5 V	5.0
10 V	10.0

Runtime error

Runtime error

An error message appears if the input channel has not been configured. In this case, please check the connected I/O module EFB.

I_NORM: Standardized analog input

42

Overview

At a Glance

This chapter describes the block I_NORM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	208
Representation	208
Runtime error	208

Brief description

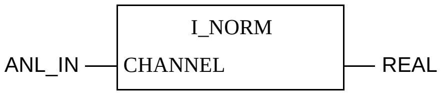
Function description

The function converts data from 16 bit integer format into REAL floating-point format. The configured integer input value is displayed with a floating-point value in the range of 0.0 to 1.0. If there are warning ranges for the current data format (e.g. 16 bit, +/- 10 V), the floating point value can be expanded (e.g. 1.016)
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
OUT	REAL	Normalized value

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- with an input value underflow (for example, -1 Volt instead of 0 ... 5 Volt).
- with an input value overflow (for example, 6 Volt instead of 0 ... 5 Volt).

Note: To evaluate the status information for the I/O module, use the function block I_NORM_WARN.

I_NORM_WARN: Standardized analog-input with warning status

43

Overview

At a Glance

This chapter describes the block I_NORM_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	210
Representation	210
Runtime error	211

Brief description

Function description

The function block converts data from 16 bit integer format into REAL floating-point format. The configured integer input value is displayed with a floating-point value in the range of 0.0 to 1.0. If there are warning ranges for the current data format (e.g. 16 bit, +/- 10 V), the floating point value can be expanded (e.g. 1.016)
In addition the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

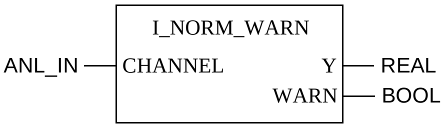
Note: This function block is not compatible with the ADU2xx function for Compact (the I_NORM Function block should be used instead). The I_NORM_WARN function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	**Data type**	**Meaning**
CHANNEL	ANL_IN	Input value
Y	REAL	Normalized value
WARN	BOOL	0: no status warning on the connected analog input EFB 1: status warning on the connected analog input EFB

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- with an input value overflow (outside the warning range, e.g. 6Volts instead of 0 ... 5Volts)
- if the connected analog input EFB is unable to generate status information, and the WARN output can, therefore, never become active. In this case, please use the I_NORM Function block.

I_PHYS: Physical analog-input

44

Overview

At a Glance

This chapter describes the block I_PHYS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	214
Representation	214
Runtime error	214

Brief description

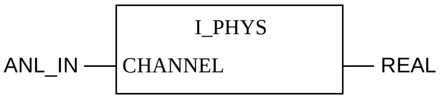
Function description

The Function outputs analog input values (voltage, current or temperature) as physical values in REAL floating-point format.
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
OUT	REAL	Physical value

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- with an input value underflow (for example, -1 Volt instead of 0 ... 5 Volt).
- in the case of an input value overflow (for example, 6 Volt instead of 0 ... 5 Volt).

Note: To evaluate the status information for the I/O module, use the Function block I_PHYS_WARN.

I_PHYS_WARN: Physical analog-input with warning-status

45

Overview

At a Glance

This chapter describes the block I_PHYS_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	216
Representation	216
Runtime error	217

Brief description

Function description

The Function block provides analog input values (voltage, current or temperature) as physical values in REAL floating-point format.
In addition the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

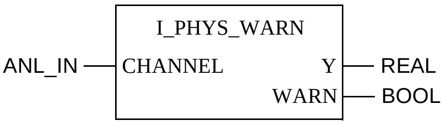
> **Note:** This function block is not compatible with the ADU2xx function for Compact (the I_PHYS function block should be used instead). The I_PHYS_WARN Function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
Y	REAL	Physical value
WARN	BOOL	0: no status warning on the connected analog input EFB 1: status warning on the connected analog input EFB

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- with an input value underflow (outside the warning range, e.g. -1Volt instead of 0 ... 5Volts).
- with an input value overflow (outside the warning range, e.g. 6Volts instead of 0 ... 5Volts).
- if the connected analog input EFB is unable to generate status information, and the WARN output can, therefore, never become active. In this case, please use the I_PHYS Function block.

I_RAW: Raw value analog input

46

Overview

At a Glance

This chapter describes the block I_RAW.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	220
Representation	220
Runtime error	220

Brief description

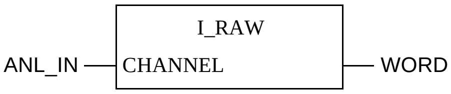
Function description

The function provides analog input values as raw values of the WORD data type. EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
OUT	WORD	Raw value

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- if there is an input range violation.

I_RAWSIM: Simulated raw value analog input

47

Overview

At a Glance

This chapter describes the block I_RAWSIM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	222
Representation	222
Runtime error	222

Brief description

Function description

The Function block simulates raw value analog inputs on 3x registers. The function block acts to supplement for the reference data editor where 3x registers cannot be written.

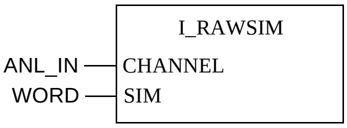
Note: Specify the processing sequence for the function blocks in a way that ensures the I_RAWSIM Function block will be executed before all the other function blocks which read the simulated raw value. To do this, connect the ENO output of the I_RAWSIM with the EN inputs of all the function blocks which read the simulated raw value.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Simulated raw value
SIM	WORD	Input value

Runtime error

Runtime error

An Error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.

I_SCALE: Scaled analog input

48

Overview

At a Glance

This chapter describes the block I_SCALE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	224
Representation	224
Runtime error	225

Brief description

Function description

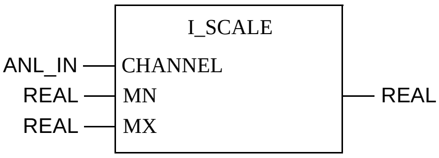
The function converts data from 16 bit integer format into REAL floating-point format. The scaling inputs MN and MX predefine the value range for the output. MN corresponds to 0 percent and MX to 100 percent. The integer input value is displayed in the floating-point range. If there are warning ranges for the current data format (e.g. 16 bit, +/- 10 V), the floating point value can be expanded to over 100 percent (e.g. 101.8 percent)
EN and ENO can be projected as additional parameters.

Note: The I_SCALE function can not be used to scale temperature measurements. Please use the I_PHYSfunction to scale temperature measurements:.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
MN	REAL	Scaling input, 0 percent
MX	REAL	Scaling input, 100 percent
OUT	REAL	Scaled value

Runtime error

Runtime error

An error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- in the case of input value underflow (for example, -1 Volt instead of 0 ... 5 Volt).
- in the case of input value overflow (for example, 6 Volt instead of 0 ... 5 Volt).

Note: To evaluate the status information for the I/O module, use the I_SCALE_WARN function block.

I_SCALE_WARN: Scaled analog input with warnings status

49

Overview

At a Glance

This chapter describes the block I_SCALE_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	228
Representation	229
Runtime error	229

Brief description

Function description

The function block converts data from 16 bit integer format into REAL floating-point format. The scaling inputs MN and MX predefine the value range for the output. MN corresponds to 0 percent and MX to 100 percent. The integer input value is displayed in the floating-point range. If there are warning ranges for the current data format (e.g. 16 bit, +/- 10 V), the floating point value can be expanded to over 100 percent (e.g. 101.6 percent)
In addition the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

Note: This function block is not compatible with the ADU2xx function for Compact (the I_SCALE function block should be used instead). The I_SCALE_WARN function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

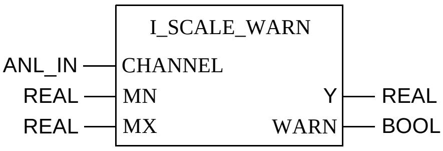
The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be configured as additional parameters.

Note: The I_SCALE_WARN function can not be used to scale temperature measurements. Please use the I_PHYS_WARNfunction to scale temperature measurements.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input value
MN	REAL	Scaling input, 0 percent
MX	REAL	Scaling input, 100 percent
Y	REAL	Scaled value
WARN	BOOL	0: no status warning on the connected analog input EFB 1: status warning on the connected analog input EFB

Runtime error

Runtime error

An error message appears,

- if the input channel is not configured. In this case, please check the connected I/O module EFB.
- with an input value underflow (outside the warning range, e.g. -1Volt instead of 0 ... 5Volts).
- with an input value overflow (outside the warning range, e.g. 6Volts instead of 0 ... 5Volts).
- if the connected analog input EFB is unable to generate status information, and the WARN output can, therefore, never become active. In this case, please use the I_SCALE function block.

I_SET: Set information from analog input channels

50

Overview

At a Glance

This chapter describes the block I_SET.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	232
Representation	233
Detailed description	234
Supported Value Ranges	236
Runtime error	237

Brief description

Function description

The function block sets the informations for the analog input channels (ANL_IN). This block enables all scaling blocks of this library to be used.

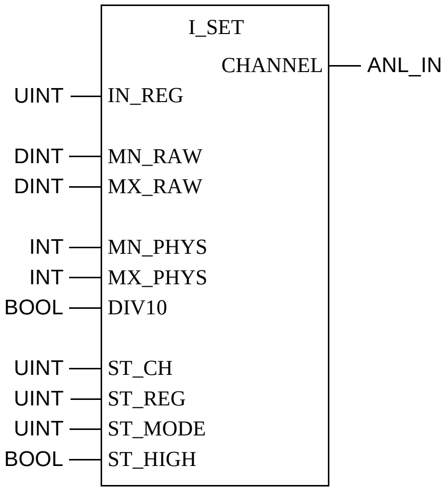
Note: The function block is only required if there is no specific block for a specific analog module available.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN_REG	UINT	Number of the raw value register (3X)
MN_RAW	DINT	0 % raw value (e.g. 768)
MX_RAW	DINT	100% raw value (e.g. 64768)
MN_PHYS	INT	lowest input value (e.g. -10 V as -10)
MX_PHYS	INT	greatest input value (e.g. +10 V as 10)
DIV10	BOOL	MN_PHYS and MX_PHYS divided by 10
ST_CH	UINT	channel number (1n) (e.g. 4)
ST_REG	UINT	Number of the status register (3X)
ST_MODE	UINT	Status mode (e.g. 1=AVI_STATUS_MODE)
ST_HIGH	BOOL	Status byte found in highbyte of the register
CHANNEL	ANL_IN	channel information to be described

Detailed description

Area of application

The function block can be used in three areas:

1. Raw value scaling, with the block: I_NORM and I_SCALE
2. Scaling in physical units, with the I_PHYS block
3. Evaluation of error information, with the blocks I_NORM, I_SCALE and I_PHYS and additional evaluation of status information (warnings) using the I_..._WARN block

Basic circuit connections

The input IN_REG must always be connected with the number of an input word (3x).

Raw value scaling

For raw value scaling, the inputs MN_RAW (minimum raw value, corresponds to 0%) and MX_RAW (maximum raw value, corresponds to 100%) must also be connected.

Scaling in physical units

For scaling in physical units the inputs MN_PHYS and MX_PHYS must also be connected.
DIV10 is an auxiliary input in the range 0.2 V ... 1 V floating point format to avoid.
Set MN_PHYS=2, MX_PHYS=10 and DIV10=1 for this range.
For most ranges this input can remain open (or be assigned 0).
e.g. +/-20 mA: here is MN_PHYS=-20, MX_PHYS=20
The input value ranges supported by I_SET can be found in the section *Supported Value Ranges, p. 236*.

Evaluation of error information

For the evaluation of error information the inputs ST_CH, ST_REG and ST_MODE and ST_HIGH must also be configured.
ST_HIGH is an auxiliary input in case the status byte (status information) is located in the registers high byte. For most ranges this input can remain open (or be assigned FALSE).
The input channel number (1 ... n) is given to ST_CH.
If ST_CH is entered, ST_REG and ST_MODE must also be entered.
ST_REG must be connected with the number of an input word (3X), where the status information is located (error and/or warnings).
ST_MODE determines how the status word is evaluated.
The following 8 modes are defined:

Value	Mode	see also module description for
1	AVI_STATUS_MODE	AVI030
2	ACI_STATUS_MODE	ACI030
3	ACO_STATUS_MODE	ACO030
4	ADU_STATUS_MODE	ADU204
5	DAU204_STATUS_MODE	DAU204
6	ADU205_STATUS_MODE	ADU205
7	AMM090_STATUS_MODE	AMM090
8	ADU214_STATUS_MODE	ADU214

Supported Value Ranges

Voltage

Unipolar

Value range	MN_PHYS	MX_PHYS	DIV10
0 ... 0.5 V	0	5	1
0 ... 1.0 V	0	10	1
0 ... 5.0 V	0	5	0
0 ... 10 V	0	10	0
0 ... 20 V	0	20	0
0,1 ... 0.5 V	1	5	1
0,2 ... 1.0 V	2	10	1
1,0 ... 5.0 V	1	5	0
2,0 ... 10, 0 V	2	10	0

Bipolar

Value range	MN_PHYS	MX_PHYS	DIV10
+/- 25 mV	-25	25	0
+/-100 mV	-100	100	0
+/-0.5 V	-5	5	1
+/- 1 V	-1	1	0
+/-5 V	-5	5	0
+/-10 V	-10	10	0
+/-20 V	-20	20	0

Current

Unipolar

Value range	MN_PHYS	MX_PHYS	DIV10
0 .. 20 mA	0	20	0
4 ... 20 mA	4	20	0

Bipolar

Value range	MN_PHYS	MX_PHYS	DIV10
+/-20 mA	-20	20	0
+/-40 mA	-40	40	0

Resistance

Unipolar

Value range	MN_PHYS	MX_PHYS	DIV10
0 .. 400 Ω	0	400	0
0 .. 500 Ω	0	500	0
0 .. 766,6 Ω	0	7666	1
0 .. 1 kΩ	0	1000	0
0 .. 2 kΩ	0	2000	0
0 .. 4 kΩ	0	4000	0

Runtime error

Runtime error

The following error messages can be triggered:

Error message	Meaning
E_EFB_USER_ERROR_1	The input IN_REG is not connected with the number of an input word (3x).
E_EFB_USER_ERROR_2 with the parameters of the faulty number	The input IN_REG is connected with an invalid number of an input word (3x).
E_EFB_USER_ERROR_3 with parameter MN_RAW	MN_RAW ≥ MX_RAW)
E_EFB_USER_ERROR_4 with parameter MN_PHYS	Unknown value for MN_PHYS
E_EFB_USER_ERROR_5 with parameter MX_PHYS	Unknown value for MX_PHYS
E_EFB_USER_ERROR_11	ST_REG not entered
E_EFB_USER_ERROR_12	ST_REG too large
E_EFB_USER_ERROR_13	ST_CH not entered

IMIO_IN: Immediate I/O module input

51

Overview

At a Glance

This chapter describes the block IMIO_IN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	240
Representation	240
Detailed description	241
Runtime error	241

Brief description

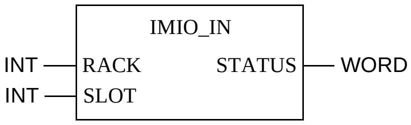
Function description

This Function block reads in I/O module signals immediately during processing. The input module must be in the local rack of the PLC.
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
RACK	INT	Subrack number (Quantum: 1; Compact: 1 ... 4)
SLOT	INT	Slot number (Quantum: 1...16; Compact: 1 ... 5)
STATUS	WORD	Status report

Detailed description

Detailed description

The input of signals takes place directly during block processing as well as during normal I/O processing at the end of a cycle.
The input module must be in the local rack of the PLC. It must also be entered into the I/O map of its configuration. The I/O module is addressed using subrack number and slot number.

Parameter description

The STATUS parameter may contain the following messages:

Status	Meaning
0000	Operation OK
2001	invalid operation type (e.g. the I/O module addressed is not an input module)
2002	Invalid rack or slot number (I/O map in the configurator contains no module entry for this slot)
2003	invalid slot number
F001	Module not OK

Runtime error

Runtime error

The ENO parameter can be used for error display:

ENO	Meaning
1	Operation OK (STATUS equals "0")
0	Operation OK (STATUS not equal to "0")

IMIO_OUT: Immediate I/O module output

52

Overview

At a Glance

This chapter describes the block IMIO_OUT.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	244
Representation	244
Detailed description	245
Runtime error	245

Brief description

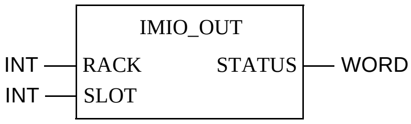
Function description

This Function block supplies the I/O module signals immediately during processing. The output module must be in the local rack of the PLC.
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
RACK	INT	Subrack number (Quantum: 1; Compact: 1 ... 4)
SLOT	INT	Slot number (Quantum: 1...16; Compact: 1 ... 5)
STATUS	WORD	Status report

Detailed description

Detailed description

The output of signals takes place immediately during block processing as well as during normal I/O processing at the end of a cycle.
The output module must be in the local rack of the PLC. It must also be entered into the I/O map of its configuration. The I/O module is addressed using subrack number and slot number.

Parameter description

Status report STATUS
The STATUS parameter may contain the following messages:

Status	Meaning
0000	Operation OK
2001	invalid operation type (e.g. the I/O module addressed is not an input module)
2002	Invalid rack or slot number (I/O map in the configurator contains no module entry for this slot)
2003	invalid slot number
F001	Module not OK

Runtime error

Runtime error

The ENO parameter can be used for error display:

ENO	Meaning
1	Operation OK (STATUS equals "0")
0	Operation OK (STATUS not equal to "0")

MIX_4I_2O: Configuring the module AMM 090 00

53

Overview

At a Glance

This chapter describes the block MIX_4I_2O.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	248
Representation	248
Detailed description	250

Brief description

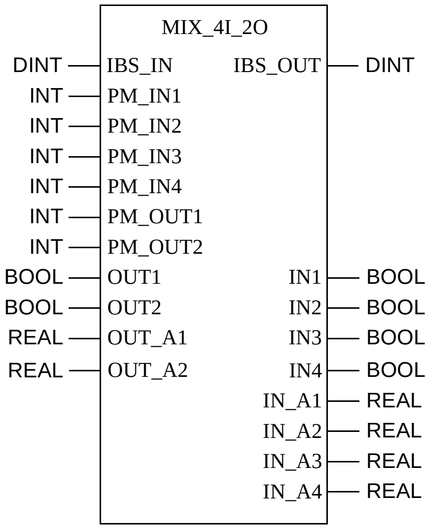
Function description

The MIX_4I_2O Function block is a software connection to an INTERBUS Momentum/IS 170 AMM 090 00 hardware module.
The function block has 4 analog inputs and 2 analog outputs, as well as 4 binary inputs and 2 binary outputs. The function block must be parametered in the same way as the hardware module.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IBS_IN	DINT	Incoming INTERBUS
PM_IN1	INT	Parameter input 1
:	:	:
PM_IN4	INT	Parameter input 4
PM_OUT1	INT	Parameter output 1
PM_OUT2	INT	Parameter output 2
OUT1	BOOL	Digital output 1
OUT2	BOOL	Digital output 2
OUT_A1	REAL	Analog output 1 of the module
OUT_A2	REAL	Analog output 2 of the module
IBS_OUT	DINT	Outgoing INTERBUS
IN1	BOOL	Digital input 1
:	:	:
IN4	BOOL	Digital input 4
IN_A1	REAL	Analog input 1 of the module
:	:	:
IN_A4	REAL	Analog input 4 of the module

Detailed description

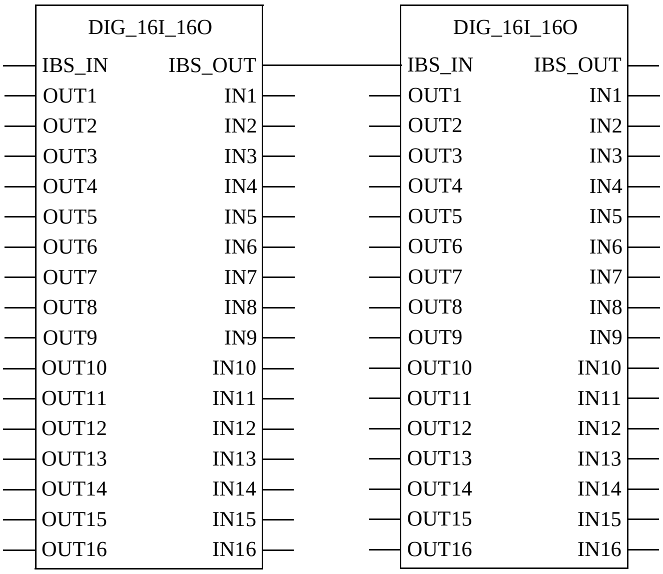
Detailed description

The function block occupies 5 input words and 5 output words in the Status-RAM.

Parameter description - inputs

IBS_IN

IBS_IN = Connection for the incoming remote bus part of INTERBUS
On the hardware, the male connector is on the top left of the module.
The module is connected here to the outgoing remote bus (IBS_OUT) of the master (1st module on the bus) or the preceding module (see diagram). The link can be made via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two bus devices.
Connection of two INTERBUS modules



PM_INx

PM_INx = Parameters for the input channels
x stands for the digit 1 to 4 which indicates the particular input channel.
These parameters are used to parameter the input channels.
The meaning of the parameter values can be found in the table below.

Parameter value	Meaning
2	+/- 20mA (+/- 5 V, when divided by 4)
3	+/- 10V
4	Channel inactive
10	420mA (1...5 V, when divided by 4)

A = Output after bus interrupt

Note: All other parameter values are reserved.

Example:
Input 3 should 4 ...20mA.
PM_IN3 = "10"

Note: The reserved parameter codes are not accepted by the module, i.e. the last parameter used will still apply. The default parameters apply until a valid new parameter is entered.

PM_OUTx

PM_OUTx = Parameters for the output channels
x stands for the digit 1 or 2 which indicates the particular output channel.

Example:
PM_OUT2 = Parameters for output channel 2
These parameters are used to parameterize the output channels.
The meaning of the parameter values can be found in the table below.

Parameter value	Meaning
1	0...20mA; Timeout A: 0mA
3	+/- 10V; Timeout A: 0V
4	Channel inactive (default)
5	0...20mA; Timeout A: 20mA
7	+/- 10V; Timeout A: +10V
9	0...20mA; Timeout A: freezes
11	+/- 10V; Timeout A: freezes

A = Output after bus interrupt

Note: All other parameter values are reserved.

Example:
Output 1 should be 0 ...20mA and set to 0mA for bus failure.
PM_OUT1 = "1"

Note: The reserved parameter codes are not accepted by the module, i.e. the last parameter used will still apply. The default parameters apply until a valid new parameter is entered.

OUTx

OUTx = Digital output channel x
x stands for the digit 1 or 2 which indicates the particular output channel.
The binary values to be produced via the INTERBUS module are supplied to the process via the relevant output (OUTx).

OUT_Ax

OUT_Ax = Analog output channel x
x stands for the digit 1 or 2 which indicates the particular output channel.
The analog values to be produced via the INTERBUS module are supplied to the process via the relevant output (OUTx).

> **Note:** The values to be applied here are standardized, i.e. given as voltage in volts or as current in milliamperes. The input as current or voltage depends on the parametering of the particular channel.

Parameter description - Outputs

IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
In the hardware, the male connector is located in the top right of the module. This is where the module is connected to the incoming remote bus (IBS_IN) of the next module via either a line or a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two INTERBUS modules.

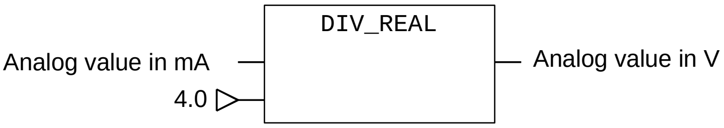
INx

INx = Digital input x
x stands for the digit 1 to 4 which indicates the corresponding input.
Binary process values are read into the INTERBUS module via the relevant input (INx).

IN_Ax

IN_Ax = Analog input channel x
x stands for the number between 1 and 4 designating the corresponding input channel. The analog process values of the INTERBUS module are read via the corresponding input (INx).

> **Note:** The values to be applied here are standardized, i.e. given as voltage in volts or as current in milliamperes. The input as current or voltage depends on the parametering of the particular channel. If a channel is parametered in the +/-5V- or 1...5V range, the incoming values are given in milliamperes. To obtain these values as voltage, divide by 4.0
> Scaling an analog value



NOA_611: Configuring the Quantum module NOA 611 00/ NOA 611 10

54

Overview

At a Glance

This chapter describes the block NOA_611.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	256
Representation	256
Detailed description	258
Runtime error	259

Brief description

Function description

The Function block NOA_611 is the software connection for an INTERBUS NOA 611 10 master module. It ensures that the data on the INTERBUS is transferred to and read by the corresponding module. The NOA 611 10 controls the bus and monitors operational performance.

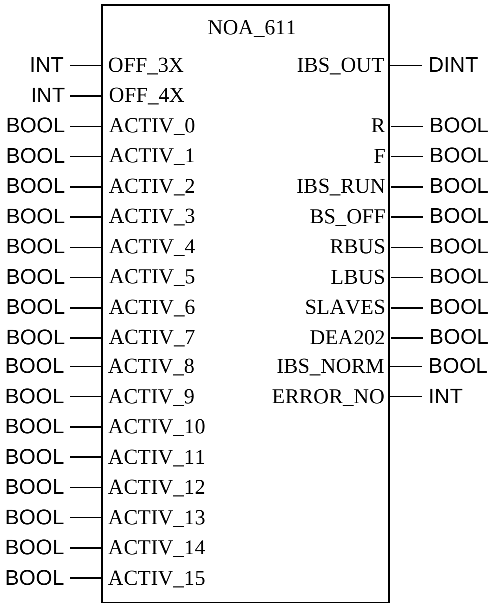
The NOA 611 10 occupies 267 input words and 264 output words in the PLC memory. The first input word and the first output word are occupied by the NOA 611 10 itself; it contains the NOA 61110 control bits and status bits. The remaining 256 input words and 256 output words contain the I/O data for the INTERBUS modules.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
OFF_3X	INT	Offset for 3x address
OFF_4X	INT	Offset for 4x address
ACTIV_0	BOOL	Starts routines which are stored under active bit 0
:	:	:
ACTIV_15	BOOL	Starts routines which are stored under active bit 15
IBS_OUT	DINT	Outgoing INTERBUS
R	BOOL	Master ready
F	BOOL	Error on NOA
IBS_RUN	BOOL	Process data is being exchanged
BS_OFF	BOOL	One or more bus segments are switched off
RBUS	BOOL	Error on remote bus
LBUS	BOOL	Error on local bus
SLAVES	BOOL	INTERBUS device indicates error
DEA202	BOOL	Initialization error on DEA202
IBS_NORM	BOOL	INTERBUS is standardized. All outputs = 0.
ERROR_NO	INT	Number of the faulty INTERBUS module

Detailed description

Parameter description - inputs

OFF_3X and OFF_4X

The parameters of the inputs for the function block are assigned the following module functions.
OFF_3X = Offset 3x address
OFF_4X = Offset 4x address
On the function block, the relevant address offsets for 3x and 4x addresses are given at the two inputs.

Example:
The NOA 611 10 is entered in the PLC configurator, as shown in the table.

Slot	Module	Detected	In.Ref.	In.End	Out.Ref.	Out.End.
1	NOA-611-10		300020	300286	400020	400283

If the initial addresses for the NOA 611 10 are "3:20" for the input words and "4:20" for the output words, then

- OFF_3X = 20 and
- OFF_4X = 20.

ACTIV_x

ACTIV_x = Routine call of active bit x
x stands for the digit 0 to 15 which indicates the particular active bit. A positive transition on ACTIV_x calls the routine stored under ACTIV_x.

Parameter description - Outputs

IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
INTERBUS connection on the front panel of the NOA 611. From here, the first module on INTERBUS is connected to the master, via either a line or a variable. For the hardware, the type of connection corresponds to the INTERBUS cable from the master to the first module on INTERBUS.

R

R = NOA 611 10 ready
The NOA 611 10 master module is ready and error-free.

F

F = NOA 611 10 faulty
The NOA 611 10 master module is faulty.

IBS_RUN

IBS_RUN = INTERBUS data is being transmitted
INTERBUS is operating without error, process data is being exchanged.

BS_OFF

BS_OFF = INTERBUS segment switched off
One or more bus segments on INTERBUS are switched off.

RBUS

RBUS = Remote bus error
An error has occurred on the remote bus.

LBUS

LBUS = Local bus error
An error has occurred on a local bus.

SLAVES

SLAVES = Error on an INTERBUS device
Indicates that a device on INTERBUS is faulty.

DEA202

DEA202 = Error on DEA202
Indicates an initialization error on the DEA202.

IBS_NORM

IBS_NORM = INTERBUS standardized
The bus is standardized. All outputs on INTERBUS take the value "0".

ERROR_NO

ERROR_NO = Device error number
Indicates the device number of a faulty device on the bus.
Example:

- Bus connection between device 1 and device 2 interrupted.
  Display: 2
- Voltage failure on device 1.
  Display: 1

Runtime error

Runtime error

An error message (E_INPUT_VALUE_OUT_OF_RANGE) appears if the offset for the 3x or 4x addresses is less than 0 or greater than the maximum permissible value.

O_DBSET: Write internal data structure ANL_OUT

55

Overview

At a Glance

This chapter describes the block O_DBSET.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	262
Representation	262

Brief description

Function description

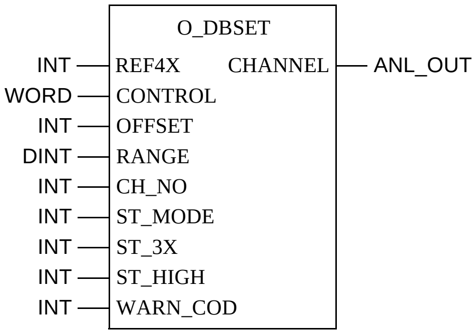
Note: The Function block is not usually needed.

The function block can be used to enter information for the output channels (ANL_OUT).
EN and ENO can be projected as additional parameters.

Representation

Symbol

Module representation:



Parameter description

Module parameter description:

Parameter	Data type	Meaning
REF4X	INT	4x raw value register
CONTROL	WORD	Control word (internal use only)
OFFSET	INT	Input null shift
RANGE	DINT	Input range (resolution)
CH_NO	INT	Channel number
ST_MODE	INT	Status mode (internal use only)
ST_3X	INT	3x status register
ST_HIGH	INT	Identifies high byte or low byte of status register
WARN_COD	INT	Warning mode (internal use only)
CHANNEL	ANL_OUT	Channel to be written

O_DEBUG: Monitoring internal data structure ANL_OUT

56

Overview

At a Glance

This chapter describes the block O_DEBUG.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	264
Representation	264

Brief description

Function description

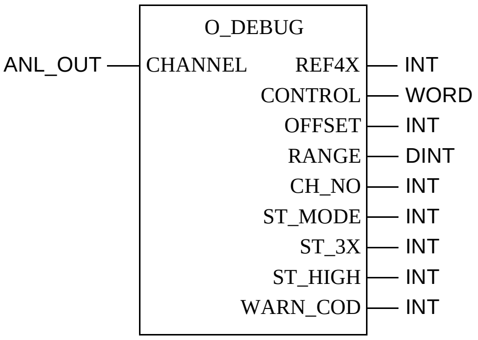
Note: The Function block is not usually needed.

The function block can be used to display information for the output channels (ANL_OUT).
EN and ENO can be projected as additional parameters.

Representation

Symbol

Module representation:



Parameter description

Module parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	channel to be monitored
REF3X	INT	4x raw value register
CONTROL	WORD	Control word (internal use only)
OFFSET	INT	Input null shift
RANGE	DINT	Input range (resolution)
CH_NO	INT	Channel number
ST_MODE	INT	Status mode (internal use only)
ST_3X	INT	3x status register
ST_HIGH	INT	Identifies high byte or low byte of status register
WARN_COD	INT	Warning mode (internal use only)

O_FILTER: Linearization for analog outputs

57

Overview

At a Glance

This chapter describes the block O_FILTER.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	266
Representation	266
Detailed description	267
Runtime error	269

Brief description

Function description

The function enables the adjustment of characteristic curves for analog raw values. Different adjustments are available:

- Linearizing with square root (standardized range)
- Correction of the "Offset" (zero offset compensation)
- Correction of "Range" (gain)

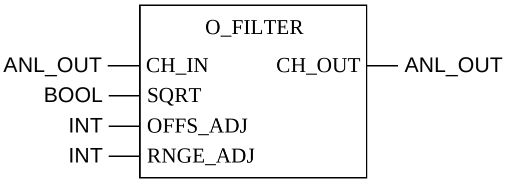
> **Note:** Correction of the automatically set values for "Offset" and "Range" is not normally necessary.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

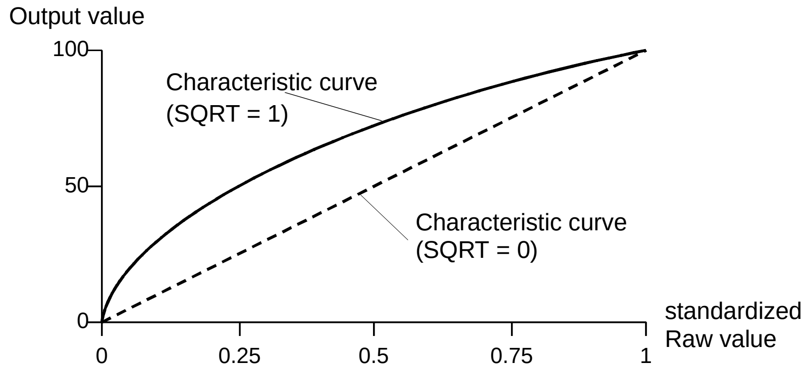
Block parameter description:

Parameter	Data type	Meaning
CH_IN	ANL_OUT	Raw value
SQRT	BOOL	Square root filter 1: Filter active 0: Filter inactive
OFFS_ADJ	INT	Adjusting offsets
RNGE_ADJ	INT	Adjusting gain
CH_OUT	ANL_OUT	Output value

Detailed description

Adjustment with square root (standardized range)

The SQRT parameter can be used to adjust an analog output value. The square root filter acts according to the following functions:
f(0) = 0, f(0.5) = 0.707, f(1) = 1.
Characteristic curve of the square root filter



Correction of the "Offset" (zero offset compensation)

Use the parameter OFFS_ADJ to modify (adjust) the calculated offset value of the output CH_OUT.

Note: Correction of the automatically set value (OFFS_ADJ = 0) is not normally necessary. If corrections are made, they should be monitored using the O_DEBUG Function block, because there will be a modification of the ANL_OUT data type (of the output).

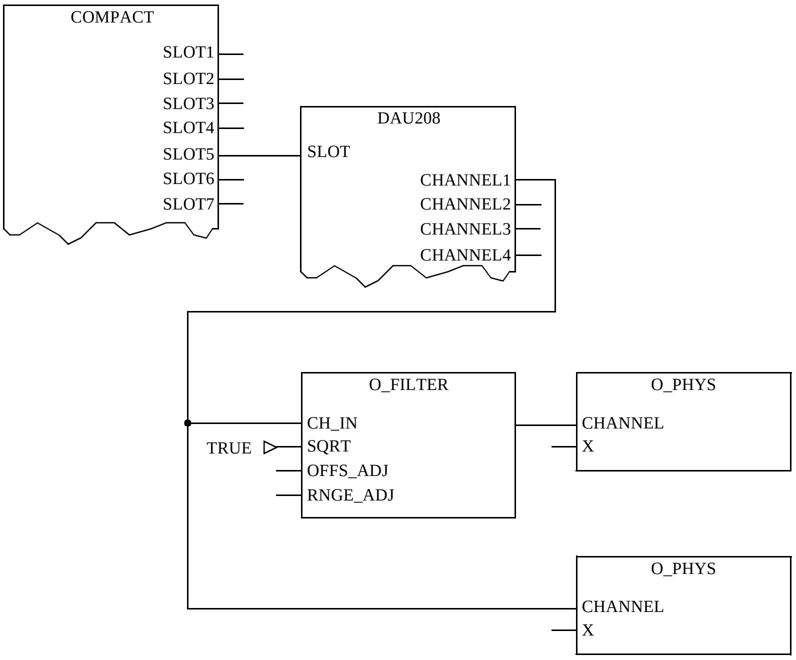
Correction of "Range" (gain)

The RNGE_ADJ parameter can be used to modify (adjust) the calculated gain of the output.

Note: Correction of the automatically set value (RNGE_ADJ = 0) is not normally necessary. If corrections are made, they should be monitored using the O_DEBUG Function block, because there will be a modification of the ANL_OUT data type (of the output)).

Example

Structure with O_FILTER



The outputs OFFS_ADJ and RNGE_ADJ of the O_FILTER (FBI_3_3) Function block are not used. They are set to "0"by default.
The following values apply for function block O_PHYS (FBI_3_4):

Input values **(DAU208 10 V)**	**Output values** **(O_PHYS)**
0 V	0.0
2.5 V	5.0
5 V	7.07
10 V	10.0

The following values apply for function block O_PHYS (FBI_3_5):

Input values **(DAU208 10 V)**	**Output values** **(O_PHYS)**
0 V	0.0
2.5 V	2.5
5 V	5.0
10 V	10.0

Runtime error

Runtime error

An error message appears if the input channel has not been configured. In this case, please check the connected I/O module EFB.

O_NORM: Standardized analog output

58

Overview

At a Glance

This chapter describes the block O_NORM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	272
Representation	272
Runtime error	272

Brief description

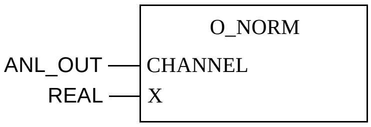
Function description

The Function block outputs values from floating point format REAL as analog values in 16 bit integer format. The floating point value in the range of 0.0 to 1.0 is displayed onto the configured integer output value.
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output value
X	REAL	Normalized value

Runtime error

Runtime error

An error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- with an output value underflow (arithmetic) (for example, -0.1 V instead of 0 ... 1.0 Volt)
- with an output value overflow (arithmetic) (for example, 1.1 instead of 0 ... 1.0 Volt)

Note: To evaluate the status information for the I/O module, use the O_NORM_WARN function block.

O_NORM_WARN: Standardized analog output with warning status

59

Overview

At a Glance

This chapter describes the block O_NORM_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	274
Representation	274
Runtime error	275

Brief description

Function description

The Function block outputs values from floating point format REAL as analog values in 16 bit integer format. The floating point value in the range of 0.0 to 1.0 is displayed onto the configured integer output value.
In addition the function block at the WARN_NEG and WARN_POS outputs indicate whether a status warning has occurred in the connected analog output EFB.

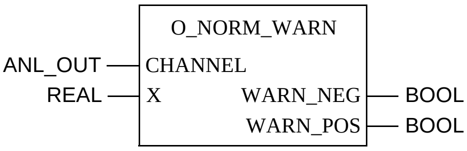
Note: This function block is not compatible with the ADU2xx and DAU2xx functions for Compact (the O_NORM Function block should be used instead). The O_NORM_WARN Function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output value
X	REAL	Normalized value
WARN_NEG	BOOL	0: no output value underflow at the closed analog output EFB 1: output value underflow at the closed analog output EFB
WARN_POS	BOOL	0: no output value overflow at the closed analog output EFB 1: output value overflow at the closed analog output EFB

Runtime error

Runtime error

An error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- with an output value underflow (arithmetic) (outside the warning range, eg. -0.1V instead of 0 ... 1.0V) 1.0Volt)
- with an output value overflow (arithmetic) (outside the warning range, eg. 1,1 instead of 0 ... 1.0V) 1.0Volt)
- if the connected analog output EFB is unable to generate status information and the warning outputs can, therefore, never become active. In this case, please use the O_NORM Function block.

O_PHYS: Physical analog output

60

Overview

At a Glance

This chapter describes the block O_PHYS.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	278
Representation	278
Runtime error	278

Brief description

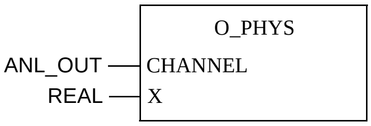
Function description

The Function block provides analog input values (voltage, current or temperature) as physical values in REAL floating-point format.
The function block is in output modules with configuration information (e.g. DAUs).
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output value
X	REAL	Physical value

Runtime error

Runtime error

An error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- with an output value underflow (for example, -1 Volt instead of 0 ... 5 Volt).
- with an output value overflow (for example, 6 Volt instead of 0 ... 5 Volt).

Note: To evaluate the status information for the I/O module, use the O_PHYS_WARNfunction block.

O_PHYS_WARN: Physical analog output with warning-status

61

Overview

At a Glance

This chapter describes the block O_PHYS_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	280
Representation	280
Runtime error	281

Brief description

Function description

The Function block provides analog input values (voltage, current or temperature) as physical values in REAL floating-point format.
The function block is used for output modules with configuration information (e.g. DAUs).
In addition the function block at the WARN_NEG and WARN_POS outputs indicate whether a status warning has occurred in the connected analog output EFB.

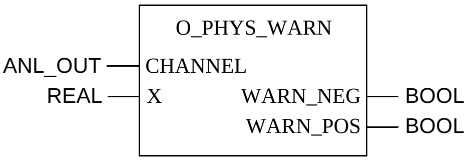
> **Note:** This function block is not compatible with the DAU2xx function for Compact (the O_PHYS function block should be used instead). The O_PHYS_WARN Function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output value
X	REAL	Physical value
WARN_NEG	BOOL	0: no output value underflow at the closed analog output EFB 1: output value underflow at the closed analog output EFB
WARN_POS	BOOL	0: no output value overflow at the closed analog output EFB 1: output value overflow at the closed analog output EFB

Runtime error

Runtime error

An Error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- with an output value underflow (outside the warning range, e.g. –1 Volt instead of 0 ... 5 Volt).
- with an output value overflow (outside the warning range, e.g. 6 Volt instead of 0 ... 5 Volt).
- if the connected analog output EFB is unable to generate status information and the warning outputs can, therefore, never become active. In this case, please use the O_PHYS Function block.

O_RAW: Raw value analog output

62

Overview

At a Glance

This chapter describes the block O_RAW.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	284
Representation	284
Runtime error	284

Brief description

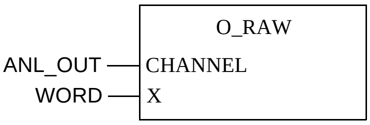
Function description

The Function block provides raw values of the WORD data type as analog output values.
EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
CHANNEL	ANL_OUT	Output value
X	WORD	Raw value

Runtime error

Runtime error

An error message is created if the input channel has not been configured. In this case, please check the connected I/O module EFB.

O_SCALE: Scaled analog output

63

Overview

At a Glance

This chapter describes the block O_SCALE.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	286
Representation	286
Runtime error	287

Brief description

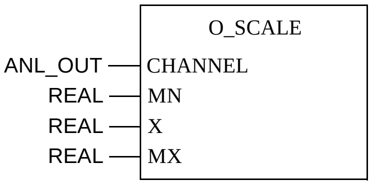
Function description

The Function block converts values from floating point formatREAL into 16 bit integer format. The scaling inputs MN and MX predefine the value range for the analog output. MN corresponds to 0 percent and MX to 100 percent of the output range (e.g. -10 ...)
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output value
MN	REAL	Scaling input, 0 percent
X	REAL	Floating-point value
MX	REAL	Scaling input, 100 percent

Runtime error

Runtime error

An error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- if the values of MN and MX are identical causing an internal module division by zero.
- with an output value underflow (for example, -1 Volt instead of 0 ... 5 Volt).
- with an output value overflow (for example, 6 Volt instead of 0 ... 5 Volt).

Note: To evaluate the status information for the I/O module, use the O_SCALE_WARN Function block.

O_SCALE_WARN: Scaled analog output with warnings status

64

Overview

At a Glance

This chapter describes the block O_SCALE_WARN.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	290
Representation	291
Runtime error	291

Brief description

Function description

The Function block converts values from floating point format REAL into 16 bit integer format. The scaling inputs MN and MX predefine the value range for the analog output. MN corresponds to 0 percent and MX to 100 percent of the output range (e.g. -10 ... 10 V).
In addition the function block at the WARN_NEG and WARN_POS outputs indicate whether a status warning has occurred in the connected analog output EFB.

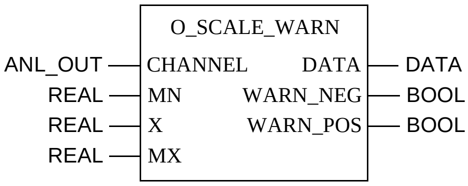
Note: This function block is not compatible with the DAU2xx function for Compact (the O_SCALE function block should be used instead). The O_SCALE_WARN Function block does not recognize the module range information, even though it is assigned to the 3x register area. Therefore, the area warn bits have to be taken directly.

The layout of the status information assigned to State RAM can be found in Online Help for Compact modules (**Help** → **Help on Compact** → **Compact I/O User's Guide**).
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
CHANNEL	ANL_OUT	Output value
MN	REAL	Scaling input, 0 percent
X	REAL	Floating-point value
MX	REAL	Scaling input, 100 percent
WARN_NEG	BOOL	0: no output value underflow at the closed analog output EFB 1: output value underflow at closed analog output EFB (X < MN)
WARN_POS	BOOL	0: no output value overflow at the closed analog output EFB 1: output value exceeded at closed analog output EFB (X > MX)

Runtime error

Runtime error

An error message appears,

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- if the values of MN and MX are identical causing an internal module division by zero.
- with an output value underflow (outside the warning range, e.g. –1 Volt instead of 0 ... 5 Volt).
- with an output value overflow (outside the warning range, e.g. 6 Volt instead of 0 ... 5 Volt).
- if the connected analog output EFB is unable to generate status information and the warning outputs can, therefore, never become active. In this case, please use the O_SCALE Function block.

O_SET: Set information from analog output channels

65

Overview

At a Glance

This chapter describes the block O_SET.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	294
Representation	295
Detailed description	296
Supported Value Ranges	298
Runtime error	299

Brief description

Function description

The function block sets the information for the analog output channels (ANL_OUT). This block enables all scaling blocks of this library to be used.

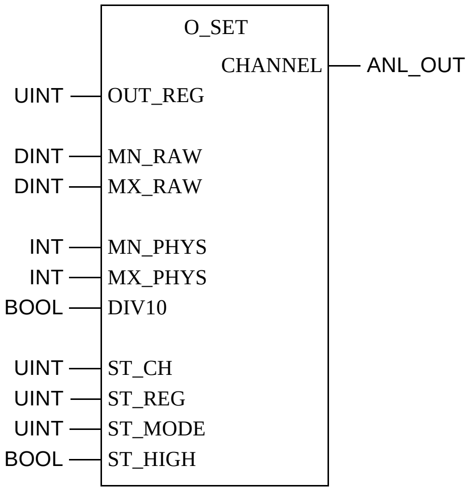
Note: The function block is only required if there is no specific block for a specific analog module available.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
OUT_REG	UINT	Number of the raw value register (4X)
MN_RAW	DINT	0 % raw value (e.g. 0)
MX_RAW	DINT	100% raw value (e.g. 4095)
MN_PHYS	INT	lowest output value (e.g. 0 V as 0)
MX_PHYS	INT	greatest output value (e.g. +10 V as 10)
DIV10	BOOL	MN_PHYS and MX_PHYS divided by 10
ST_CH	UINT	channel number (1n) (e.g. 4)
ST_REG	UINT	Number of the status register (3X)
ST_MODE	UINT	Status mode (e.g. 3=ACO_STATUS_MODE)
ST_HIGH	BOOL	Status byte found in highbyte of the register
CHANNEL	ANL_OUT	channel information to be described

Detailed description

Area of application

The function block can be used in three areas:
1. Raw value scaling, with the block: O_NORM and O_SCALE
2. Scaling in physical units, with the O_PHYS block
3. Evaluation of error information, with the blocks O_NORM, O_SCALE and O_PHYS and additional evaluation of status information (warnings) using the O_..._WARN block

Basic circuit connections

The input OUT_REG must always be connected with the number of an output word (4x).

Raw value scaling

For raw value scaling, the inputs MN_RAW (minimum raw value, corresponds to 0%) and MX_RAW (maximum raw value, corresponds to 100%) must also be connected.

Scaling in physical units

For scaling in physical units the inputs MN_PHYS and MX_PHYS must also be connected.
DIV10 is an auxiliary input in the range 0.2 V ... 1 V floating point format to avoid.
Set MN_PHYS=2, MX_PHYS=10 and DIV10=TRUE for this range.
For most ranges this input can remain open (or be assigned FALSE).
e.g. +/-20 mA: here is MN_PHYS=-20, MX_PHYS=20
The input value ranges supported by O_SET can be found in the section *Supported Value Ranges, p. 298*.

Evaluation of error information

For the evaluation of error information the inputs ST_CH, ST_REG and ST_MODE and ST_HIGH must also be configured.

ST_HIGH is an auxiliary input in case the status byte (error information) is located in the registers high byte. For most ranges this input can remain open (or be assigned FALSE).

The input channel number (1 ... n) is given to ST_CH.

If ST_CH is entered, ST_REG and ST_MODE must also be entered.

ST_REG must be connected with the number of an input word (3X), where the status information is located (error and/or warnings).

ST_MODE determines how the status word is evaluated.

The following 8 modes are defined:

Value	Mode	see also module description for
1	AVI_STATUS_MODE	AVI030
2	ACI_STATUS_MODE	ACI030
3	ACO_STATUS_MODE	ACO030
4	ADU_STATUS_MODE	ADU204
5	DAU204_STATUS_MODE	DAU204
6	ADU205_STATUS_MODE	ADU205
7	AMM090_STATUS_MODE	AMM090
8	ADU214_STATUS_MODE	ADU214

Supported Value Ranges

Voltage

Unipolar

Value range	MN_PHYS	MX_PHYS	DIV10
0 ... 0.5 V	0	5	1
0 ... 1.0 V	0	10	1
0 ... 5.0 V	0	5	0
0 ... 10 V	0	10	0
0 ... 20 V	0	20	0
0,1 ... 0.5 V	1	5	1
0,2 ... 1.0 V	2	10	1
1,0 ... 5.0 V	1	5	0
2,0 ... 10, 0 V	2	10	0

Bipolar

Value range	MN_PHYS	MX_PHYS	DIV10
+/- 25 mV	-25	25	0
+/-100 mV	-100	100	0
+/-0.5 V	-5	5	1
+/- 1 V	-1	1	0
+/-5 V	-5	5	0
+/-10 V	-10	10	0
+/-20 V	-20	20	0

Current

Unipolar

Value range	MN_PHYS	MX_PHYS	DIV10
0 .. 20 mA	0	20	0
4 ... 20 mA	4	20	0

Bipolar

Value range	MN_PHYS	MX_PHYS	DIV10
+/-20 mA	-20	20	0
+/-40 mA	-40	40	0

Runtime error

Runtime error

The following error messages can be triggered:

Error message	Meaning
E_EFB_USER_ERROR_1	The input OUT_REG is not connected with the number of an output word (4x).
E_EFB_USER_ERROR_2 with the parameters of the faulty number	The input OUT_REG is connected with an invalid number of an output word (4x).
E_EFB_USER_ERROR_3 with parameter MN_RAW	MN_RAW ≥ MX_RAW)
E_EFB_USER_ERROR_4 with parameter MN_PHYS	Unknown value for MN_PHYS
E_EFB_USER_ERROR_5 with parameter MX_PHYS	Unknown value for MX_PHYS
E_EFB_USER_ERROR_11	ST_REG not entered
E_EFB_USER_ERROR_12	ST_REG too large
E_EFB_USER_ERROR_13	ST_CH not entered

QPR_16I_12O: Configuring the TIO-module QPR 346 00 / 10 / 20 / 21

66

Overview

At a Glance

This chapter describes the block QPR_16I_12O.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	302
Representation	303
Detailed description	304

Brief description

Function description

The Function block QPR_16I_12O is a software connection for the following INTERBUS modules:

- TIO/IS 170 BAM 346 00
- TIO/IS 170 BAM 346 10
- TIO/IS 170 BAM 346 20
- TIO/IS 170 BAM 346 21

The function block has 16 binary inputs and 12 binary outputs, which can be operated simultaneously or just as inputs or outputs. In addition, the module can be programmed in ASCII code via the built-in RS 232 interface.
In an unprogrammed state, the module behaves in the same way as a TIO with 16 binary inputs and 12 binary outputs. In a programmed state, the internal QPR links have priority over signals created at the function block, i.e. the QPR outputs accept the value generated by the internal link. In this case, the output value indicated in Concept need not agree with the actual value in the QPR.

Example:

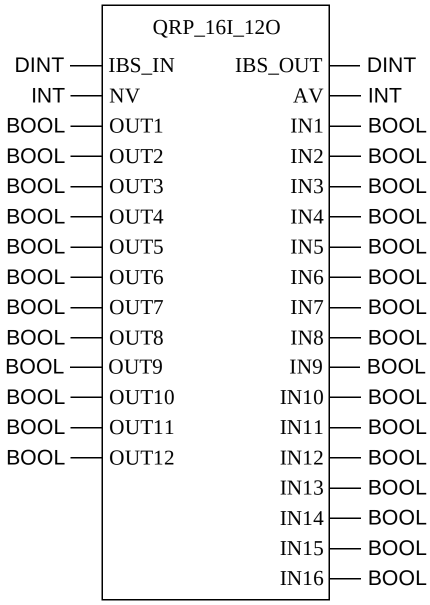
- With Concept, output OUT5 is set to "1".
- With the internal QPR link, output OUT5 is set to "0".
- The value at output 5 of the QPR is "0" in Concept, but a "1" is indicated
- Module programming is described in the user manual for TIO modules with preceding logic operation.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IBS_IN	DINT	Incoming INTERBUS
NV	INT	Nominal value
OUT1	BOOL	Output 1 of the TIO
OUT2	BOOL	Output 2 of the TIO
:	:	:
OUT12	BOOL	Output 12 of the TIO
IBS_OUT	DINT	Outgoing INTERBUS
AV	INT	Actual value
IN1	BOOL	Input 1 of the TIO
IN2	BOOL	Input 2 of the TIO
:	:	:
IN16	BOOL	Input 16 of the TIO

Detailed description

Detailed description

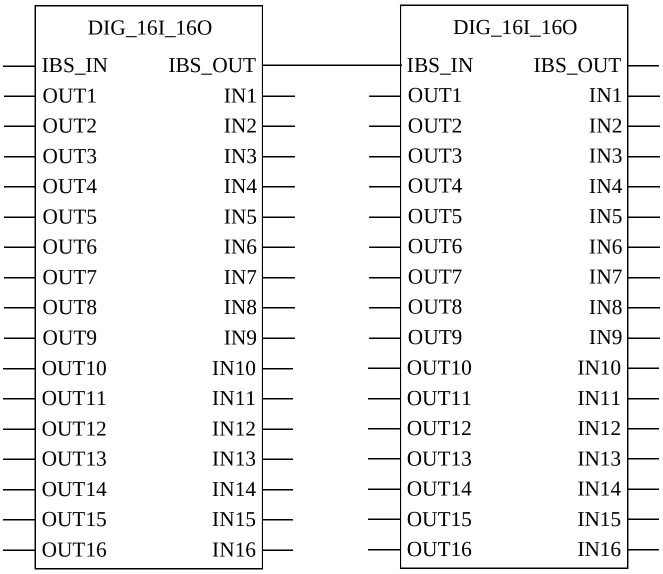
The QPR_16I_12O Function block in Concept functions in the same way as its hardware counterpart.
However, its operation has been simplified by programming it as a function block in Concept. The module occupies two input words and two output words in the master.

Note: The outputs programmed in the QPR cannot be represented by the QPR_16I_12O Function block, as the value is generated by the module itself. The PLC cannot influence the value which was generated by the module.

Parameter description - inputs

IBS_IN

IBS_IN = Connection for the incoming remote bus part of INTERBUS
On the hardware, the male connector is on the top left of the module.
The module is connected here to the outgoing remote bus (IBS_OUT) of the master (1st module on the bus) or the preceding module (see diagram). The link can be made via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two bus devices.
Connection of two INTERBUS modules



NV

NV = Nominal value
A number for the programmable counter or a time for a delay switch can be entered here. For the times, 1 = 1ms i.e., 30000 = 30s.

OUTx

OUTx = Output x
x stands for a number between 1 and 12 which refers to the corresponding output.
The binary values displayed by the INTERBUS module are supplied to the process via the relevant output (OUTx).

Parameter description - Outputs

IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
On the hardware, the male connector is on the top right of the module.
The module is connected to the incoming remote bus (IBS_IN) of the following module, either via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two INTERBUS modules.

AV

AV = Actual value
The actual number on a counter or the actual time of a delay switch is indicated at this output. For the times, 1 = 1ms i.e., 30000 = 30s.

INx

INx = Input x
x stands for the digit 1 to 16 which indicates the particular input.
Binary process values are read into the INTERBUS module via the relevant input (INx).

QUANTUM: Configuring a main rack

67

Overview

At a Glance

This chapter describes the block QUANTUM.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	308
Representation	308
Runtime error	309

Brief description

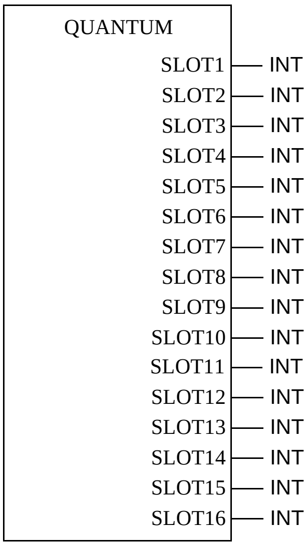
Function description

The Function block is used to edit the configuration data of a Quantum primary backplane for subsequent use by the scaling EFBs.
To configure a QUANTUM primary subrack, the QUANTUM Function block is inserted into the configuration section. The function blocks for the configuration of analog modules or the DROP Function block for the I/O station are connected at its SLOT outputs.
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
SLOT1	INT	Slot 1
:	:	:
SLOT16	INT	Slot 16

Runtime error

Runtime error

Internal I/O map errors will cause an Error message.

R_INT_WORD: Type conversion (REAL -> INT -> WORD)

68

Overview

At a Glance

This chapter describes the block R_INT_REAL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	312
Representation	312
Runtime error	312

Brief description

Function description

This Function block converts a input value from data type REAL to data type INT and subsequently to data type WORD.

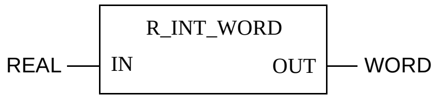
In contrast to the conversion block REAL_TO_WORD (IEC library), the R_INT_WORD block implements a conversion in INT value before the task of the REAL value. This results in the input value of –1.0, for example, being issued as an output value of FFFF (and not like the REAL_TO_WORD block which has an output value of 0).

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	WORD	Output value

Runtime error

Error message

An error message appears,
- an unauthorized floating point number is placed at the input,
- The value range of the data type INTis violated.

R_UINT_WORD: Type conversion (REAL -> UINT -> WORD)

69

Overview

At a Glance

This chapter describes the block R_UINT_WORD.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	314
Representation	314
Runtime error	314

Brief description

Function description

This Function block converts a input value from data type REAL to data type UINT and subsequently to data type WORD.

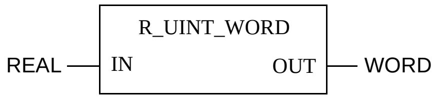
In contrast to the conversion block REAL_TO_WORD (IEC library), the R_UINT_WORD block implements a conversion in UINT value (value range 0.0 - 65535.5) before the output of the WORD value. This results in the input value of –1.0, for example, causes an error message, the output ENO is set and the output value is unchanged (and not like the REAL_TO_WORD block which has no error message and an output value of 0).

EN and ENO can be used as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	WORD	Output value

Runtime error

Error message

An error message appears, if
- an unauthorized floating point number is placed at the input,
- The value range of the data type UINT is violated.

SCALRTOW: Scaling (REAL -> WORD)

70

Overview

At a Glance

This chapter describes the block SCALRTOW.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	316
Representation	317
Runtime error	318

Brief description

Function description

The function scales a REAL-input value to a WORD-output value according to a given scale.

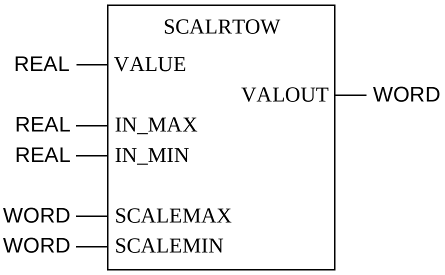
Note: The values for SCALEMAX and SCALEMIN are converted internally before the evaluation according to UINT.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Formulas

A linear scaling takes place according to the following formula:

$$\mathrm{VALOUT} = (\mathrm{VALUE} - \mathrm{IN\_MIN}) \times \frac{\mathrm{SCALEMAX} - \mathrm{SCALEMIN}}{\mathrm{IN\_MAX} - \mathrm{IN\_MIN}} + \mathrm{SCALEMIN}$$

Restrictions:
- If VALUE ≥ IN_MAX, then VALOUT = SCALEMAX.
- If VALUE ≤ IN_MIN, then VALOUT = SCALEMIN.

Note: The maximum value of VALOUT is 7FFF Hex (32767 dec.). If this maximum value is exceeded and error is returned and the ENO parameter (if configured) receives the value OFF.

Parameter description

Block parameter description:

Parameter	Data type	Meaning
VALUE	REAL	Input value
IN_MAX	REAL	Upper limit for input value
IN_MIN	REAL	Lower limit for input value
SCALEMAX	WORD	Upper limit for output value
SCALEMIN	WORD	Lower limit for output value
VALOUT	WORD	Output value

Runtime error

Error message

An error message appears, if

- invalid REAL-values are placed on the inputs. In this case the output value is not changed.
- scaling is invalid, e.g. SCALEMAX < SCALEMIN. In this case the output value is not changed.
- The value of the VALUE-input is not between the given values for IN_MAX and IN_MIN. In this case ENO is set to "0" and the output value is set either to the value of SCALEMAX or SCALEMIN, depending on which value has been violated.

SCALWTOR: Scaling (WORD -> REAL)

71

Overview

At a Glance

This chapter describes the block SCALWTOR.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	320
Representation	321
Runtime error	322

Brief description

Function description

The function scales a WORD-input value to a REAL-output value according to a given scale.

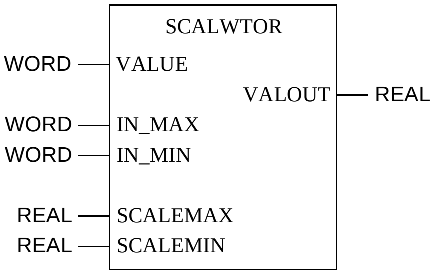
Note: The values for IN_MAX and IN_MIN are converted internally before the evaluation according to UINT

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Formulas

A linear scaling takes place according to the following formula:

$$\mathrm{VALOUT} = (\mathrm{VALUE} - \mathrm{IN\_MIN}) \times \frac{\mathrm{SCALEMAX} - \mathrm{SCALEMIN}}{\mathrm{IN\_MAX} - \mathrm{IN\_MIN}} + \mathrm{SCALEMIN}$$

Restrictions:

- If VALUE ≥ IN_MAX, then VALOUT = SCALEMAX.
- If VALUE ≤ IN_MIN, then VALOUT = SCALEMIN.

Parameter description

Block parameter description:

Parameter	Data type	Meaning
VALUE	WORD	Input value
IN_MAX	WORD	Upper limit for input value
IN_MIN	WORD	Lower limit for input value
SCALEMAX	REAL	Upper limit for output value
SCALEMIN	REAL	Lower limit for output value
VALOUT	REAL	Output value

Runtime error

Error message

An error message appears, if

- invalid REAL-values are placed on the inputs. In this case the output value is not changed.
- scaling is invalid, e.g. SCALEMAX < SCALEMIN. In this case the output value is not changed.
- The value of the VALUE-input is not between the given values for IN_MAX and IN_MIN. In this case ENO is set to "0" and the output value is set either to the value of SCALEMAX or SCALEMIN, depending on which value has been violated.

UNI_I: Configuring universal TIO input modules

72

Overview

At a Glance

This chapter describes the block UNI_I.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	324
Representation	324
Detailed description	325

Brief description

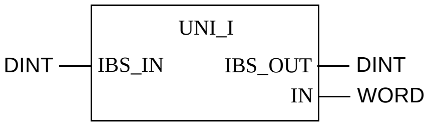
Function description

The UNI_I_O function block is a software connection to a universal INTERBUS hardware module (input only).
The function block has one output for this.
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IBS_IN	DINT	Incoming INTERBUS
IBS_OUT	DINT	Outgoing INTERBUS
IN	WORD	Input of an INTERBUS module

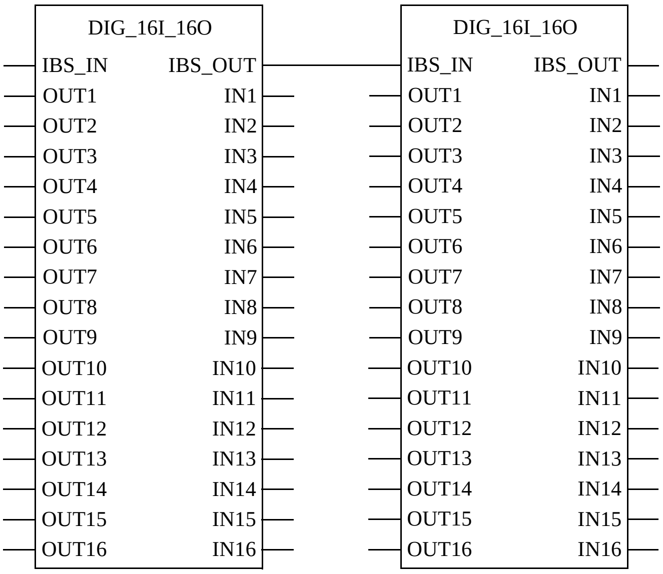
Detailed description

Detailed description

The function block occupies one input word in the Status-RAM.

IBS_IN

IBS_IN = Connection for the incoming remote bus part of INTERBUS
The module is connected here to the outgoing remote bus (IBS_OUT) of the master (1st module on the bus) or the preceding module (see diagram). The link can be made via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two bus devices.
Connection of two INTERBUS modules



IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
The module is connected to the incoming remote bus (IBS_IN) of the following module, either via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two INTERBUS modules.

IN

IN = Input
The input reads input information from the INTERBUS module in the form of a word.

UNI_I_O: Configuring universal TIO input/output modules

73

Overview

At a Glance

This chapter describes the block UNI_I_O.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	328
Representation	328
Detailed description	329

Brief description

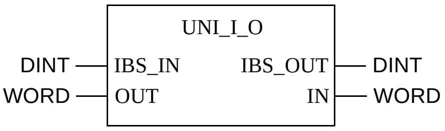
Function description

The UNI_I_O function block is a software connection to a universal INTERBUS hardware module (input/output).
The function block has one input and one output for this.
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IBS_IN	DINT	Incoming INTERBUS
OUT	WORD	Output of an INTERBUS module
IBS_OUT	DINT	Outgoing INTERBUS
IN	WORD	Input of an INTERBUS module

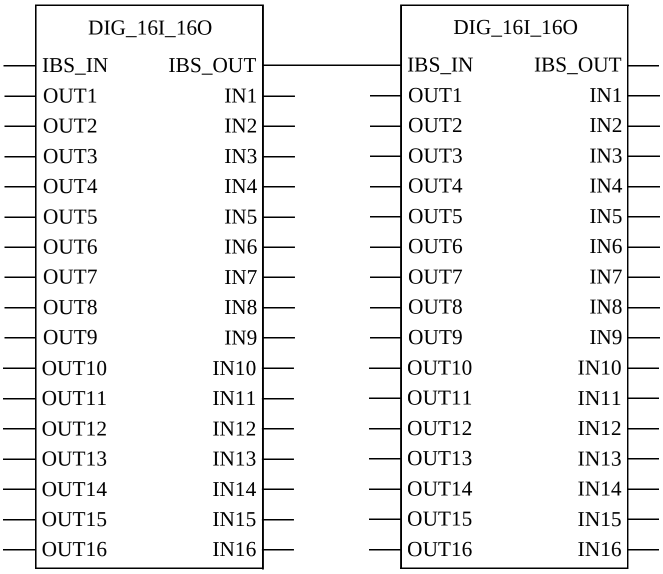
Detailed description

Detailed description

The function block occupies 1 input word and 1 output word in the Status-RAM.

IBS_IN

IBS_IN = Connection for the incoming remote bus part of INTERBUS
The module is connected here to the outgoing remote bus (IBS_OUT) of the master (1st module on the bus) or the preceding module (see diagram). The link can be made via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two bus devices.
Connection of two INTERBUS modules



OUT

OUT = Output
The output provides output information from the INTERBUS module in the form of a word.

IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
The module is connected to the incoming remote bus (IBS_IN) of the following module, either via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two INTERBUS modules.

IN = Input

IN = Input
The input reads input information from the INTERBUS module in the form of a word.

UNI_O: Configuring universal TIO output modules

74

Overview

At a Glance

This chapter describes the block UNI_O.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	332
Representation	332
Detailed description	333

Brief description

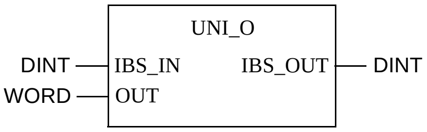
Function description

The UNI_I_O function block is a software connection to a universal INTERBUS hardware module (output only). The function block has one input for this.
EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IBS_IN	DINT	Incoming INTERBUS
OUT	WORD	Output of an INTERBUS module
IBS_OUT	DINT	Outgoing INTERBUS

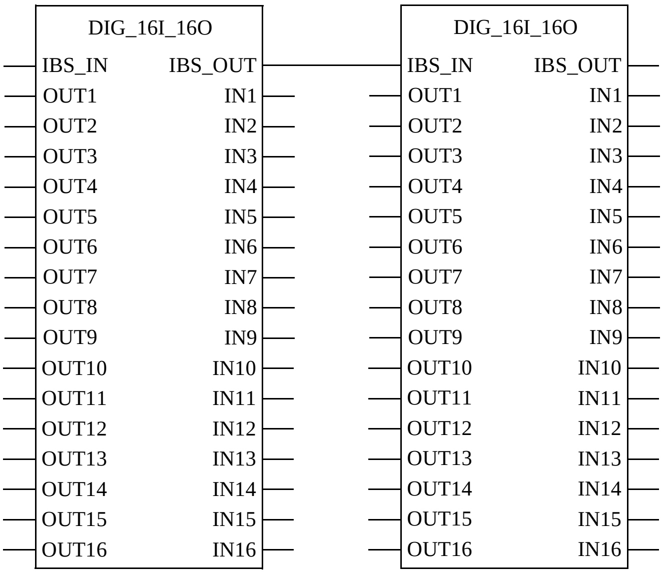
Detailed description

Detailed description

The function block occupies one output word in the Status-RAM.

IBS_IN

IBS_IN = Connection for the incoming remote bus part of INTERBUS
The module is connected here to the outgoing remote bus (IBS_OUT) of the master (1st module on the bus) or the preceding module (see diagram). The link can be made via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two bus devices.
Connection of two INTERBUS modules



OUT = Output
The output provides output information from the INTERBUS module in the form of a word.

IBS_OUT

IBS_OUT = Connection for the outgoing remote bus part of INTERBUS
The module is connected to the incoming remote bus (IBS_IN) of the following module, either via a line or via a variable. For the hardware, the type of connection corresponds to the INTERBUS cable between two INTERBUS modules.

W_INT_REAL: Type conversion (WORD -> INT -> REAL)

75

Overview

At a Glance

This chapter describes the block W_INT_REAL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	336
Representation	336

Brief description

Function description

This Function block converts a input value from data type WORD to data type INT and subsequently to data type REAL.

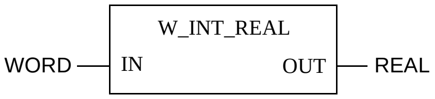
In contrast to the conversion block WORD_TO_REAL (IEC library), the W_INT_REAL block implements a conversion in INT value before the task of the REAL value. This results in the input value FFFF, for example, being issued as an output value of –1.0 (and not like the WORD_TO_REAL block which has an output value of 9.183409e-41).

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	WORD	Input value
OUT	REAL	Output value

W_UINT_REAL: Type conversion (WORD -> UINT -> REAL)

76

Overview

At a Glance

This chapter describes the block W_UINT_REAL.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	338
Representation	338

Brief description

Function description

This Function block converts a input value from data type WORD to data type UINT and subsequently to data type REAL.

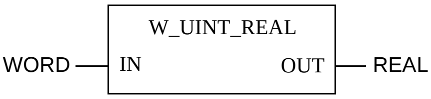
In contrast to the conversion block WORD_TO_REAL (IEC library), the W_INT_REAL block implements a conversion in UINT value before the task of the REAL value. This results in the input value FFFF, for example, being issued as an output value of –1.0 (and not like the WORD_TO_REAL block which has an output value of 9.183409e-41). This results in the input value FFFF, for example, being issued as an output value of 65535.0 (and not like the WORD_TO_REAL block which has an output value of 9.183409e-41).

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	WORD	Input value
OUT	REAL	Output value

XBP: Configuring a primary backplane expander

77

Overview

At a Glance

This chapter describes the block XBP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	340
Representation	341

Brief description

Function description

The Function block is used to edit the configuration data of a Quantum primary backplane expander for subsequent use by the scaling EFBs.

To configure a Quantum primary backplane expander, the XBP Function block is inserted into the configuration section (See *Procedure for expansion of the local backplane using XBE modules (Quantum), p. 13*). It is connected to its SLOT input at the corresponding SLOT x-output of the QUANTUM Function block. The function block for configuring the analog module is connected to the SLOT x-outputs.

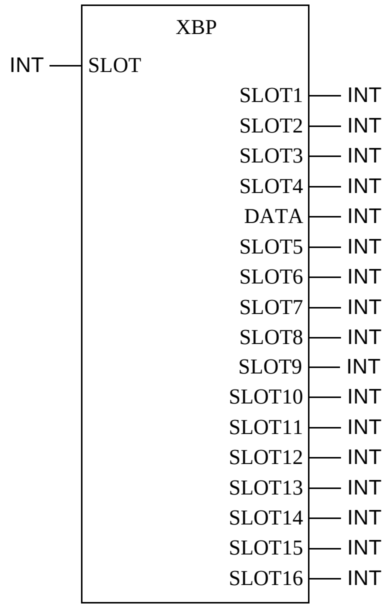
Note: The XBP function block is only used to configure **central** backplane expanders. To configure distributed expansions, use the function block XDROP (See *XDROP: Configuring a I/O Station Backplane , p. 343*).

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
SLOT	INT	140 XBE 100 00 slot in the central rack
SLOT1	INT	Slot 1
:	:	:
SLOT16	INT	Slot 16

XDROP: Configuring a I/O Station Backplane

78

Overview

At a Glance

This section describes function block XDROP.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Brief description	344
Representation	345
Runtime error	345

Brief description

Function description

The function block is used to prepare the configuration data of distributed I/O station for subsequent processing by module configuration EFBs.
To configure a expansion for an I/O station backplane, the SLOT input of XDROP function block is connected with the SLOT input of the DROP function block in the Configuration Section. The same number must be entered for the NUMBER input of the XDROP function block as for the NUMBER input of the DROP function block.
The Function blocks for configuration of the analog modules of the I/O stations are connected to the X_SLOT outputs.

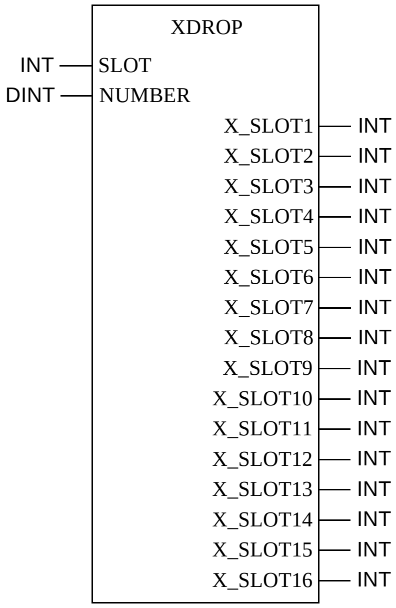
> **Note:** The XDROP function block is only used to configure expansions for distributed backplanes. To configure **central** backplane expanders, use the function block XBP (See *XBP: Configuring a primary backplane expander, p. 339*).

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

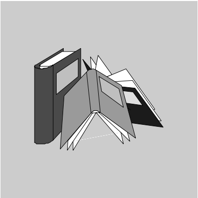
Parameters	Data type	Meaning
SLOT	INT	XBP slot in the distributed backplane
NUMBER	DINT	Number of the distributed station
X_SLOT1	INT	Expansion slot 1
:	:	:
X_SLOT16	INT	Expansion slot 16

Runtime error

Runtime error

If no "Head" has been configured for the I/O station backplane, an error message appears (E_EFB_NOT_CONFIGURED).

Glossary



A

active Window
The window, which is currently selected. Only one window can be active at any given time. When a window is active, the color of the title bar changes, so that it is distinguishable from the other windows. Unselected windows are inactive.

Actual Parameters
Current connected Input / Output Parameters.

Addresses
(Direct) addresses are memory ranges in the PLC. They are located in the State RAM and can be assigned Input/Output modules.
The display/entry 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 "Analog Input" data type and is used when processing analog values. The 3x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

ANL_OUT
ANL_OUT stands for the "Analog Output" data type and is used when processing analog values. The 4x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

ANY
In the above version "ANY" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD elementary data types and related Derived Data Types.

ANY_BIT In the above version "ANY_BIT" covers the BOOL, BYTE and WORD data types.

ANY_ELEM In the above version "ANY_ELEM" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD data types.

ANY_INT In the above version "ANY_INT" covers the DINT, INT, UDINT and UINT data types.

ANY_NUM In the above version "ANY_NUM" covers the DINT, INT, REAL, UDINT and UINT data types.

ANY_REAL In the above version "ANY_REAL" covers the REAL data type.

Application Window The window containing the workspace, menu bar and the tool bar for the application program. The name of the application program appears in the title bar. An application window can contain several Document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII-Mode The ASCII (American Standard Code for Information Interchange) mode is used to communicate 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 has a motherboard (requiring SA85 driver) with two slots for PC104 daughter-boards. In this way, one PC104 daughter-board is used as a CPU and the other as the INTERBUS controller.

B

Backup file (Concept-EFB) The backup file is a copy of the last Source coding file. The name of this backup file is "backup??.c" (this is assuming that you never have more than 100 copies of the source coding file). The first backup file has the name "backup00.c". If you have made alterations to the Definitions file, which do not cause any changes to the EFB interface, the generation of a backup file can be stopped by editing the source coding file (**Objects** → **Source**). If a backup file is created, the source file can be entered as the name.

Base 16 literals Base 16 literals are used to input whole number values into the hexadecimalsystem. The base must be denoted using the prefix 16#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

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

Base 2 literals Base 2 literals are used to input whole number values into the dualsystem. The base must be denoted using the prefix 2#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_0000 or 2#11100000 (decimal 224)

Base 8 literals Base 8 literals are used to input whole number values into the octosystem. The base must be denoted using the prefix 8#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
8#3_77 or 8#377 (decimal 255)
8#34_0 or 8#340 (decimal 224)

Binary Connections Connections between FFB outputs and inputs with the data type BOOL.

Bitsequence A data element, which consists of one or more bits.

BOOL BOOL stands for the data type "boolean". The length of the data element is 1 bit (occupies 1 byte in the memory). The value range for the variables of this data type is 0 (FALSE) and 1 (TRUE).

Bridge A bridge is a device, which connects networks. It enables communication between nodes on two networks. Each network has its own token rotation sequence - the token is not transmitted via the bridge.

BYTE BYTE stands for the data type "bit sequence 8". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 8 bits. A numerical value range can not be assigned to this data type.

C

Clipboard The clipboard is a temporary memory for cut or copied objects. These objects can be entered in sections. The contents of the clipboard are overwritten with each new cut or copy.

Coil A coil is a LD element which transfers the status of the horizontal short on its left side, unchanged, to the horizontal short on its right side. In doing this, the status is saved in the relevant variable/direct address.

Compact format (4:1) The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.

Constants Constants are Unlocated variables, which are allocated a value that cannot be modified by the logic program (write protected).

Contact A contact is a LD element, which transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address. A contact does not change the value of the relevant variable/direct address.

D

Data transfer settings Settings which determine how information is transferred from your programming device to the PLC.

Data Types The overview shows the data type hierarchy, as used for inputs and outputs of functions and function blocks. Generic data types are denoted using 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 drop

A remote network with a super-ordinate PLC can be controlled using a Distributed Control Processor (D908). When using a D908 with remote PLC, the super-ordinate PLC considers the remote PLC as a remote I/O drop. The D908 and the remote PLC communicate via the system bus, whereby a high performance is achieved with minimum effect on the cycle time. The data exchange between the D908 and the super-ordinate PLC takes place via the remote I/O bus at 1.5Mb per second. A super-ordinate PLC can support up to 31 D908 processors (addresses 2-32).

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs in Windows. The user can also use the DDE interface in the extended monitor to invoke their 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 to the PLC via the server. The user can therefore alter data directly in the PLC, while monitoring and analyzing results. When using this interface, the user can create their own "Graphic Tool", "Face Plate" or "Tuning Tool" and integrate into the system. The tools can be written in any language, i.e. Visual Basic, Visual C++, which supports DDE. The tools are invoked, when the user presses one of the buttons in the Extended Monitor dialog field. Concept Graphic Tool: Configuration signals can be displayed as a timing diagram using the DDE connection between Concept and Concept Graphic Tool.

Declaration

Mechanism for specifying the definition of a language element. A declaration usually covers the connection of an identifier to a language element and the assignment of attributes such as data types and algorithms.

Definitions file (Concept-EFB)

The definitions file contains general descriptive information on the selected EFB and its formal parameters.

Derived Data Type

Derived data types are data types, which are derived from Elementary Data Types and/or other derived data types. The definition of derived data types is found in the Concept data type editor.

A distinction is made between global data types and local data types.

Derived Function Block (DFB)

A derived function block represents the invocation of a derived function block type. Details of the graphic form of the invocation can be found in the "Functional block (instance)". In contrast to the invocation of EFB types, invocations of DFB types are denoted by double vertical lines on the left and right hand side of the rectangular block symbol.

The body of a derived function block type is designed using FBD language, LD language, ST language, IL language, however, this is only the case in the current version of the programming system. Furthermore, derived functions can not yet be defined in the current version.

A distinction is made between local and global DFBs.

Device Address The device address is used to uniquely denote a network device in the routing path. The address is set on the device directly, e.g. using the rotary switch on the back of the modules.

DFB Code The DFB code is the section’s DFB code, which can be executed. The size of the DFB code is mainly dependant upon the number of blocks in the section.

DFB instance data The DFB instance data is internal data from the derived function block used in the program.

DINT DINT stands for the data type "double length whole number (double integer)". Entries are made as integer literal, base 2 literal, basis 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this datatype reaches from -2 exp (31) to 2 exp (31) -1.

Direct Representation A method of displaying variables in the PLC program, from which the assignment to the logical memory can be directly - and indirectly to the physical memory - derived.

Document Window A window within an application window. Several document windows can be open at the same time in an application window. However, only one document window can ever be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.

DP (PROFIBUS) DP = Remote Peripheral

Dummy An empty file, which consists of a text heading with general file information, such as author, date of creation, EFB designation etc. The user must complete this dummy file with further entries.

DX Zoom This property enables the user to connect to a programming object, to monitor and, if necessary change, its data value.

E

EFB code The EFB code is the section’s EFB code, which can be executed. In addition the used EFBs count in DFBs.

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 body for example can not be modified with the DFB editor (Concept-DFB). EFB types are programmed in "C" and are prepared in a pre-compiled form using libraries.

EN / ENO (Enable / Error signal) If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is in this case automatically set to "0". If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After the error-free execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during the execution of these algorithms, ENO is automatically set to "0". The output behavior of the FFB is independent of whether the FFBs are invoked without EN/ENO or with EN=1. If the EN/ENO display is switched on, it is imperative that the EN input is switched on. Otherwise, the FFB is not executed. The configuration of EN and ENO is switched on or off in the Block Properties dialog box. The dialog box can be invoked with the **Objects** → **Properties...** menu command or by double-clicking on the FFB.

Error If an error is recognized during the processing of a FFB or a step (e.g. unauthorized input values or a time error), an error message appears, which can be seen using the **Online** → **Event Viewer...** menu command. For FFBs, the ENO output is now set to "0".

Evaluation The process, through which a value is transmitted for a Function or for the output of a Function block during Program execution.

F

FFB (Functions/ Function blocks) Collective term for EFB (elementary functions/function blocks) and DFB (Derived function blocks)

Field variables A variable, which is allocated a defined derived data type with the key word ARRAY (field). A field is a collection of data elements with the same data type.

FIR Filter (Finite Impulse Response Filter) a filter with finite impulse answer

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 supplies an exact data element when processing. a function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.
Details of the graphic form of the function invocation can be found in the "Functional block (instance)". In contrast to the invocation of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique number via the graphic block, this number is automatically generated and can not be altered.

Function block (Instance) (FB)

A function block is a program organization unit, which correspondingly calculates the functionality values that were defined in the function block type description, for the outputs and internal variable(s), if it is invoked as a certain instance. All internal variable and output values for a certain function block instance remain from one function block invocation to the next. Multiple invocations of the same function block instance with the same arguments (input parameter values) do not therefore necessarily supply the same output value(s).
Each function block instance is displayed graphically using a rectangular block symbol. The name of the function block type is stated in the top center of the rectangle. The name of the function block instance is also stated at the top, but outside of the rectangle. It is automatically generated when creating an instance, but, depending on the user's requirements, it can be altered by the user. Inputs are displayed on the left side of the block and outputs are displayed on the right side. The names of the formal input/output parameters are shown inside the rectangle in the corresponding places.
The above description of the graphic display is especially applicable to the function invocation and to DFB invocations. Differences are outlined 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, divided into input, output and internal variables; 2. a set of operations, which are performed with elements of the data structure, when a function block type instance is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (invoked) several times.

Function Number

The function number is used to uniquely denote a function in a program or DFB. The function number can not be edited and is automatically assigned. The function number is always formed as follows: .n.m

n = section number (current number)
m = Number of the FFB object in the section (current number)

G

Generic Data Type
A data type, which stands in place of several other data types.

Generic literals
If the literal’s data type is not relevant, simply specify the value for the literal. If this is the case, Concept automatically assigns the literal a suitable data type.

Global Data
Global data are Unlocated variables.

Global derived data types
Global derived data types are available in each Concept project and are occupied in the DFB directory directly under the Concept directory.

Global DFBs
Global DFBs are available in each Concept project. The storage of the global DFBs is dependant upon the settings in the CONCEPT.INI file.

Global macros
Global macros are available in each Concept project and are occupied in the DFB directory directly under the Concept directory.

Groups (EFBs)
Some EFB libraries (e.g. the IEC library) are divided into groups. This facilitates EFB location especially in expansive libraries.

H

Host Computer
Hardware and software, which support programming, configuring, testing, operating and error searching in the PLC application as well as in a remote system application, in order to enable source documentation and archiving. The programming device can also be possibly used for the display of the process.

I

I/O Map
The I/O and expert modules from the various CPUs are configured in the I/O map.

Icon
Graphical representation of different objects in Windows, e.g. drives, application programs and document windows.

IEC 61131-3
International standard: Programmable Logic Controls - Part 3: Programming languages.

IEC Format (QW1)
There is an IEC type designation in initial position of the address, followed by the five-figure address.

- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier)
An identifier is a sequence of letters, numbers and underscores, which must begin with either a letter or underscore (i.e. the name of a function block type, an instance, a variable or a section). Letters of a national typeface (i.e.: ö,ü, é, õ) can be used, except in project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as two separate identifiers. Several leading and multiple successive underscores are not allowed.
Identifiers should not contain any spaces. No differentiation is made between upper and lower case, e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers should not be Keywords.

IEC Program Memory
The IEC memory consists of the program code, EFB code, the section data and the DFB instance data.

IIR Filter
(Infinite Impulse Response Filter) a filter with infinite impulse answer

Initial step
The first step in a sequence. A step must be defined as an initial step for each sequence. The sequence is started with the initial step when first invoked.

Initial value
The value, which is allocated to a variable when the program is started. The values are assigned in the form of literals.

Input bits (1x references)
The 1/0 status of the input bits is controlled via the process data, which reaches from an input device to the CPU.

Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 100201 signifies an output or marker bit at the address 201 in the State RAM.

Input parameter (Input)
Upon invocation of a FFB, this transfers the corresponding argument.

Input words (3x references) An input word contains information, which originates from an external source and is represented by a 16 bit number. 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 follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 300201 signifies an input word at the address 201 in the State RAM.

Input/output marker bits (0x references) An input/output marker bit can be used to control real output data using an output unit of the control system, or to define one or more discrete outputs in the state RAM. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 000201 signifies an output or marker bit at the address 201 in the State RAM.

Instance Name An identifier, which belongs to a certain function block instance. The instance name is used to clearly denote a function block within a program organization unit. The instance name is automatically generated, but it can be edited. The instance name must be unique throughout the whole program organization unit, and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears. The automatically generated instance name is always formed as follows: FBI_n_m

FBI = Function Block Instance
n = section number (current number)
m = Number of the FFB object in the section (current number)

Instancing Generating an Instance.

Instruction (IL) Instructions are the "commands" of the IL programming language. Each instruction begins on a new line and is performed by an operator with a modifier if necessary, and if required for the current operation, by one or more operands. If several operands are used, they are separated by commas. A character can come before the instruction, which is then followed by a colon. The commentary must, where available, be the last element of the line.

Instruction (LL984) When programming electrical controls, the user should implement operation-coded instructions in the form of picture objects, which are divided into a recognizable contact form. The designed program objects are, on a user level, converted to computer usable OP codes during the download process. The OP codes are decoded in the CPU and processed by the firmware functions of the controller in a way that the required control is implemented.

Instruction (ST) Instructions are the "commands" of the ST programming language. Instructions must be concluded by semicolons. Several instructions can be entered in one line (separated by semicolons).

Instruction list (IL)

IL is a text language according to IEC 1131, which is shown in operations, i.e. conditional or unconditional invocations of Functions blocks and Functions, conditional or unconditional jumps etc. through instructions.

INT

INT stands for the data type "whole number (integer)". Entries are made as integer literal, base 2 literal, basis 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this datatype reaches from -2 exp (15) to 2 exp (15) -1.

Integer literals

Integer literals are used to input whole number values into the decimalsystem. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP)

The new INTERBUS (PCP) I/O drop type is entered into the Concept configurator, to allow use of the INTERBUS PCP channel and the INTERBUS process data pre-processing (PDV). This I/O drop type is assigned the INTERBUS switching module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only in the fact that it has a clearly larger I/O range in the control state RAM.

Invocation

The process, through which an operation is carried out.

J

Jump

Element of the SFC language. Jumps are used to skip zones in the sequence.

K

Keywords

Keywords are unique combinations of characters, which are used as special syntactical components, as defined in Appendix B of the IEC 1131-3. All keywords which are used in the IEC 1131-3 and therefore in Concept, are listed in Appendix C of the IEC 1131-3. These keywords may not be used for any other purpose, i.e. not as variable names, section names, instance names etc.

L

Ladder Diagram (LD)
Ladder Diagram is a graphic programming dialog according to IEC1131, which is optically oriented to the "rung" of a relay contact plan.

Ladder Logic 984 (LL)
The terms Ladder Logic and Ladder Diagram refer to the word Ladder being executed. In contrast to a circuit diagram, a ladder diagram is used by electrotechnicians to display an electrical circuit (using electrical symbols), which should show the course of events and not the existing wires, which connect the parts with each other. A usual user interface for controlling the actions of automation devices permits a Ladder Diagram interface, so that electrotechnicians do not have to learn new programming languages to be able to implement a control program.
The structure of the actual Ladder Diagram enables the connection of electric elements in such a way that generates a control output, which is dependant upon a logical power flow through used electrical objects, which displays the previously requested condition of a physical electrical device.
In simple form, the user interface is a video display processed by the PLC programming application, which sets up vertical and horizontal grid, in which programming objects are classified. The diagram contains the power grid on the left side, and when connected to activated objects, the power shifts from left to right.

Landscape
Landscape means that when looking at the printed text, the page is wider than it is high.

Language Element
Every basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable.

Library
Collection of software objects, which are intended for re-use when programming new projects, or even building new libraries. Examples are the libraries of the Elementary function block types.
EFB libraries can be divided up into Groups.

Link
A control or data flow connection between graphical objects (e.g. steps in the SFC Editor, function blocks in the FBD Editor) within a section, represented graphically as a line.

Literals
Literals are used to provide FFB inputs, and transition conditions etc using direct values. These values can not be overwritten by the program logic (read only). A distinction is made between generic and standardized literals.
Literals are also used to allocate a constant, a value or a variable an initial value. Entries are made as base 2 literal, base 8 literal, basis 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 the local DFBs and are placed in the DFB directory under the project directory.

Local DFBs Local DFBs are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local Link The local network is the network, which connects the local nodes with other nodes either directly or through bus repeaters.

Local macros Local macros are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local network nodes The local node is the one, which is currently being configured.

Located variable A state RAM address (reference addresses 0x, 1x, 3x,4x) is allocated to located variables. The value of these variables is saved in the state RAM and can be modified online using the reference data editor. These variables can be addresses using their symbolic names or their reference addresses.

All inputs and outputs of the PLC are connected to the state RAM. The program can only access peripheral signals attached to the PLC via located variables. External access via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems, is also possible via located variables.

M

Macro

Macros are created with the help of the Concept DFB software.
Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).
A distinction is made between local and global macros.

Macros have the following properties:
- Macros can only be created in the FBD and LD programming languages.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- DFB invocation in a macro
- Declaring variables
- Using macro-specific data structures
- Automatic transfer of the variables declared in the macro.
- Initial value for variables
- Multiple instancing of a macro in the entire program with differing variables
- The name of the section, variable names and data structure names can contain up to 10 different exchange marks (@0 to @9).

MMI

Man-Machine-Interface

Multi element variables

Variables to which a Derived data type defined with STRUCT or ARRAY is allocated.
A distinction is made here between field variables and structured variables.

N

Network

A network is the collective switching of devices to a common data path, which then communicate with each other using a common protocol.

Network node

A node is a device with an address (1...64) on the Modbus Plus network.

Node

Node is a programming cell in a LL984 network. A cell/node consists of a 7x11 matrix, i.e. 7 rows of 11 elements.

O

Operand An operand is a literal, a variable, a function invocation or an expression.

Operator An operator is a symbol for an arithmetic or boolean operation, which is to be carried out.

Output parameter (outputs): A parameter, through which the result(s) of the evaluation of a FFB is/are returned.

Output/marker words (4x references) An output / marker word can be used to save numerical data (binary or decimal) in the state RAM, or to send data from the CPU to an output unit in the control system. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

P

Peer CPU The Peer CPU processes the token execution and the data flow between the Modbus Plus network and the PLC user logic.

PLC Memory programmable controller

Portrait Portrait means that the sides are larger than the width when printed.

Print-out Expressions consist of operators and operands.

Program The uppermost program organization unit. A program is closed on a single PLC download.

Program organization unit A function, a function block, or a Program. This term can refer to either a type or an instance.

Program redundancy system (Hot Standby) A redundancy system consists of two identically configured PLC machines, which communicate with one another via redundancy processors. In the case of a breakdown of the primary PLC, the secondary PLC takes over the control check. Under normal conditions, the secondary PLC does not take over the control function, but checks the status information, in order to detect errors.

Project

General description for the highest level of a software tree structure, which specifies the super-ordinate project name of a PLC application. After specifying the project name you can save your system configuration and your control program under this name. All data that is created whilst setting up the configuration and program, belongs to this super-ordinate project for this specific automation task.
General description for the complete set of programming and configuration information in the project database, which represents the source code that describes the automation of a system.

Project database

The database in the host computer, which contains the configuration information for a project.

Prototype file (Concept-EFB)

The prototype file contains all the prototypes of the assigned functions. In addition, if one exists, a type definition of the internal status structure is specified.

R

REAL

REAL stands for the data type "floating point number". The entry can be real-literal or real-literal with an exponent. The length of the data element is 32 bits. The value range for variables of this data type extends from +/- 3.402823E+38.

> **Note:** Dependent on the mathematical processor type of the CPU, different ranges within this permissable value range cannot be represented. This applies to values that are approaching ZERO and for values that approach INFINITY. In these cases NAN (**N**ot **A N**umber) or INF (**INF**inite will be displayed in the animation mode instead of a number value.

Real literals

Real literals are used to input floating point values into the decimal system. Real literals are denoted by a decimal point. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literals with exponents

Real literals with exponents are used to input floating point values into the decimal system. Real literals with exponents are identifiable by a decimal point. The exponent indicates the power of ten, with which the existing number needs to be multiplied in order to obtain the value to be represented. The base can have a preceding negative sign (-). The exponent can have a preceding positive or negative sign (+/-). Single underscores (_) between numbers are not significant. (Only between numbers, not before or after the decimal point 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

Every direct address is a reference that begins with an indicator, which specifies whether it is an input or an output and whether it is a bit or a word. References that begin with the code 6, represent registers in the extended memory of the state RAM.
0x range = Coils
1x range = Discrete inputs
3x range = Input registers
4x range = Output registers
6x range = Register in the extended memory

> **Note:** The x, which follows each initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

Register in the extended memory (6x-reference)

6x references are holding registers in the extended memory of the PLC. They can only be used with LL984 user programs and only with a CPU 213 04 or CPU 424 02.

Remote Network (DIO)

Remote programming in the Modbus Plus network enables maximum performance when transferring data and dispenses of the need for connections. Programming a remote network is simple. Setting up a network does not require any additional ladder logic to be created. All requirements for data transfer are fulfilled via corresponding entries in the Peer Cop Processor.

RIO (Remote I/O)

Remote I/O indicates a physical location of the I/O point controlling devices with regard to the CPU controlling them. Remote inp./outputs are connected to the controlling device via a twisted communication cable.

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.

Runtime error

Errors, which appear during program processing on the PLC, in SFC objects (e.g. Steps) or FFBs. These are, for example, value range overflows with figures or timing errors with steps.

S

SA85 module The SA85 module is a Modbus Plus adapter for IBM-AT or compatible computers.

Scan A scan consists of reading the inputs, processing the program logic and outputting the outputs.

Section A section can for example be used to describe the mode of functioning of a technological unit such as a motor.
A program or DFB consists of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages may be used within a section at any one time.
Each section has its own document window in Concept. For reasons of clarity, it is however useful to divide a very large section into several small ones. The scroll bar is used for scrolling within a section.

Section Code Section Code is the executable code of a section. The size of the Section Code is mainly dependent upon the number of blocks in the section.

Section Data Section data is the local data in a section such as e.g. literals, connections between blocks, non-connected block inputs and outputs, internal status memory of EFBs.

Note: Data which appears in the DFBs of this section is not section data.

Separator Format (4:00001) The first digit (the reference) is separated from the five figure address that follows by a colon (:).

Sequence language (SFC) The SFC Language Elements enable a PLC program organization unit to be divided up into a number of Steps and Transitions, which are connected using directional Links. A number of actions belong to each step, and transition conditions are attached to each transition.

Serial Connections With serial connections (COM) the information is transferred bit by bit.

Source code file (Concept-EFB) The source code file is a normal C++ source file. After executing the **Library** → **Create files** menu command, this file contains an EFB-code frame, in which you have to enter a specific code for the EFB selected. To do this invoke the **Objects** → **Source** menu command.

Standard Format (400001) The five figure address comes directly after the first digit (the reference).

Standardized literals

If you would like to manually determine a literal’s data type, this may be done using the following construction: ’Data type name’#’value of the literal’.

Example
INT#15 (Data type: integer, value: 15),
BYTE#00001111 (Data type: byte, value: 00001111)
REAL#23.0 (Data type: real, value: 23.0)

To assign the data type REAL, the value may also be specified in the following manner: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the memory space for all variables, which are accessed via References (Direct representation) in the user program. For example, discrete inputs, coils, input registers, and output registers are situated in the state RAM.

Status Bits

For every device with global inputs or specific inp./outputs of Peer Cop data, there is a status bit. If a defined group of data has been successfully transferred within the timeout that has been set, the corresponding status bit is set to 1. If this is not the case, this bit is set to 0 and all the data belonging to this group is deleted (to 0).

Step

SFC-language element: Situation, in which the behavior of a program occurs, regarding its inputs and outputs of those operations which are defined by the actions belonging to the step.

Step name

The step name is used to uniquely denote a step in a program organization unit. The step name is generated automatically, but it can be edited. The step name must be unique within the entire program organization unit, otherwise an error message will appear.
The automatically generated step name is always formed as follows: S_n_m

S = step
n = section number (current number)
m = Number of the step in the section (current number)

Structured text (ST)

ST is a text language according to IEC 1131, in which operations, e.g. invocations of Function blocks and Functions, conditional execution of instructions, repetitions of instructions etc. are represented by instructions.

Structured variables

Variables to which a Derived data type defined with STRUCT (structure) is allocated. A structure is a collection of data elements with generally different data types (elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept includes the providing of I/O-map SY/MAX-I/O modules for remote contolling by the Quantum PLC. The SY/MAX remote backplane 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 executed for you for labelling and inclusion in the I/O map of the Concept configuration.

T

Template file (Concept-EFB) The template file is an ASCII file with layout information for the Concept FBD Editor, and the parameters for code creation.

TIME TIME stands for the data type "time". The entry is time literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1. The unit for the TIME data type is 1 ms.

Time literals Permissable units for times (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or combinations of these. The time must be marked with the prefix t#, T#, time# or TIME#. The "overflow" of the unit with the highest value is permissible, e.g. the entry T#25H15M is allowed.

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 possession of the transfer right via a single device. The token passes round the devices in a rotating (increasing) address sequence. All devices follow the token rotation and can receive all the possible data that is sent with it.

Total IEC memory The total IEC memory consists of the IEC program memory and the global data.

Traffic Cop The traffic cop is an IO map, which is generated from the user-IO map. The traffic cop is managed in the PLC and in addition to the user IO map, contains e.g. status information on the I/O stations and modules.

Transition The condition, in which the control of one or more predecessor steps passes to one or more successor steps along a directed link.

U

UDEFB

User-defined elementary functions/function blocks
Functions or function blocks, which were created in the C programming language, and which Concept provides in libraries.

UDINT

UDINT stands for the data type "unsigned double integer". Entries are made as integer literal, base 2 literal, basis 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1.

UINT

UINT stands for the data type "unsigned integer". Entries are made as integer literal, base 2 literal, basis 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this data type extends from 0 to (2exp 16)-1.

Unlocated variable

Unlocated variables are not allocated a state RAM address. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the internal system and can be changed using the reference data editor. These variables are only addressed using their symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc., should be primarily declared as unlocated variables.

V

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.
Variables consist of at least one variable name and one data type.
If a variable is assigned a direct address (reference), it is called a located variable.
If the variable has no direct address assigned to it, it is called an unlocated variable.
If the variable is assigned with a derived data type, it is called a multi element variable.
There are also constants and literals.

W

Warning If a critical status is detected during the processing of a FFB or a step (e.g. critical input values or an exceeded time limit), a warning appears, which can be seen using the **Online** → **Event Viewer...** menu command. For FFBs, the ENO remains set to "1".

WORD WORD stands for the data type "bit sequence 16". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. A numerical value range can not be assigned to this data type.

Index

A

ACI030, 33
ACI040, 37
ACO020, 41
ACO130, 45
ADU204, 49
ADU205, 53
ADU206, 57
ADU214, 61
AII330, 65
AII33010, 69
AIO330, 73
AMM090, 77
ANA_16I, 81
ANA_4I_2O, 95
ANA_4I_2O_C, 105
ANA_4I_2O_V, 109
ANA_4I_M, 87
ANA_4O, 113
ANA_8I, 119
ANA_IO
  ACI030, 33, 37
  ACO020, 41
  ACO130, 45
  ADU204, 49
  ADU205, 53
  ADU206, 57
  ADU214, 61
  AII330, 65
  AII33010, 69
  AIO330, 73
  AMM090, 77
  ANA_16I, 81
  ANA_4I_2O, 95
  ANA_4I_2O_C, 105
  ANA_4I_2O_V, 109
  ANA_4I_M, 87
  ANA_4O, 113
  ANA_8I, 119
  ARI030, 125
  ATI030, 129
  AVI030, 133
  AVO020, 137
  BKF_201, 139
  BNO_671, 147
  COMPACT, 153
  DAU202, 157
  DAU204, 161
  DAU208, 165
  DIG_16I, 169
  DIG_16I_12O_MON, 175
  DIG_16I_16O, 181
  DIG_16O, 187
  DROP, 193
  I_DBSET, 197
  I_DEBUG, 199
  I_FILTER, 201
  I_NORM, 207
  I_NORM_WARN, 209
  I_PHYS, 213
  I_PHYS_WARN, 215
  I_RAW, 219
  I_RAWSIM, 221
  I_SCALE, 223

I_SCALE_WARN, 227
I_SET, 231
IMIO_IN, 239
IMIO_OUT, 243
MIX_4I_2O, 247
NOA_611, 255
O_DBSET, 261
O_DEBUG, 263
O_FILTER, 265
O_NORM, 271
O_NORM_WARN, 273
O_PHYS, 277
O_PHYS_WARN, 279
O_RAW, 283
O_SCALE, 285
O_SCALE_WARN, 289
O_SET, 293
QPR_16I_12O, 301
QUANTUM, 307
R_INT_WORD, 311
R_UINT_WORD, 313
SCALRTOW, 315
SCALWTOR, 319
UNI_I, 323
UNI_I_O, 327
UNI_O, 331
W_INT_REAL, 335
W_UINT_REAL, 337
XBP, 339
XDROP, 343

Analog IO Config
- I_FILTER, 201
- I_SET, 231
- O_FILTER, 265
- O_SET, 293

Analog IO Debug
- I_DBSET, 197
- I_DEBUG, 199
- O_DBSET, 261
- O_DEBUG, 263

Analog IO Scaling
- I_NORM, 207
- I_NORM_WARN, 209
- I_PHYS, 213
- I_PHYS_WARN, 215
- I_RAW, 219
- I_RAWSIM, 221
- I_SCALE, 223
- I_SCALE_WARN, 227
- O_NORM, 271
- O_NORM_WARN, 273
- O_PHYS, 277
- O_PHYS_WARN, 279
- O_RAW, 283
- O_SCALE, 285
- O_SCALE_WARN, 289
- R_INT_WORD, 311
- R_UINT_WORD, 313
- SCALRTOW, 315
- SCALWTOR, 319
- W_INT_REAL, 335
- W_UINT_REAL, 337

Analog value processing Momentum, 23
- Example, 26
- Procedure, 24

ARI030, 125
ATI030, 129
AVI030, 133
AVO020, 137

B

BKF_201, 139
BNO_671, 147

C

COMPACT, 153
Compact IO Config
- ADU214, 61
- COMPACT, 153
- DAU202, 157
- DAU204, 161
- DAU208, 165

Compact IO config
 ADU204, 49
 ADU205, 53
 ADU206, 57
Configuring a I/O Station subrack, 193
Configuring a primary backplane expander, 339
Configuring an I/O Station Backplane, 343
Configuring module AAI 140 00, 81
Configuring the AMM 090 00 module, 247
Configuring the Compact module ADU 204, 49
Configuring the Compact module ADU 205, 53
Configuring the Compact module ADU 206, 57
Configuring the Compact Module ADU 214, 61
Configuring the Compact module BKF 201, 139
Configuring the Compact module DAU 202 / DAU 252 / DAU 282, 157
Configuring the Compact module DAU 204, 161
Configuring the Compact module DAU 208, 165
Configuring the main rack, 153, 307
Configuring the module AAI 030 00, BAI 036 00, 119
Configuring the module AAI 520 40, 87
Configuring the module ADM 390 10, 175
Configuring the module BAO 126 00, 113
Configuring the Quantum module ACI 030 00, 33
Configuring the Quantum module ACI 040 00, 37
Configuring the Quantum module ACO 020 00, 41
Configuring the Quantum module ACO 130 00, 45
Configuring the Quantum module AII 330 00, 65
Configuring the Quantum module AII 330 10, 69
Configuring the Quantum module AIO 330 00, 73
Configuring the Quantum module AMM 090 00, 77
Configuring the Quantum module ARI 030 10, 125
Configuring the Quantum module ATI 030 00, 129
Configuring the Quantum module AVI 030 00, 133
Configuring the Quantum module AVO 020 00, 137
Configuring the Quantum modules NOA 611 00/NOA 611 10, 255
Configuring the TIO-mode BAM 096 00, 105, 109
Configuring the TIO-module BAM 096 00, 95
Configuring the TIO-module BDI 346 00 / 546 50 / 746 50, 169
Configuring the TIO-module BDM 346 00, 181
Configuring the TIO-module BNO 671 00, 147
Configuring the TIO-module QPR 346 00 / 10 / 20 / 21, 301
Configuring the TIO-mopdule BDO 346 00 / BDO 946 50, 187
Configuring universal TIO input modules, 323
Configuring universal TIO input/output modules, 327
Configuring universal TIO output modules, 331

D

DAU202, 157
DAU204, 161
DAU208, 165
DIG_16I, 169
DIG_16I_12O_MON, 175
DIG_16I_16O, 181
DIG_16O, 187
DROP, 193

F

Function
    Parameterization, 3
Function block
    Parameterization, 3

I

I_DBSET, 197
I_DEBUG, 199
I_FILTER, 201
I_NORM, 207
I_NORM_WARN, 209
I_PHYS, 213
I_PHYS_WARN, 215
I_RAW, 219
I_RAWSIM, 221
I_SCALE, 223
I_SCALE_WARN, 227
I_SET, 231
IBS
    ANA_16I, 81
    ANA_4I_2O, 95
    ANA_4I_2O_C, 105
    ANA_4I_2O_V, 109
    ANA_4I_M, 87
    ANA_4O, 113
    ANA_8I, 119
    BKF_201, 139
    BNO_671, 147
    DIG_16I, 169
    DIG_16I_12O_MON, 175
    DIG_16I_16O, 181
    DIG_16O, 187
    MIX_4I_2O, 247
    NOA_611, 255
    QPR_16I_12O, 301
    UNI_I, 323
    UNI_I_O, 327
    UNI_O, 331
IMIO
    IMIO_IN, 239
    IMIO_OUT, 243
IMIO_IN, 239
IMIO_OUT, 243
Immediate I/O module input, 239
Immediate I/O module output, 243

L

Linearization for analog outputs, 265
Linearization for analog-inputs, 201

M

MIX_4I_2O, 247
Monitor internal data structure ANL_OUT, 263
Monitoring internal data structure ANL_IN, 199

N

NOA_611, 255

O

O_DBSET, 261
O_DEBUG, 263
O_FILTER, 265
O_NORM, 271
O_NORM_WARN, 273
O_PHYS, 277
O_PHYS_WARN, 279
O_RAW, 283
O_SCALE, 285
O_SCALE_WARN, 289
O_SET, 293

P

Parameterization, 3
Physical analog output, 277
Physical analog output with warning status, 279
Physical analog-input, 213
Physical analog-input with warnings status, 215

Q

QPR_16I_12O, 301
QUANTUM, 307
Quantum IO Config, 339
- ACI030, 33
- ACI040, 37
- ACO130, 45
- AII330, 65
- AII33010, 69
- AIO330, 73
- ARI030, 125
- ATI030, 129
- AVI030, 133
- AVO020, 137
- DROP, 193
- QUANTUM, 307
- XDROP, 343

Quantum IO config
- ACO020, 41
- AMM090, 77

R

R_INT_WORD, 311
R_UINT_WORD, 313
Raw value analog input, 219
Raw value analog output, 283

S

Scaled analog input, 223
Scaled analog input with warning status, 227
Scaled analog output, 285
Scaled analog output with warning status, 289
Scaling (REAL -> WORD), 315
Scaling (WORD -> REAL), 319
SCALRTOW, 315
SCALWTOR, 319
Set information from analog input channels, 231
Set information from analog output channels, 293
Simulated raw value analog input, 221
Standardized analog input with warning status, 209
Standardized analog output, 271
Standardized analog output with warning status, 273
Standardized analog-input, 207

T

Type Conversion (REAL -> INT -> WORD), 311
Type Conversion (REAL -> UINT -> WORD), 313
Type Conversion (WORD -> INT -> REAL), 335
Type Conversion (WORD -> UINT -> REAL), 337

U

UNI_I, 323
UNI_I_O, 327
UNI_O, 331

W

W_INT_REAL, 335
W_UINT_REAL, 337
Write internal data structure ANL_OUT, 261
Writing internal data structure ANL_IN, 197

X

XBP, 339
XDROP, 343